Contribution to automatic text classification : metrics and evolutionary algorithms

Ahmad Mazyad

Contributions to Automatic Text Classification: Metrics and Evolutionary Algorithms

Ahmad Mazyad

Supervisors: Prof. Cyril Fonlupt
MCF Fabien Teytaud

Reviewers: Prof. Adnan Yassine
DR Evelyne Lutton
MCF, HDR Pierre Parrend

LISIC - Laboratoire d'Informatique Signal et Image de la Côte d'Opale
Université du Littoral Côte d'Opale

This dissertation is submitted for the degree of
PhD. in Computer Science

OSMOSE November 2018

Contents

Nomenclature

Acronyms / Abbreviations

CF	Collection Frequency
χ^2	Chi-Squared
DT	Decision Tree
FS	Feature Selection
GP	Genetic Programming
GR	Gain Ratio
ICF	Inverse Category Frequency
IDF	Inverse Document Frequency
IGB	Information Gain Based
IG	Information Gain
NC	Nearest Centroid
NER	Named Entity Recognition
OR	Odds Ratio
PA	Passive-Aggressive
POS	Parts-Of-Speech

RF Relevance Frequency

RIDGE Ridge Classifiifier

SGD Stochastic Gradient Descent

STW Supervised Term-Weighting

SVM Support Vector Machine

TBRS Term-Based Random Sampling

TC Text Classification

TF-CF Term Frequency-Collection Frequency

TF-χ^2 Term Frequency-Chi Squared

TF-GR Term Frequency-Gain Ratio

TF-ICF Term Frequency-Inverse Category Frequency

TF-IDF Term Frequency-Inverse Document Frequency

TF-IG Term Frequency-Information Gain

TF-OR Term Frequency-Odds Ratio

TF-RF Term Frequency-Relevance Frequency

TF Term Frequency

TWS Term-Weighting Scheme

VSM Vector Space Model

1

Introduction

Every day, an enormous amount of electronic data is generated. These data could be numbers, text, graphic, audio, video and many other types of data. Data contains knowledge of different types (e.g., medical, statistical, financial or scientific). This knowledge is virtually important to all fields (e.g., business, science, and industry).

Therefore, the ability to automatically and efficiently extract interesting knowledge from these data is becoming crucial.

Textual data hold an essential share of the daily generated data. For instance, each day, 734 million comments are posted on Facebook[1], 656 million tweets are sent[2], 864 thousand new Wikipedia page edits are published[2]. Also, the number of emails sent each day is 205 billion[3]. In 2015, the average number of business emails received per user per day totaled 88 emails, and it is expected to grow to an average of 96 messages in 2019. With this massive number of messages received

[1] https://www.domo.com/blog/data-never-sleeps-3-0/
[2] https://www.domo.com/learn/data-never-sleeps-5
[3] http://www.radicati.com/?p=12960.

on a daily basis, it becomes more and more difficult for users to manage their email accounts, e.g., sorting, selection .

Text mining (also called text data mining) is a subfield of data mining which is a computational process of identifying useful and understandable knowledge from a dataset. It uses methods that comprise different fields such as statistics, machine learning, natural language processing, optimization and database technology [1; 2; 3].

Text mining [1] process usually involves several phases. The first phase in the mining process is about preparing textual data for the training phase. This phase has two goals, First, it aims to present the textual data in a format compatible with the learning model. And secondly, it applies different techniques that improve the text mining efficiency. For instance, Feature Selection (FS) and stemming reduce the number of features, hence, they cut the computational cost. Text preprocessing includes many different techniques. Tokenization is the first step to be applied. It transforms a free text document (e.g., an email, a tweet or a news document) into a vector of tokens. Case folding is another preprocessing technique. It maps all letters to lower case.

Stemming and lemmatization are techniques that aim to map different forms of a word to a single one. Stop-words removal is the task of removing common words that have no discriminative power. At first sight, those techniques could seem trivial. However, they include many challenges such as the handling of hyphenation and acronyms in tokenization. All the techniques mentioned above are presented in details in Chapter 2.

Text representation [3; 4] is a vital step in text mining in which text documents are represented according to the learning model requirements. At this point, the representation level (e.g., character level, word level) and the representation model (e.g., Vector Space Model (VSM), Bag-of-Words (BoW) model, n-gram model) are chosen. Term-weighting is also an important step which assigns scores to terms using Term Weighting Scheme (TWS) [5]. TWS assigns high scores for keywords and low scores for unuseful terms. Several TWSs have been proposed such as Term Frequency-Inverse Document Frequency (TF-IDF) and its variants, Information Gain (*IG*), Chi-squared (χ^2), Odds Ratio (*OR*) and many others. These TWS could

be organized into two groups: supervised TWS and unsupervised TWS, according to whether they make use of available information on the documents memberships. TWSs are used in filter-based feature selection (see Section 2.3) such as χ^2 and *IG* [6]. They are also used as a way to detect stopwords [7]. Term-weighting also improves the model performance by assigning higher scores to essential terms [8; 9]. TWSs are generally adopted from information retrieval and statistics. They are generated according to human a priori and mathematical rules. They are usually simple mathematical expressions. Unfortunately, depending on the application, it is not easy to know a priori which TWS will be effective. In Chapter 5, we discuss and propose an automatic way to generate specialized TWSs via Genetic Programming (GP).

FS [6; 10] is another important technique in which a set of the most useful terms are selected. All other terms are filtered out. The FS techniques are adopted for three reasons: simplification of constructed models, shorter training times and enhanced generalization by reducing overfitting (reducing of variance). Processing techniques also include Parts-Of-Speech (POS) tagging, Named Entity Recognition (NER), and many others. Depending on the task, different steps may improve or hurts the text mining results. In Text Classification (TC), it is shown that case folding improves the classification performance [11]. However, it hurts the performance in sentiment analysis [12]. All the above preprocessing techniques are presented and discussed in Chapter 2.

The second phase focuses on the mining functions (models). The performance of the mining model may be improved by tuning the model parameters. The choice of the model depends on the application. Each application has its own set of learning algorithms. Mining task may be organized into two groups: the supervised learning tasks and the unsupervised learning tasks. Each group has a different type of algorithms. For instance, the k-means algorithm could be applied for unsupervised clustering such as social network analysis and image segmentation and Support Vector Machine (SVM) is fit for supervised learning such as TC and sentiment analysis. Section 2.5 of Chapter 2 presents seven different classification algorithms including SVM, Decision Tree (DT) and bagging and boosting.

Finally, in the last phase, models are evaluated in order to assess their performance. Several evaluation procedures and metrics are presented in Section 2.6 of Chapter 2.

Typical text mining tasks include TC, text clustering, document retrieval, language identification, authorship identification, concept/entity extraction, sentiment analysis, document summarization, and entity relation modeling (i.e., learning relations between named entities).

TC is a supervised learning task that aims to automatically assign a set of predefined categories to a text document. This task is achieved by constructing a model from a set of text documents.

A fundamental step in constructing a model (learning a classifier) is to represent text documents in a suitable format recognizable by this classifier. In text preprocessing, text documents are tokenized creating a bag of words (features/unique terms, also called grammar). Features in TC are mainly words, but could also be n-grams [13; 14; 15] (n consecutive terms). VSM is also used, so each text document is represented as a vector of index terms in which each term is associated with a weight (score) that measures how informative/discriminative the correspondent term is. Weights are computed by a TWS. Filtering, stemming, cleansing, stopword removal are also performed. They aim at reducing the computational cost by reducing the number of unique terms and consequently the number of dimensions of the vector space. These techniques may also in some cases (e.g., case folding) improve the accuracy of the classifier.

Finally, a classifier is trained using different supervised learning models and then evaluated. Some of the well known and efficient learning models are C4.5, Random Forest (RF), and SVM.

The classification problem may be a binary task, a multi-class task or multi-label task. A multi-label classification task is generally transformed into multiple binary single-label tasks. This transformation known as Binary relevance strategy is the most popular in TC. However, it introduces two issues. First, the terms distribution are only considered in terms of positive and negative category and secondly, the strategy does not consider the label dependency. In Chapter 4, we propose a new TWS based on the *IG* scheme that, to some extent, solves the first point, without impacting the complexity of the weighting task.

2

Text Classification

Text Classification (TC) and generally all text mining tasks usually involve a sequence of steps (see Figure 2.1):

- Text preprocessing such as tokenizing, filtering (stopword removal), stemming, cleansing ...
- Feature Selection (FS): select useful features
- Feature Weighting: giving a score for each feature
- Data Mining/Pattern Discovery: creating/training a model
- Interpretation/Evaluation: analyzing results

This chapter is organized as follows: Section 2.1 presents the different preprocessing techniques. It also discusses the impact of each one on the TC task. In Section 2.2, the representation of text are discussed. Section 2.3 presents the different filter selection techniques which are organized in three groups: filter-based

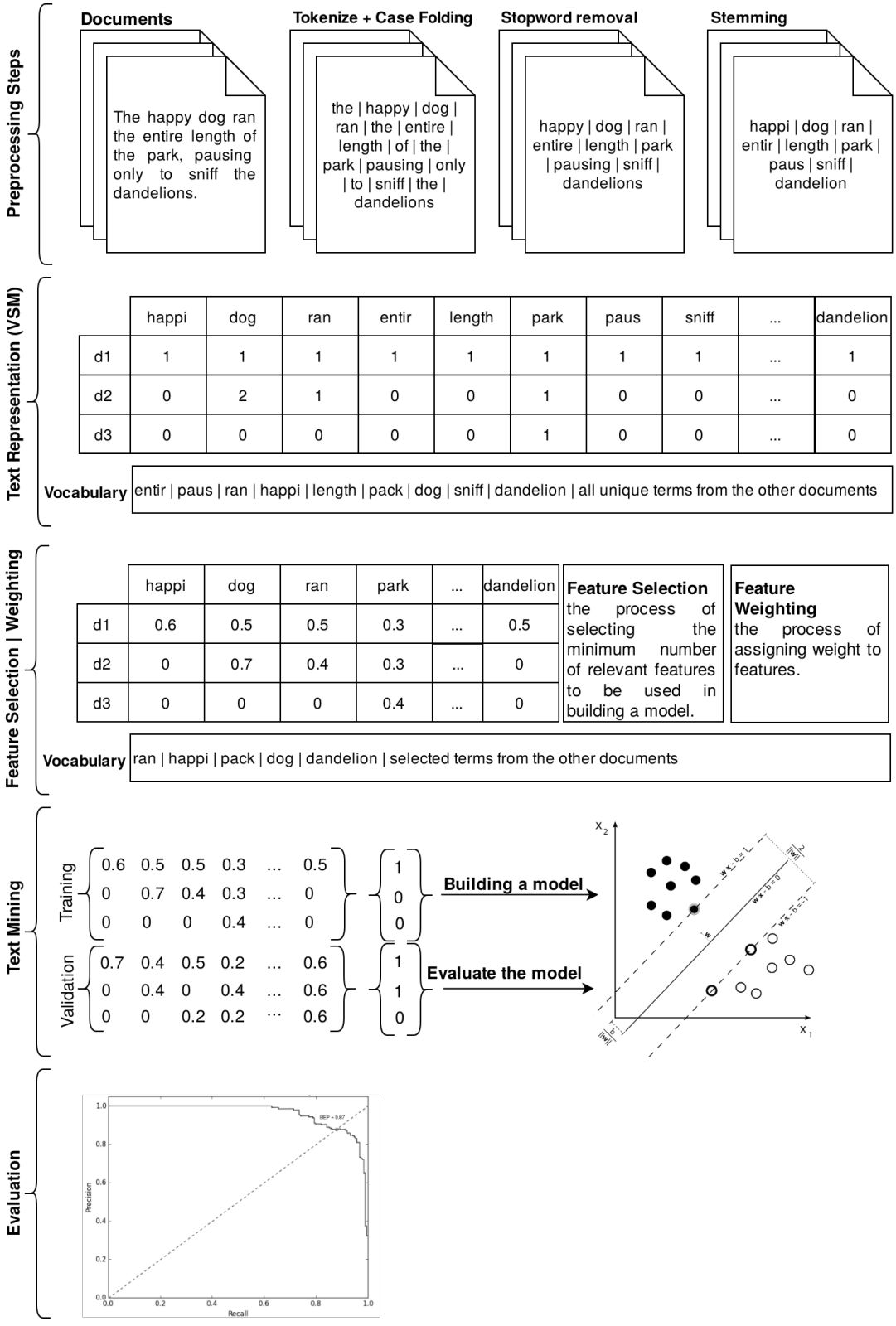

Figure 2.1 Steps of the text classification process.

methods, wrapper methods, and embedded methods. Section 2.4 very briefly defines term-weighting. Section 2.5 presents seven different classification algorithms chosen to assess the impact of term-weighting on the classification performance. And finally, Section 2.6 presents and discusses the evalution process.

2.1 Preprocessing Techniques

Before going into details about the preprocessing techniques, it is important to define three important notions: Word, Term, and Token. A word is a delimited string of characters in a text and a term is a normalized word (see Section 2.1.2), whereas a token is an instance of a term or a word occurring in a document.

2.1.1 Tokenization

Tokenization is the process of splitting text into tokens. It is typically the first process in any natural language processing application. However, it is directly impacted by the choice of the representation model and the representation level (character, word, Parts-Of-Speech (POS), phrase) (see Section 2.2). Tokens are often space delimited alphabetic strings (word level representation), but they can be numbers, punctuation, alpha-numerics, etc.
Tokenization is generally considered a trivial task, especially when considering languages such as English where words are separated by white space characters (segmented languages). However, a thorough examination reveals many obstacles that should be handled such as accronyms, abbreviations and hyphenation.

Language Dependent Challenges

Text in some languages such as Chinese, Japanese and Thai is seen a sequence of characters without spaces in between. In such languages, to obtain words, particular methods should be applied.
Furthermore, about every language have specificities that should be considered independently. For instance, in Arabic, pronouns are typically attached to verbs (the sentence "I ate" is translated into one single word in Arabic).

Accronyms and Abbreviations

In most languages, a period generally indicates the end of a sentence. Typically, it is considered as an independent token. However, when attached to abbreviations (e.g., dr., mr., .fr, etc.) or when occurring in acronyms (e.g., N.B., U.S.A, P.S., etc.), periods are, in these cases, an integral part of the token.

Hyphenation

Hyphenation presents another challenge for tokenization algorithms. The challenge is to address the question 'should the hyphenated word be considered as a single token or as multiple tokens?'
In English, hyphens are used for diverse purposes ranging from attaching prefixes to words especially when the prefix ends with a vowel (co-editor) to showing word breaks (two-, three-, or fourfold; dis-abled at the end of the line) to forming compounds (well-known, sugar-free, break-in, twenty-two, California-based).
Breaking up all hyphenated words in disregard of the hyphenation type and of the task may hurt the learning precision. For some cases, it is easy to see that the hyphenated word should be considered as single tokens (dis-abled, twenty-two, co-editor, etc.). However, some cases are unclear and should be addressed by considering the task. For instance, on the one hand, in Named Entity Recognition (NER), in the hyphenated word "California-based", the "California" part should be treated separately in order to be recognized as a location entity, however, on the other hand, in part-of-speech tag, the same hyphenated word "California-based" is preferably considered as one single token in order to be tagged correctly.

Named Entity Recognition (NER)

In Natural Language Processing (NLP), named entities are information units relevant to a specific application, like names (person, location, organization, real-world objects, etc.), numeric expressions (time, date, money, etc.), bibliographic references, etc.
Named entities are generally composed of multiple words, e.g., "Université du Littoral Côte d'Opale" or "Équipe de France de Football". Hence, splitting up words

based on white spaces is not an option and may break up entities, losing valuable information, helpful in various machine learning tasks.
The problem is, therefore, detecting named entities in order to tokenize them as single tokens. However, detecting named entities is not an easy task for several reasons:

- In many cases, a solution such as using dictionaries could not be considered due to the enormous numbers of entities.
- New named entities are continually appearing [16] such as the name of a new president.
- Some named entities have hundreds of variants. For example, the name Muammar Gaddafi have 413 variants and the name Mikhail Saakashvili have 256 variants [16].
- Named Entities might be abbreviated (U.S.A, USA, US, etc.).

Numerical Cases

A trivial parser can hurt mining or retrieval results when splitting what should be regarded as a single token.

- Dates (June 15, 2018).
- Phone numbers (+33 06 11 11 11 11).
- Time (04:16 am).
- Social security number, credit card number, etc.
- URIs, URLs, Email addresses, IP addresses, etc.

Comparison of Tokenization Techniques

Few researchers have compared the tokenization techniques on TC performance. Two tokenizing schemas ([:blank:]) and ([:blank:] and [:punct:]) have been inspected for SpamAssassin corpora on spam filtering tasks in [17]. It was stated

that the choice of the tokenizing schema is relevant in the context of spam filtering. The influence of tokenization, specifically the use of bigrams compared to unigrams, have been analyzed on three corpora (ModApte split of Reuters-21578, 20Newsgroup (see Subsection 4.2.1), and Springer) for TC in [18]. In the study, bigrams poorly impacted the results on both Reuters and 20Newsgroup corpora. However, it outperformed unigrams on Springer. Authors assumed that bigram tokenization is only useful for collection with long documents. Finally, in [11], alphabetic and alphanumeric tokenization have been compared in English and Turkish languages. It was stated that the alphabetic tokenizer should always be applied in the Turkish language independently of the domain. In the English language, mixed results were reported.

2.1.2 Normalization

Text normalization refers to a set of tasks that transform text into a more standard form.
Depending on the level of text representation (see Section 2.2), and the mining application, different techniques could be applied. Besides stemming, lemmatization and case folding, normalization tasks include spelling correction, values formatting, accents and acronyms processing.

Stemming

Sproat in [19] defined a stem as a morphological unit to which an affix attaches. Stemming is the process of reducing inflected or derived form to its original form by removing the affixes attached to it (e.g., CLASSES, CLASSED, CLASSIFY, CLASSIFICATION, etc.) are all inflected forms of the stem 'CLASS'.
In Natural Language Processing (NLP), many stemming algorithms have been proposed primarily for the English language. These stemming algorithms cut off the end of words in the hope that related words map to the same stem. Technically, the stem does not need to be a correct word.
Stemming algorithms could be classified into three types of algorithms:

- Rule-based algorithms such as Lovins stemmer, Porter stemmer, Paice/Husk stemmer, Dawson stemmer.
- Statistical-based algorithms such as N-Gram stemmer, Hidden Markov Model and Yet Another Suffix Stripper (YASS).
- Mixed algorithms such as Krovetz stemmer, corpus-based stemmer, and context-based stemmer.

Lovins stemmer, a rule-based algorithm, is the first popular stemmer proposed in [20]. However, other algorithms less known have already been designed such as the algorithm developed by Michael Lesk and presented in [21].
The Lovins stemmer includes a list of 294 endings compiled from multiple sources (i.e. endings list used at Harvard and augmented catalog developed by Project Intrex). Furthermore, the algorithm includes twenty-nine conditions associated with specific endings and thirty-five transformational rules used in converting stem terminations (Recoding procedure).
In the first step, the algorithm tries to match the longest ending that satisfies the associated condition code ("the longest-match principle"). The matched ending is removed. In the second step, the algorithm iterate over the thirty-five transformational rules and applying the relevant ones.
For example, the ending "ation" is the longest ending in the word "station". However, the ending associated condition ("Minimum stem length = 3") is not satisfied and therefore, the second longest ending "ion" which in this case, fulfill the condition ("Minimum stem length = 3 and do not remove ending after l or n") is removed, leaving "stat". No transformational rule could be applied to "stat", and therefore, based on Lovins algorithm, the word "station" is stemmed into "stat".

Porter stemmer is the most famous and most used stemming algorithm. It is one of the oldest stemming algorithms, and it supports a wide range of languages[1] and most importantly, it is the most accurate algorithm [22]. Like Lovins stemmer, Porter stemmer is a rule-based algorithm.
The algorithm was first proposed in [23; 24; 25, chap. 6]. Since then, the algorithm has been derived developing two new algorithms that could be found in [26; 27].

[1] https://tartarus.org/martin/PorterStemmer/

The Porter algorithm adopts a suffix stripping approach. The algorithm includes a list of suffixes where each suffix is associated with a condition that governs its removal (transformation).
The association of a suffix and a condition is called a transformation rule. The transformational rules are of the form "(condition) S1 → S2" where a suffix *S1* is transformed into a new suffix *S2*.
Here are some examples:

- (m>0) ATOR → ATE: the suffix *ALLI* should be transformed into *AL* if the stem contains at least one word part, i.e., one or more vowels followed by one or more consonants (e.g., operator → operate).

- (m>1 and (*S or *T)) ION → : the suffix *ION* should be removed if the stem contains one word part and ends with *S* or *T*.

The algorithm follows five steps [24] where suffixes are removed gradually[2]. In each step, a set of transformational rules are tested. Once a rule is successfully applied, the algorithm advances to the next step. So the word *relationalities* is transformed into *relationaliti* (Step 1), then stripped to *relational* (Step 2), and finally to *relation* (Step 4). The stem does not match any rule in the two other steps (Step 3 and 5).

Paice shows that light stemmers such as Porter stemming algorithm increases false negatives and hence, he proposes the Lancaster stemming algorithm (also called Paice/Husk stemmer) [28; 29]. Lancaster stemming algorithm is an iterative rule-based algorithm with 115 rules that try to match and apply, one rule at a time, according to the final letter of the word until no match could be found or the last stem is obtained. The algorithm could be very aggressive. For instance, the word *nationalism* is transformed into *nat*. *nationalism* is first stripped to *national* (msi3> -ism → -), then to *nation* (la2> -al → -) and finally to *nat* (noi> -ion → -). Paice/Husk stemmer is considered very aggressive and therefore, it is more prone to over-stemming [22].

Dawson stemmer is an extension of Lovins stemmer. In [30], Dawson modifies the Lovins stemming in two ways. On the one hand, in Dawson's algorithm, only

[2]http://snowball.tartarus.org/algorithms/porter/stemmer.html

the partial-matching procedure is used to conflate words, hence, discarding the recording procedure with the thirty-five transformational rules. On the other hand, the list of endings is greatly extended from 294 endings to 1200 endings.

The rule-based algorithms are language-dependent. For instance, the above-presented algorithms are developed for English. Many other rule-based algorithms exist for English [31] and other languages such as Arabic [32; 33; 34], Croatian [35], French [36; 37], Hindi [38; 39], Indonesian [40], etc. Porter presented a series of stemming algorithms for Russian, for German and Dutch, for the Romance languages Spanish, French, Portuguese and Italian, for Norwegian and Danish, and for Finnish [41].

In order to overcome the shortfall of language-dependency in rule-based algorithms, researchers proposed algorithms based on statistical information.
The single n-gram stemming algorithm [42] belongs to the statistical-based algorithms, and therefore, it is language independent. The algorithm relies on the fact that morphological affixes attached to stem are stripped in some of the derived character n-grams. For instance, the stem *nation* could be derived from *nations*, *national*, *nationality,* and *nationalism* by using 5-grams.
The algorithm selects the word-internal n-gram with the highest Inverse Document Frequency (IDF) which measure the rarity of a term across all documents reasoning that the morphological affixes occur often and around many different words, and hence, they carry low IDF.

[43]proposed an unsupervised model based on Hidden Markov Models that generates statistical stemmers. In this work, authors used three topologies of the Hidden Markov Model (see Figure 2.2) where states are divided into two disjoint sets, a prefix set of states that generate a first part of the word (the stem) and a suffix set that generates the last part of the word. To obtain the stem of a given word, the method computes the Viterbi path, i.e., the most probable sequence of HMM states, corresponding to the this word. This path is then analyzed in order to detect the point where a transition occurs from a state of the prefix set to a state of the suffix set. The stem is, therefore, the sequence of characters generated by the states of the prefix set.

Like the single n-gram stemming algorithm and the HMM-based algorithm, Yet Another Suffix Stripper (YASS) [44] is a statistical-based stemming algorithm and

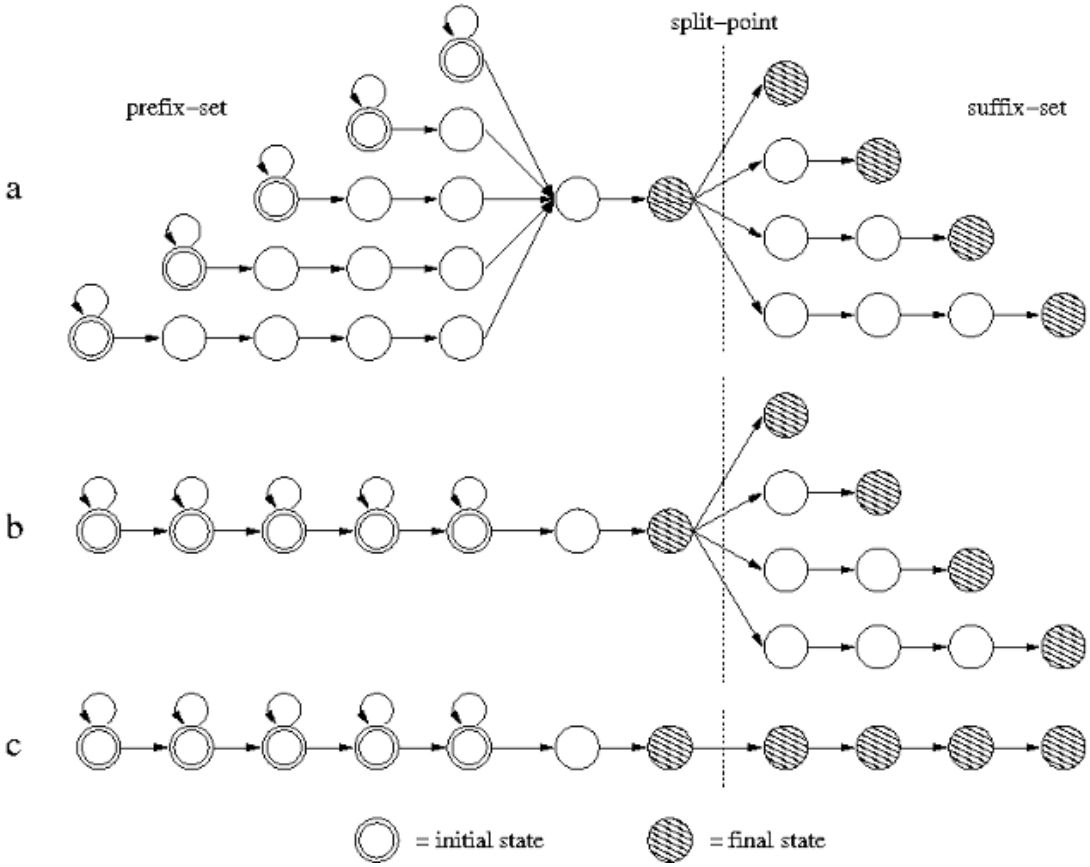

Figure 2.2 The three topologies of the HMM that has been used for the experiments taken from [43].

therefore, it does not need linguistic expertise.
The algorithm defines a distance function used to find similar words and grouping them into clusters. The clusters are considered as equivalence classes. Stems are then derived from classes centroids.

The development and improvement of stemming algorithms have been active since its beginning around 1960. Many different algorithms and approaches have been proposed that range from dictionary-based algorithms to rule-based algorithms to statistical algorithms and mixed methods (see Figure 2.3).
There is limited research on comparing the different stemming approaches. A study comparing all the presented algorithms have been done in [22], however, it is not clear how the algorithms have been evaluated, and on what basis, they have been compared. After presenting the algorithms, their advantages and limitations have been shown in the form of three tables focusing on the time consumption, the aggressiveness, the language dependency, and the error rate. It is reported that Porter stemmer produces the best output and have less error rate than the other stemmers.

In [45], the advantages and disadvantages of rule-based approaches and statistical approaches have been discussed, and the performance of Porter stemmer, YASS and GRAPH-BASED STEMMER (GRAS) [31] have been compared in term of strength [46] and computational time. In term of strengh, it is noted that YASS is very aggressive on all languages (e.g., Bengali, English, and French), and that GRAS performs equally well compared to rule-based stemmer, stating that YASS outperforms the other stemmers. Based on computational time, it is reported that YASS performs the worst, five times slower than GRAS its closest competitor.

In [42], four stemming approaches (Snowball stemmer [26], 4-grams and their proposed algorithm Pseudo-4 and Pseudo-5) have been evaluated and compared. It is stated that although the snowball algorithm outperformed the Single N-gram algorithm on seven of eight languages, the difference was only statistically significant in Dutch.

In [44], authors compared the retrieval performance of YASS algorithm to three other algorithms (Lovins, Porter and N-gram stemmers). It is stated that the performance of YASS is comparable to Porter Stemmer. The performance of four rule-based algorithms (Lovins, Porter 1, Porter 2 and Paice/Husk stemmers)

have been investigated in [47]. The study focus on four factors on which the algorithms have been compared, the Index Compression Factor, the Word Stemmed Factor (WSF), the Correctly Stemmed Words Factor (CSWF) and the Average Words Conflation Factor (AWCF) (i.e. higher the AWCF, higher the accuracy of the stemmer). It is indicated that Porter1 and Porter2 stemmers have less compression strength than the other two stemmers. It is also stated that under-stemming and over-stemming errors happened more often in Lovins and Porter1 stemmers. Furthermore, it is reported that Paice/Husk stemmer is comparatively better. Finally, it is stated that the results obtained by Porter2 stemmer are quite accurate; still, it produces over-stemming errors. The details of the four factors listed above are beyond the scope of this thesis. The interested readers might refer to [47].

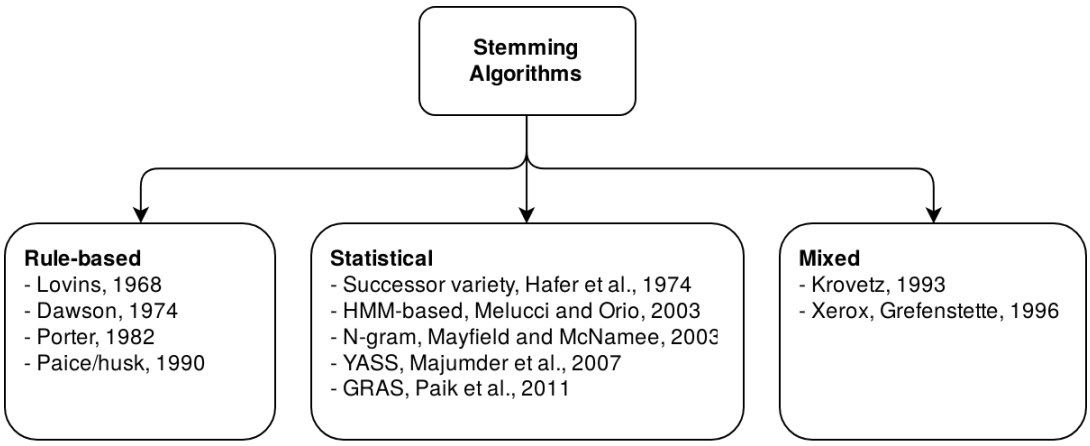

Figure 2.3 A number of stemming algorithms.

Lemmatization

Similar to stemming, lemmatization aims to map inflectional forms and derivationally related forms to a common base form (i.e. the words 'write', 'written', 'wrote', 'writing' may be grouped under the base form 'write'). However, unlike stemming in which a stemmed word could be unreal, lemmatization refers to a more appropriate and accurate approach.
Lemmatizers are typically language-dependent algorithms [48; 49; 50; 51; 52; 53;

54]. Most approaches are dictionary based. Words are confronted to a dictionary that maps the word variants to its base form.
In [51; 52; 55], rule-based approaches are used. For instance, an iterative algorithm is presented in [52] and applied to the Swedish language. The complete procedure of Hellberg lemmatizer is shown in Figure 2.4.

Automatic learning approaches of lemmatizers have also been attempted. In [56], two-level morphological rules learned from a list of word pairs using three possible elementary operations single character insertion, deletion and replacement were applied to English, Xhosa and Afrikaans languages. In [57], Jongejan et al. proposed a supervised approach to learn lemmatization rules for Dutch and German languages automatically. A process trains a set of rules in such as way that for each full form, an elected rule should produce the correct lemma. Figure 2.5 shows the training process of the set of rules.

Case Folding

Applications like information retrieval or speech recognition map all letters to lower case. In such applications, letter case is not essential. In other mining applications (TC, information extraction), the case could be helpful in avoiding some ambiguities (e.g., *US* the country and *us* the pronoun). Finally, in sentiment analysis, the case is significant. Capital letters generally show a strong attitude. In [12], using a maximum entropy model [58], Parikh et al. find that the attribute holding the number of capital letters in a word have contributed the most to the classification model. The impact of case folding alongside other preprocessing steps on TC has been investigated in [11] for two news datasets and two emails datasets in two different languages (English and Turkish). It was concluded that lowercase transformation improves classification performance regardless of the domain and language.

2.1.3 Stop-Words Removal

Stop-words are typically defined as common, non-informative, non-discriminative words. In [59], Hans Peter Luhn, a pioneer in information retrieval and the first to discuss the notion, described stop-words as insignificant words, the opposite of

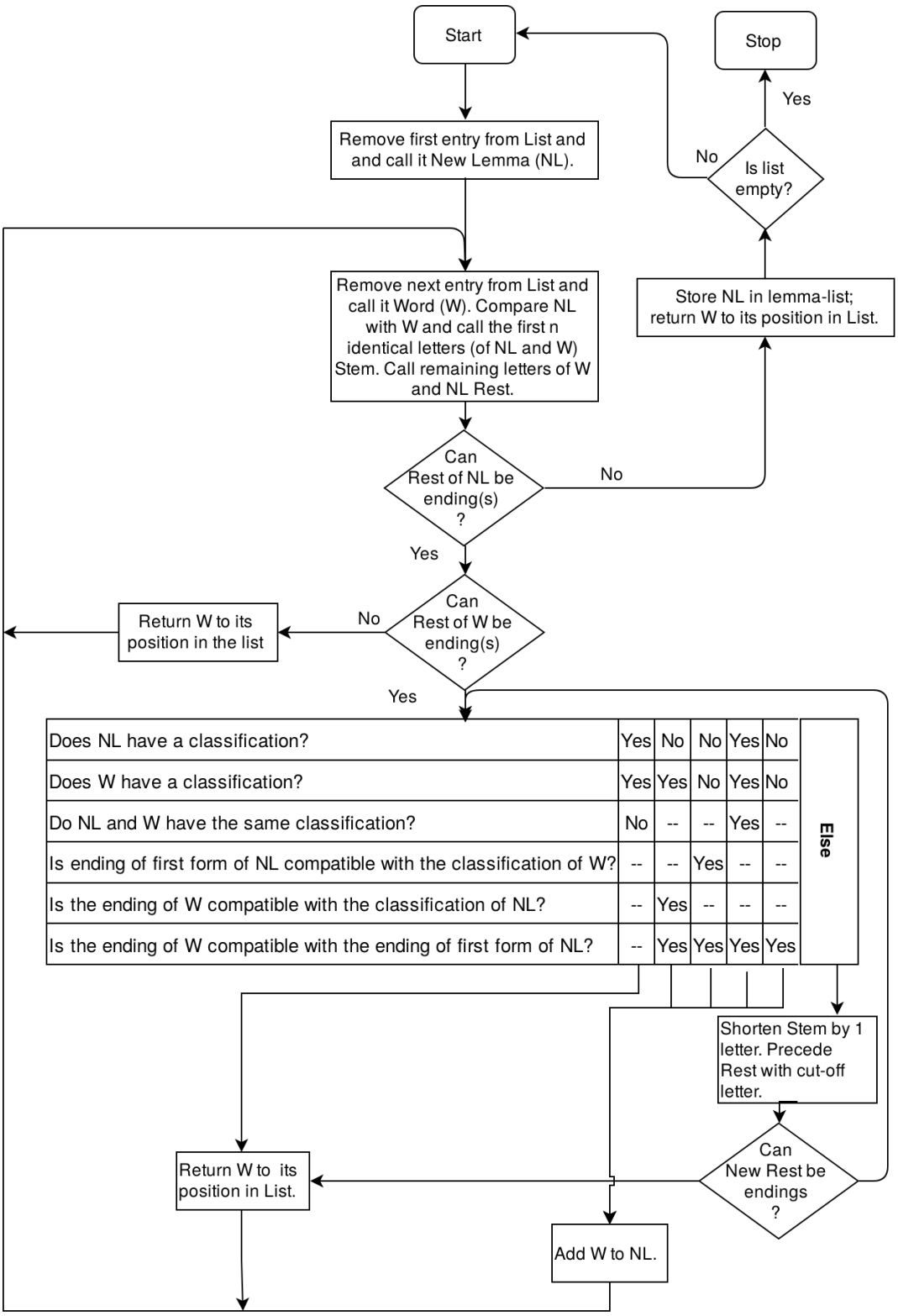

Figure 2.4 Flow chart of the Hellberg lemmatizer

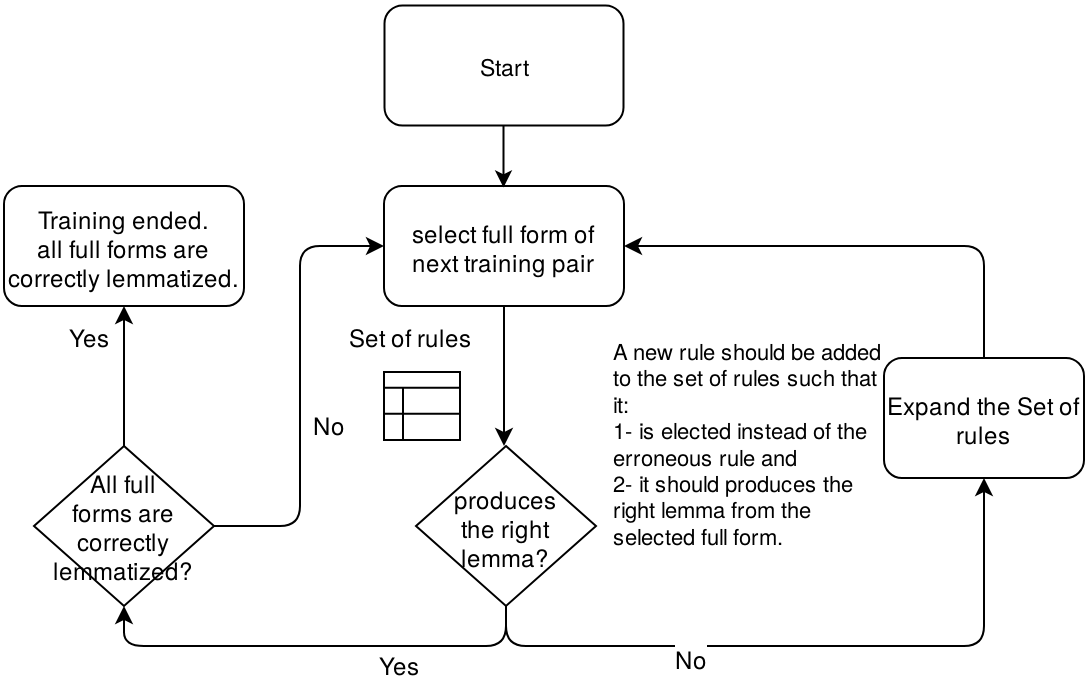

Figure 2.5 The process of building a rule set from training pairs as described in [57]

keywords.

The stop-word removal is a commonly used preprocessing step to reduce the indexes space and to improve the performance of the classification task [7; 60; 61; 62; 63]. For instance, words like “the” or “is” are common words and do not have discriminative abilities, they can not be helpful for classification. Similarly, words that occur in every document, or words that are equally distributed over all classes do not have discriminative abilities. Furthermore, very rare words do not have enough information to be contributed.

Stop-words removal has been approached by several ways. The first and the most used approach addresses removal by rejecting words appearing in a language-dependent compiled list of stop-words such as Van’s stop-list [64], Brown’s stop-list [65] (see Figure 2.6).

Several automatic methods have been proposed and explicitly applied to text classification [7; 61; 62; 63; 64; 65; 66]. Aside from the standard approach and approaches combining multiple stop-lists, the automatic methods could be classified into four types:

- Methods based on Zipf's law,
- Term-Based Random Sampling (TBRS),
- Term Weighting Scheme (TWS).

Methods based on Zipf's Law

Zipf's law [67] maintains that in human behavior, events frequently respect a particular distribution such that the number of occurrences of an event is inversely proportional to its rank. It is shown empirically that given a large sample of words, the law also applies to the frequency of words. Stop-words removal methods inspired by Zipf's law include removing the most frequent words (high Term Frequency (TF)); removing words that occur once which significantly reduces the dimensionality of feature space; removing words with low inverse document frequency (see Section 2.4); removing words with low TF [7; 66].

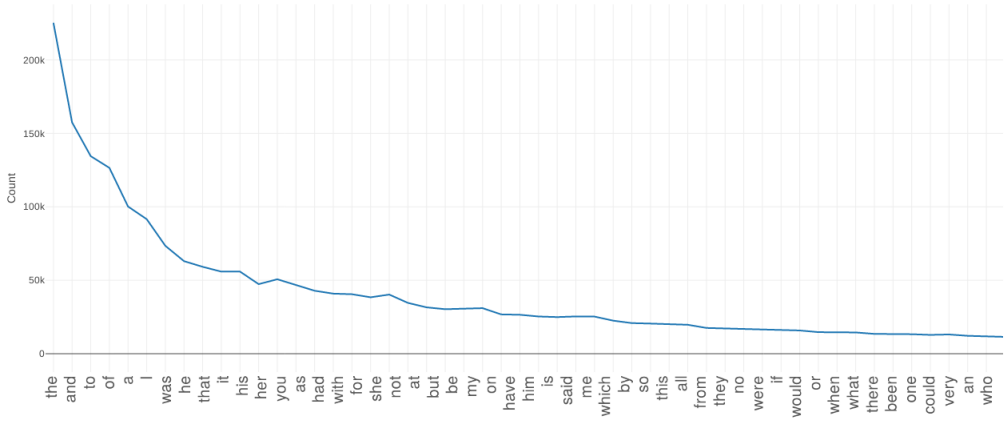

Figure 2.6 Zipf's law: most common word in english

Term-Based Random Sampling (TBRS)

TBRS is first introduced by Lo et al. in [66] to derive the stop-word list in information retrieval. TBRS is inspired by query expansion in [68]. The method selects documents that contain the query term rather than selecting similar words to expand the query.

TBRS iterates over random samples where each document in a sample contains a randomly selected term. Terms are then ranked using Kullback-Leibler divergence measure [69] defined as follow:

$$w(t) = P_x.log_2\frac{P_x}{P_c}$$

In the above formula, P_x is the normalized TF of the term t within the sample, and P_c is the normalized TF of the term t in the whole collection.
Even though TBRS have first been proposed for information retrieval, the method has later been applied and compared with other approaches in the context of TC [7; 70].

Term-Weighting Schemes (TWS)

Term-weighting approach is one of the most used stop-word removal method, second only to the standard approach in which terms are confronted with a compiled list.
The approach uses a TWS to assign a score to each term that depicts its importance. The stop-word list is then filled with terms having a score lower than a chosen threshold. Finally, terms occurring in the created stop-word list are removed from the vocabulary.

Comparison of Stop-Word Removal Methods

Standard stop-list removal and Mutual information stop-list removal are commonly applied for TC without careful consideration of its impact on the task performance. Few works, however, have been done on comparing the different existing approaches. Aside from researches proposing new methods, we were not surprised to find only one study comparing these approaches since automatic stop-word removal task could be thought of as a FS routine which is extensively studied (Report to Section 2.3 for more extensive comparison).
The effect of different stop-word removal approaches have been investigated in sentiment analysis using six twitter datasets in [7]. It is stated that concerning feature space reduction, removing terms occurring once reduces the feature space

extensively, Mutual Information comes second, and removing high-frequency words have no real impact. Furthermore, it is stated that the removal of IDF stop-list has an adverse impact on classification performance, on the other hand, the mutual information stop-list reports the best classification performance. Finally, it is concluded that the removal of terms occurring once has the best trade-off impact on classification performance, feature space reduction and data sparsity.
The effectiveness of four Zipf's law-based baseline methods alongside the proposed TBRS has been investigated and compared for information retrieval in [66]. It is reported that the proposed method shows comparable performance to the baseline methods while being more computationally effective. Moreover, it is concluded that the combination of standard stop-list and any stop-list produced by either of the examined methods improves the retrieval performance.

2.1.4 Part-of-Speech (POS) Tagging

The POS tagging is the process of labeling words by its corresponding lexical categories (also known as tags and word classes) such as nouns, verb, adjective, etc. An example of POS-tagging is shown in Figure 2.7. Initially, the POS list (also called tagset) included only eight tags: noun, pronoun, verb, adverb, preposition, conjunction, particle, and article. However, bigger tagsets have been proposed. In [71], the Penn Treebank POS tagset including 45 word classes is presented (see Table 2.1), another 87-tag tagset proposed for the Brown corpus [72]. An even bigger list found in [73], the C7 tagset includes 146 categories.

The importance of POS tagging comes from the amount of information that could be extracted about a word and its neighbors. For instance, the distinction between heteronyms (words that spell identical but have a distinct meanings or different pronunciations) based on its POS tag improves the speech recognition system, and it can make a more natural speech synthesis system. The adjective *invalid* [ɪnˈvæl.ɪd] pronounced differently from the noun *invalid* [ˈɪn.və.lɪd]. Also, knowing whether a pronoun is possessive or personal gives us information about neighboring words (personal pronouns are likely to be followed by a verb, whereas possessive pronouns are followed by a noun). In addition to speech recognition, POS tagging is significant for information retrieval, text parsing, NER by identifying

nouns, stemming as different POS takes different morphological affixes. POS tagging has also been applied to TC. In [74], the authors used POS tagging to reduce the misinterpretation of similar words. The use of different POS have been explored for TC [75]. In this paper, Chua states that nouns are the best type to describe a category's content, reasoning that nouns names entities such as a concept and an idea. The author also asserts that the list of nouns as features is more efficient than a list created using χ^2 or IG.

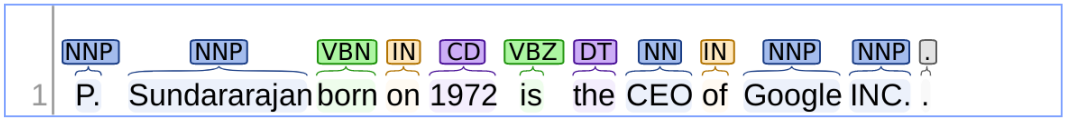

Figure 2.7 POS-tagging using the Online Stanford CoreNLP

2.1.5 Named Entity Recognition (NER)

NER aims to automatically identify and classify named entities in texts into pre-defined categories such as person, location, date (See Figure 2.8).

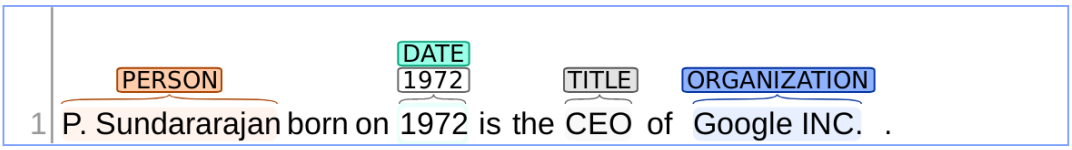

Figure 2.8 NER using Stanford CoreNLP

NER is useful in various applications such as summarization, information extraction, TC, natural language understanding, question answering, content recommendation, and more.
NER is addressed by two main approaches: a rule-based approach mostly adapted for detection of temporal and numerical expressions (e.g., days of the week, months, years, money), and a machine learning based approach.

An example of a rule-based system used for tagging temporal expressions in texts could be found in *NLTK's temporal expression tagger*. The below code snippet in Listing 1 is taken from the beginning of the file.

Table 2.1 The Penn Treebank POS tagset without punctuation marks [71]

Tag	Description	Example
CC	Coordinating conjunction	and, or
CD	Cardinal number	one, two, 15
DT	Determiner	the, them, these
EX	Existential there	there
FW	Foreign word	voilà, mais
IN	Conjunction, subordinating or preposition	among, in, into
JJ	Adjective	clean, nice
JJR	Adjective, comparative	cleaner, nicer
JJS	Adjective, superlative	cleanest, nicest
LS	List item marker	
MD	Modal	may, could, might
NN	Noun, singular or mass	computer, machine, air
NNS	Noun, plural	computers, machines
NNP	Proper noun, singular	France, Alice
NNPS	Proper noun, plural	three Alices
PDT	Predeterminer	both, all
POS	Possessive ending	’s
PRP	Personal pronoun	him, we
PRP$	Possessive pronoun,	her, our
RB	Adverb	quickly, clearly
RBR	Adverb, comparative	greater
RBS	Adverb, superlative	hardest, best
RP	Particle	up, across
SYM	Symbol (mathematical or scientific)	$\times, =$
TO	to	to run
UH	interjection	oops, googbye
VB	verb, base form	eat, hit, run
VBD	verb, past tense	ate, hit, ran
VBG	verb, gerund or present participle	hitting
VBN	verb, past participle	assigned
VBP	verb, non-3rd person singular present	think
VBZ	verb, 3rd person singular present	thinks
WDT	wh-determiner	that, which, whatever
WP	wh-pronoun, personal	that, which, whom
WP$	wh-pronoun, possessive	whose
WRB	wh-adverb	how, why

Listing 1 Snippets from NLTK's temporal expression tagger: timex.py

```
# Predefined strings.
numbers = "(^a(?=\s)|one|two|three|four|five|six|seven|eight|\
          nine|ten|eleven|twelve|thirteen|fourteen|fifteen|\
          sixteen|seventeen|eighteen|nineteen|twenty|thirty|\
          forty|fifty|sixty|seventy|eighty|ninety|hundred|thousand)"
day = "(monday|tuesday|wednesday|thursday|friday|saturday|sunday)"
...
dmy = "(year|day|week|month)"
rel_day = "(today|yesterday|tomorrow|tonight|tonite)"
exp1 = "(before|after|earlier|later|ago)"
exp2 = "(this|next|last)"
iso = "\d+[/-]\d+[/-]\d+ \d+:\d+:\d+\.\d+"
year = "((?<=\s)\d{4}|^\d{4})"
regxp1 = "((\d+|(" + numbers + "[-\s]?)+) " + dmy + "s? " + exp1 + ")"
regxp2 = "(" + exp2 + " (" + dmy + "|" + week_day + "|" + month + "))"
...

def tag(text):
    ...
    ...
    # Captures expressions such as 'number of days' ago, etc.
    found = reg1.findall(text)
    found = [a[0] for a in found if len(a) > 1]
    for timex in found:
        timex_found.append(timex)

    # Variations of this thursday, next year, etc
    found = reg2.findall(text)
    found = [a[0] for a in found if len(a) > 1]
    for timex in found:
      timex_found.append(timex)

    ...
```

Machine learning based NER uses generally supervised learning algorithm (e.g. Conditional Random Fields [76; 77], Support Vector Machine (SVM) [78; 79], Hidden Markov Model [80], etc.) to detect entities, however semi-supervised learning algorithms (bootstrapping) and unsupervised learning algorithms (clustering) could also be found [81; 82].

In [83], a FS method that considers both singular terms and named entities as features have been proposed. Confidence Weight scheme (ConfWeight) is used to apply scores to general terms. An improved version of ConfWeight is proposed and used to evaluate the named entities. The proposed method is evaluated and compared to Chi-squared (χ^2) and Information Gain (*IG*) methods. It is noted that considering the named entity information in FS methods improves the classification performance.

2.2 Text Representation

A text document is one or more pieces of text of a written natural language. A natural language is a human language such as Arabic, French or Chinese that has evolved naturally as a means of communication among people [84].

Text documents in natural languages are very complex to be processed directly in their raw format, due to the lack of structured data and the significant amount of useless information they contain.

Text Representation is about representing text suitably to be handled by machine learning algorithms. Many different ways exist to represent text. First, we can always represent text as a string of characters. This simple representation is the most general way to represent text as it can be used to represent any textual data in any natural language. However, this representation is often useless when it comes to semantic analysis which is needed in most machine learning tasks. In order to be able to do semantic analysis, the representation should at least recognize words. A word level representation is the first level of semantic richness as words are the basic unit of any human communication natural language. This level of representation is also the most common way to represent text documents. Word level representation presents many promises. For example, by recognizing words,

we can easily identify the most frequent words in a collection of documents and many useful statistics such as the number of documents that contain a specific word. Moreover, we can use words to form topics. For example, some words are positive, and others are negative. So word level representation can be used for sentiment analysis. However, it is not always easy to identify words in text documents. For instance, in the Chinese language, text is seen as a sequence of characters without whitespace delimitation. In such languages, particular techniques are used to segment words. These techniques may introduce errors. Instead of words, we can also easily represent text as a sequence of terms usually by normalizing words using stemming [24] and case folding. Word level representation present many promises. However, it is not always easy to identify words in text documents. For instance, in the Chinese language, text is seen as a sequence of characters without whitespace delimitation. In such languages, particular techniques are used to segment words. These techniques may introduce errors. Part of Speech (POS) Tagging is the next level in semantic richness. By adding POS tags as additional information for the word level representation, new interesting statistics could be counted, such as the number of nouns, and verbs; but also, it opens new possibilities such as extracting relations between nouns and verbs, etc. Generally, POS tagging is used for NER tasks. The next level of representation is done by adding syntactic structures information which is useful for example in detecting writing styles and correcting grammar mistakes. An example of syntactic structures obtained using Standford CoreNLP is shown in the Figure 2.9. We may also go futher to recognize named entities and relations between them (See Subsection 2.1.5).

Besides the choice of the level of text representation, there are many models to represent a document of textual data. The models differ on the assumption they make about words and documents. Term-independence and document-independence are the two primary assumptions.

Term independence asserts that, given the relevance of a term, we cannot make any statement as to the relevance of the other terms. The term independence assumption implies that terms of the documents in a collection are not related. Document-independence asserts that the relevance of a document does not affect the relevance of the other. Different models may accept either both assumptions, only one of them or refuse both. A popular model for representing text document

that accepts both assumptions is Vector Space Model (VSM). The algebraic model represents documents as a vector of index terms where each term is associated with a weight. The weight tells how much information a term contributes to the semantics of a document. VSM is quite popular in TC [6; 85; 86; 87; 88] and it exhibits good performance, however, due to the term independence assumption, it is clearly not adapted for spell checking as an example. In order to lessen the impact of the term independence assumption, some authors uses n-grams, i.e., the sequence of n words as features of the vector space [13; 14; 15; 89; 90].

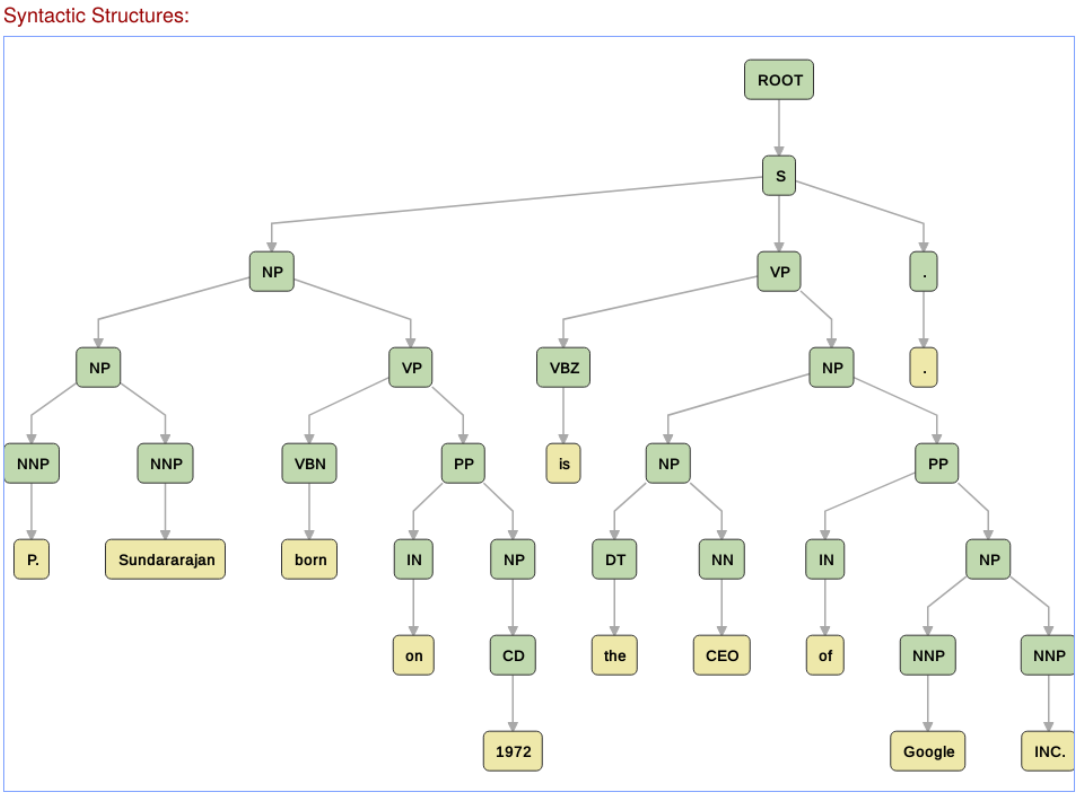

Figure 2.9 Syntactic Structures of the text appeared in Figure 2.8.

2.3 Feature Selection (FS)

In TC, the number of features generally ranges from ten thousands to hundred thousands and more, many of which are highly correlated variables, i.e., redundant information that yield the same information about the classes or irrelevant variables

that hold noisy information making prediction harder [91]. FS (variable selection), i.e., the process of selecting the minimum number of relevant features to be used in building a model, plays an vital role in TC and generally, in data mining tasks. FS presents a number of benefits. By reducing the number of features, it helps in making data simpler and easier to understand, reducing the computational complexity of the model construction, lessening the impact of the curse of dimensionality i.e. reducing the size of training dataset required, and improving the model performance (lower overfitting) [92].
A FS algorithm is typically a search algorithm which includes an evaluation method which goal is to find a subset of features that optimize a certain performance measure.
FS algorithms has been classified into three types of algorithms, based on the evaluation method:

- filter methods,
- wrapper methods,
- embedded methods.

2.3.1 Filter Methods

In general, Filter-based FS methods refer to the process of applying a statistical metric that evaluates the goodness of features. Filter-based FS algorithms are general methods, independent of the machine learning model.
In general, a score is assigned to each feature to depict the usefulness of the feature. Features are then ranked according to their score. Finally, a number of features are selected filtering out the rest of features (see Figure 2.10). The number of features (may also be a percentage or even a cut-off point) to be selected is a parameter generally obtained by cross-validation.
Filter-based FS methods require less computational time and are generally more robust to overfitting. However, due to the independence assumption of variables, redundant features are casually not filtered.
A considerable number of measures have been used in feature scoring including:

- Pearson Correlation [93; 94].

- Mutual Information proposed by Shannon in his paper "A mathematical theory of communication" in 1948, re-published in [95] used for FS in [5; 6; 96; 97; 98; 99; 100; 101].
- χ^2 proposed by Pearson in [102] re-published in [103] used for FS in [5; 6; 97]
- Spearman Correlation proposed in [104] and used in [105]
- Kendall Correlation proposed in [106] and used in [107; 108]
- Fisher Score [109]
- Welch's t-test proposed in [110] and used in [111] to detect the most important eigenbrains (i.e., eigenvectors[112] obtained by principal component analysis) for diagnosis of Alzheimer's disease.
- Infinite FS was introduced and used for FS in [113] applied to object classification.
- Feature Similarity [114].
- Traditional TWS (binary, TF, Term Frequency-Inverse Document Frequency (TF-IDF)) discussed in [8] and tested in [115; 116; 117].
- Supervised Term-Weighting (STW) methods (Term Frequency-Relevance Frequency (TF-RF) [9], Term Frequency-Inverse Category Frequency (TF-ICF) [86], Term Frequency-Odds Ratio (TF-OR) [118], Term Frequency-Gain Ratio (TF-GR) [5])

However, besides TF-IDF which is very popular, Term Frequency-Chi Squared (TF-χ^2) and Term Frequency-Information Gain (TF-IG) are the most commonly used feature scoring methods for TC tasks [10; 91].

For further readings, please refer to Chapter 3.

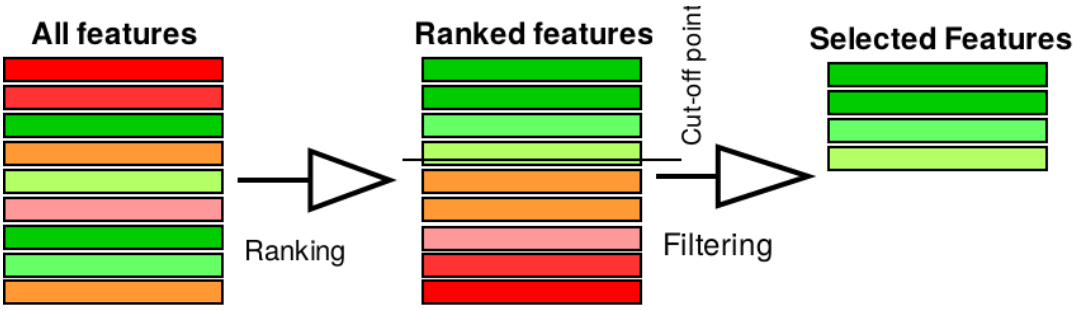

Figure 2.10 Filter-based FS

2.3.2 Wrapper Methods

Wrapper-based FS methods, in contrast to filter-based methods, use the induction algorithm to evaluate the subsets of features. The algorithm generates a subset of features; then it uses the subset to train a learning model. The error rate of the model is then computed and assigned to the subset. The best subset is the one with the lowest error rate (see Figure 2.11). Wrapper methods generally use optimization techniques to search the feature space with the fitness function as a learning algorithm. These techniques include:

- Genetic algorithm [119; 120; 121; 122; 123; 124; 125; 126; 127].
- Ants Colony [128; 129; 130].
- HillClimbing [131; 132; 133].
- Particle Swarm Optimization [134; 135; 136; 137].
- etc.

Although the wrapper methods tend to retrieve the best performing subset of features for a specific model, however, due to the use of the learning model as the fitness function, two issues arise, first these methods are typically very computationally complex, and second, they hold a higher risk of overfitting when the training data is insufficient.

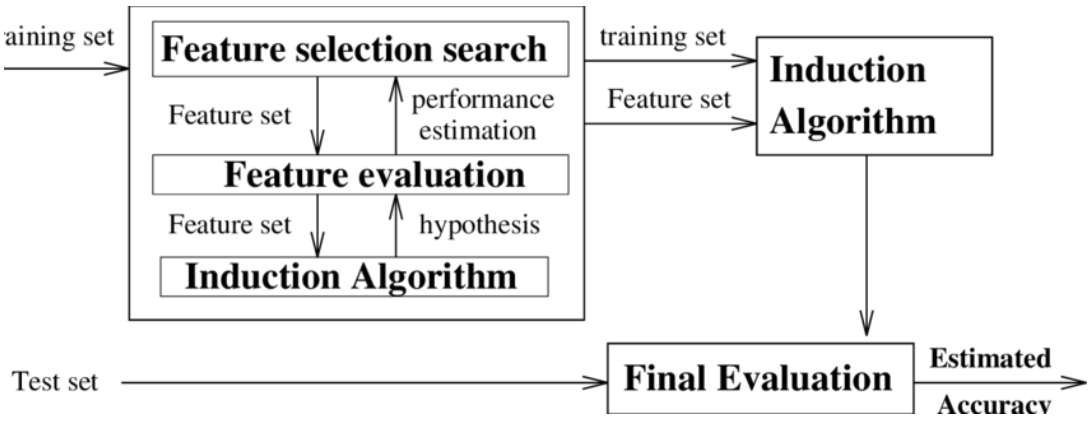

Figure 2.11 Wrapper-based FS [132]

2.3.3 Embedded Methods

Embedded methods try to overcome the shortcomings of wrapper methods. The name Embedded comes from the fact that the method is embedded in the learning algorithm, i.e., it is an inseparable part of the model training process. For instance, Decision Tree (DT) have their embedded FS methods. A DT algorithm involves a greedy search that finds the feature that best splits the instance space into two or more sub-spaces according to some metrics on the input features, at each iteration. At the end of the process, only the selected features are part of the DT model. This set of selected features is optimized for the DT and it is very probable that it won't work for the training of different models. Another example of embedded methods could be found in SVM. The SVM model generally uses a small set of observations (support vectors) in the training process. Hence, only terms belonging to these documents are considered.

2.4 Term-Weighting

Besides the direct utility of assigning weights to features, TWSs are critical for feature extraction, as well for filter-based FS methods. Additionally, TWSs can be used for automatic stop-word detection as shown in the Subsection 2.1.3.
In Chapter 3, we present in detail the TWSs applied to TC.

2.5 Classification Methods

In machine learning, TC is a supervised learning task since the true labels of the training data are fed to the learning algorithms. The algorithm tries to approximate a function $g : X \rightarrow Y$ that maps each document instance of the input space $x \in X$ to one of the pre-defined categories $y \in Y$ based on a training set of tuples (x, y). There is a wide range of approaches available for classifier learning such as SVM, DT, random forest, k-NN, etc.

2.5.1 Probabilistic Classifiers

Probabilistic classifiers are a set of algorithms that apply the Bayes theorem:

$$P(y|x) = \frac{P(y)P(x|y)}{P(x)} \tag{2.1}$$

$P(x|y)$ is then estimated by considering the naive independence assumption stating that features are not related to each other.

$$P(x|y) = \prod_i P(x_i|y) \tag{2.2}$$

Finally, applying the Bayes theorem with the independence assumption leads to the following classification rule:

$$\hat{y} = \arg\max_y P(y) \prod_i P(x_i|y) \tag{2.3}$$

2.5.2 Decision Tree (DT) Classifiers

DT learning is a supervised learning method that uses a tree as its predictive model (Figure 2.12). It can be used for both classification and regression. DT algorithms include Iterative Dichotomiser 3 (ID3) [138], C4.5 [138; 139], Classification and Regression Trees (CART) [140], etc.

The classification tree is a collection of nodes with all edges pointing away from the root node. Nodes can be either internal or leaf. An internal node represents

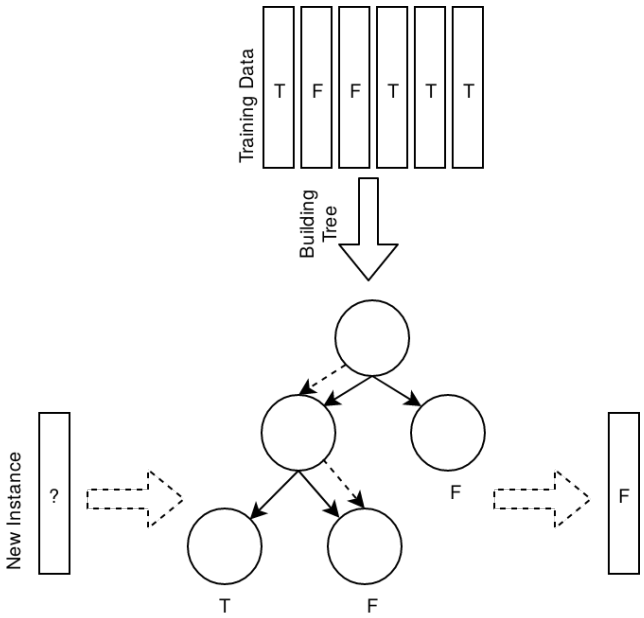

Figure 2.12 Decision Tree (DT)

an inducing rule (i.e., a test on a feature) that splits the instance space into two or more sub-spaces. A leaf node represents a decision that is, a terminal node representing the target value (also called the outcome).

The algorithm to construct the classification tree involves a greedy search that finds at each iteration the feature that best splits the instance space into two or more sub-spaces according to some metrics (e.g., *IG*) on the input features. These metrics generally measure the impurity (diversity) of the target variables within the subsets. The goal is then to find the split that minimizes the impurity. Metrics include *IG* [138], gini impurity, cross-entropy.

IG is an entropy-based metric from information theory. It measures the amount of information one random variable contributes about another random variable i.e., the reduction in entropy of one variable given information about another. Given a dataset D with n target values (i.e, classes), the entropy of D is defined as follows:

$$H(D) = -\sum_{i=1}^{n} p_i \log_2(p_i) \tag{2.4}$$

Where p_i is the proportion of elements belonging to i^{th} class. In terms of entropy, $IG(D,V)$ of a variable V relative to the dataset D is:

$$IG(D,V) = H(D) - H(D|V) \tag{2.5}$$

$$= H(D) - \sum_{v \in values(V)} \frac{|D_v|}{|D|} H(D_v) \tag{2.6}$$

Here, D_v is a subset of D where V have the value v.

IG, in contrary to entropy and gini impurity, measures the purity. Thus, the feature (or variable) that maximizes the *IG* is used to split the instance space.

Gini impurity is a measure of how heterogeneous a dataset is. It is very similar to entropy (see Figure 2.13).

Given a dataset D with n classes, and let p_i be the proportion of elements belonging to i^{th} class, the gini impurity I of the dataset could be defined as follows:

$$I(D) = 1 - \sum_{i=1}^{n} p_i^2 \tag{2.7}$$

2.5.3 Support Vector Machines (SVM)

SVMs [141] are one of the most popular choices of supervised classification algorithms in machine learning applications. It was used for classification first in [85]. SVM classifiers try to group the members of different classes into different areas of the input vector space by using a hyperplane (see Figure 2.14). In the SVM model, instances are represented as points (real vector) in that space.

Given a binary classification task where the instance label is represented by $y_i = 1$ if $\vec{x}_i$ belongs to the positive class and $y_i = 0$ if it belongs to the negative class, and a training dataset of n instances represented as follows:

$$(\vec{x}_1, y_1), (\vec{x}_2, y_2), ..., (\vec{x}_n, y_n)$$

in the vector space, any hyperplane could be expressed as follows:

$$\vec{w}.\vec{x} - b = 0 \tag{2.8}$$

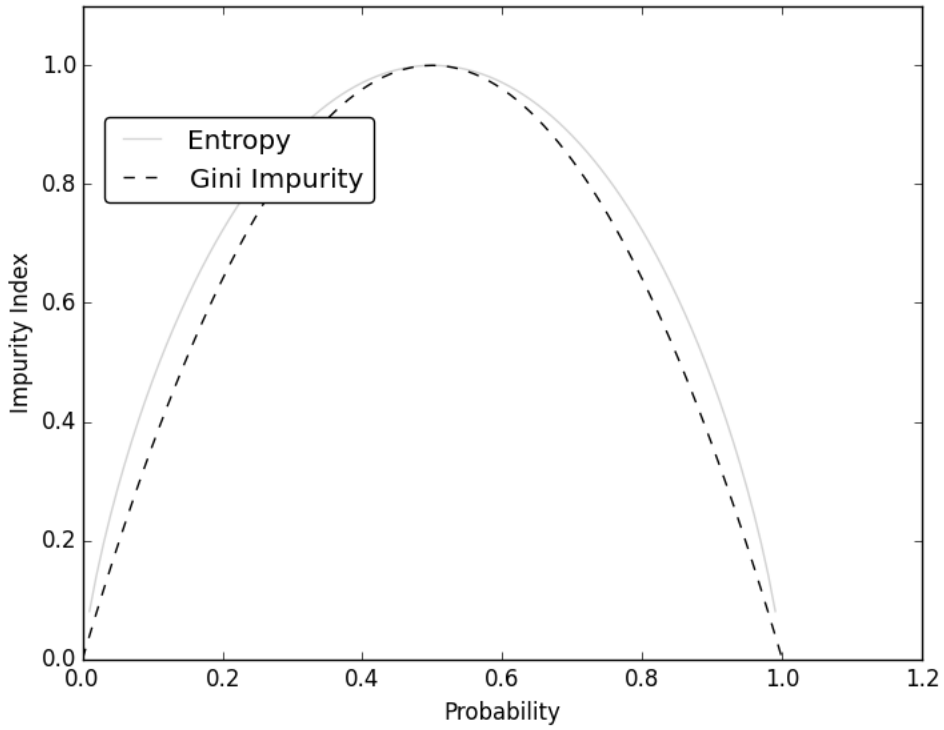

Figure 2.13 Impurity index: entropy and gini impurity.

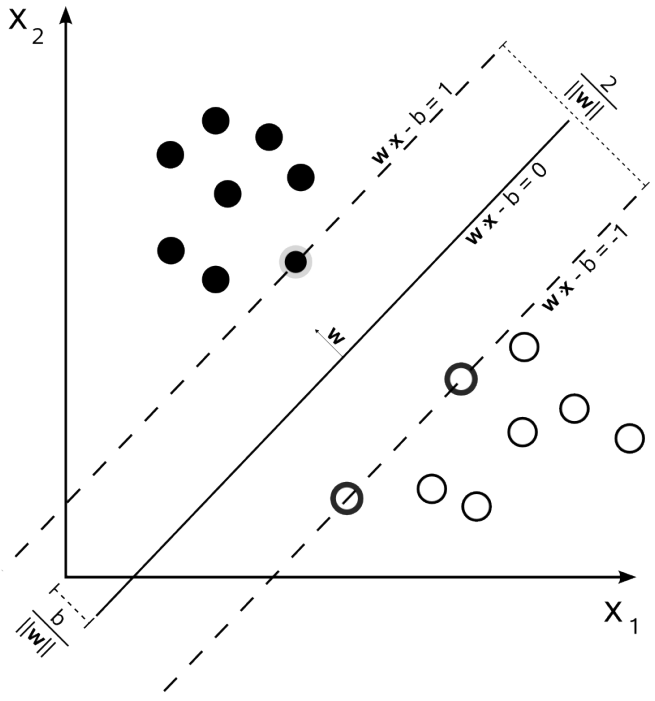

Figure 2.14 Maximum-margin hyperplane for an SVM trained for the binary classification task.

When the training dataset is linearly separable, SVM finds a maximum margin hyperplane that separates the positive points from the negative points by solving the following optimization problem:

$$\text{Minimize } ||\vec{w}|| \text{ subject to } y_i(\vec{w}.\vec{x_i} - b) >= 1, \text{ for } i \in [1,n] \tag{2.9}$$

Finally, the classification function is, therefore:

$$y = f(\vec{x}) = \text{sgn}(\vec{w}.\vec{x} - b) \tag{2.10}$$

Where $\vec{x}$ and b are the solutions to the above optimization problem. For non linearly separable training data, the hyperplane that leads to the fewer incorrect classification is selected. It is done by introducing a hinge loss function l [142]:

$$l(y) = max(0, 1 - y_i.y) \tag{2.11}$$

It can be seen from the above formula that the hinge loss is equal to 0 when the right class is predicted and $|y| >= 1$. However, a penalty is applied when $|y|$ is smaller than 1 or when the prediction is erroneous (y and y_i have opposite sign). This penalty increases linearly with y. The optimization problem to minimize becomes:

$$\left[\frac{1}{n}\sum_{i=1}^{n}\max(0, y_i(\vec{w}.\vec{x}_i - b))\right] + \lambda||\vec{w}||^2 \tag{2.12}$$

2.5.4 Bagging and Boosting

Bagging (also called Bootstrap Aggregation) [143] and Boosting belongs to machine learning ensemble meta-algorithms in which several learning models are combined. They were proposed in order to control the bias-variance tradeoff [144]. In one hand, boosting combines a set of weak learners (high bias) in order to obtain one strong learner (low bias) [145]. On the other hand, bagging reduces the variance by combining multiple strong learning models.

In bagging, given a training dataset D, n subsets of D are generated by random sampling with replacement. The n learning models are then trained on the above subsets and results are finally averaged. Figure 2.15 presents a schematic showing the process of bagging. A popular improvement to the bagging algorithm is the random forests [146; 147]. In random forests, features that can be evaluated at each split point are limited to a random subset. This change makes predictions from the DT less correlated, and hence, it reduces the variance.

Boosting as mentioned earlier combines several weak learning models creating one strong model. One significant difference from bagging is that models are dependent. Models are learned in sequence and training a model requires information from the earlier trained model. In boosting, misclassified samples, at the last iteration, are given more weights, so they are picked more often in the next round. They are considered more complex and hence requires more training iterations. Figure 2.16 present a diagram showing the boosting process.

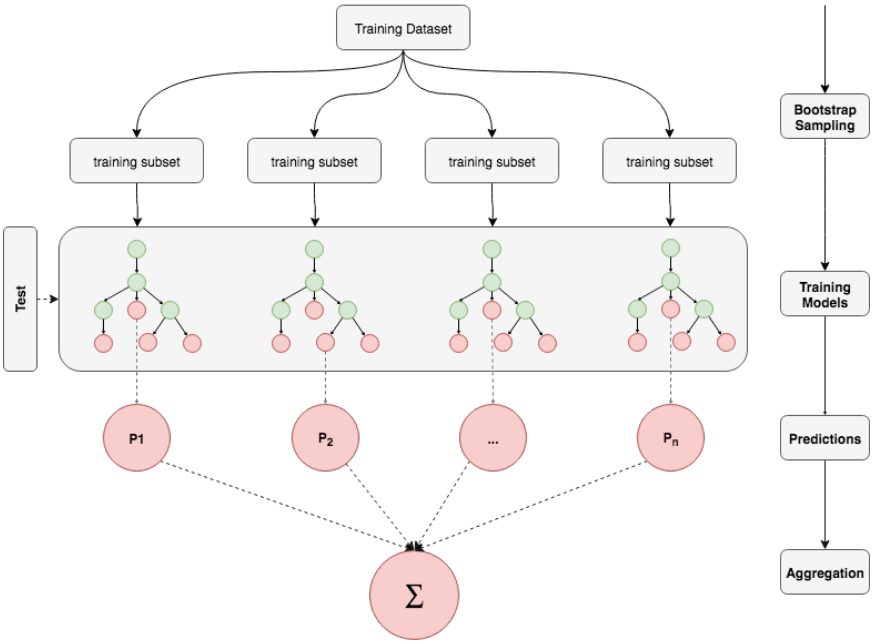

Figure 2.15 Diagram showing the process of the bagging algorithm

2.5.5 Passive-Aggressive (PA)

PA proposed by Crammer et al. in [148] is a learning algorithm focused on online learning and large-scale dataset. The method treats a flow of documents and outputs a prediction once a document is received. Later, whenever a document true-label is discovered, the method redefines its prediction function. Given a dataset D of n points: $(\vec{x}_1, y_1), (\vec{x}_2, y_2), ..., (\vec{x}_n, y_n)$, the classification function is simply the same as in SVM:

$$f(\vec{x}) = \text{sgn}(\vec{w}.\vec{x} - b)$$

The algorithm is also based on the hinge loss defined by:

$$l(y) = max(0, 1 - y_i.y))$$

Aside from that, the algorithm works in rounds. In the beginning, it initializes the weight vector $\vec{w}_1$ to $(0, .., 0)$. Then, at each round i, an update rule is applied that

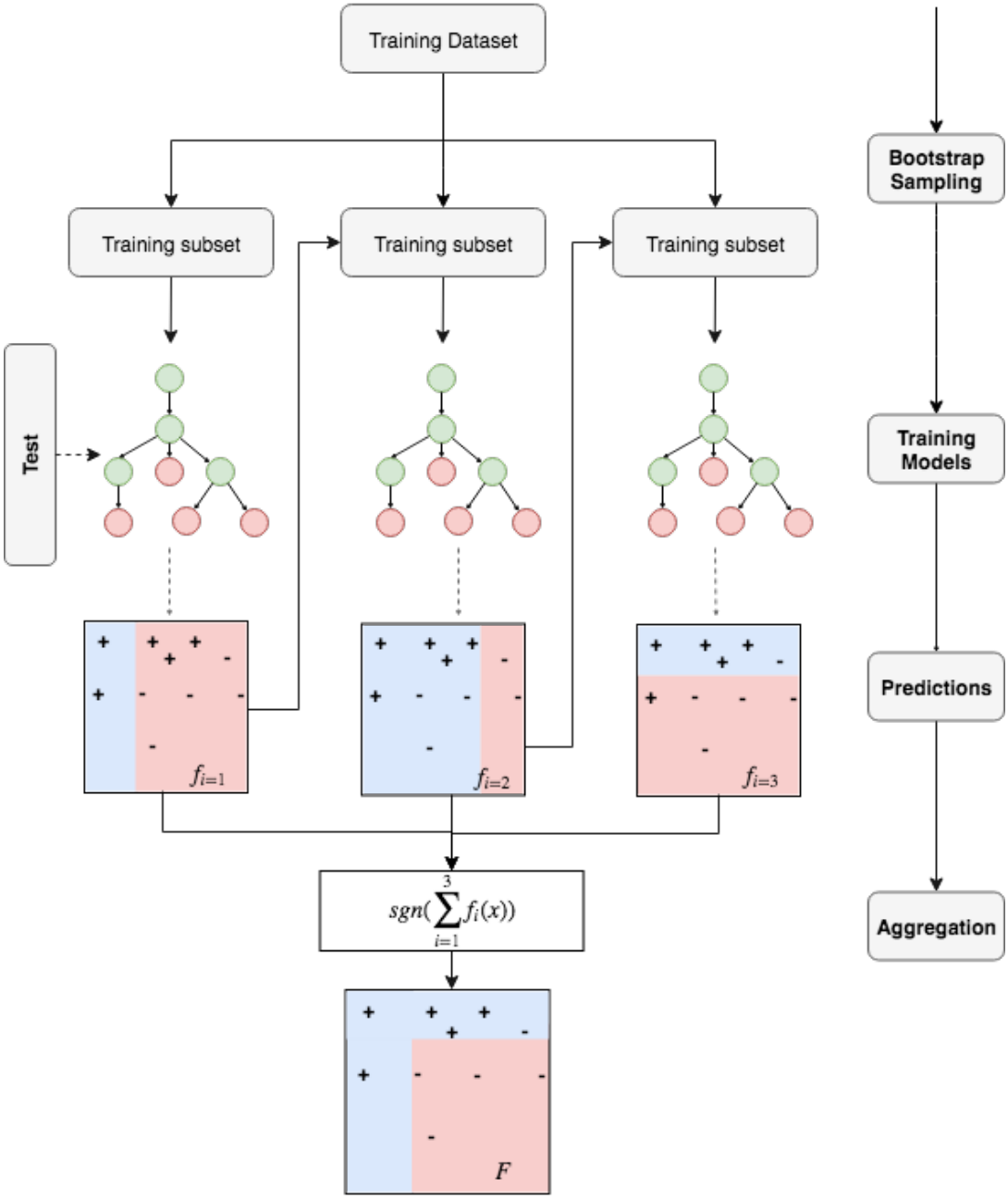

Figure 2.16 Diagram showing the boosting process

sets the weight vector to be the solution to the following optimization problem:

$$\vec{w}_{i+1} = \arg\min_{\vec{w}} \frac{1}{2}||\vec{w} - \vec{w}_i||^2 \text{ subject to } l(\vec{w}.\vec{x}_i, y_i) = 0.$$

In one hand, when the hinge loss is zero, the algorithm is passive and no modification is done, $w_{t+1} = w_t$. On the other hand, whenever the loss is positive (incorrect prediction), the algorithm aggressively updates the new weight vector $\vec{w}_{i+1}$ so that it satisfies the constraint $l = 0$.

The aggressive behavior of the algorithm may lead to undesirable results. In order to satisfies the constraint, one single noise instance may cause many prediction errors. In order to overcome this issue, [148] introduces a new coefficient (slack variable) $\xi > 0$ to soften the constraint of maximizing the margin. The technique was already used by [141] to derive soft margin classifiers. The resulting optimization problem is:

$$\vec{w}_{i+1} = \arg\min_{\vec{w}} \frac{1}{2}||\vec{w} - \vec{w}_i||^2 + C\xi^2 \text{ subject to } l(\vec{w}.\vec{x}_i, y_i) \leqslant \xi. \tag{2.13}$$

Here, C is the aggressiveness parameter which controls the influence of the slack term on the objective function.

2.5.6 Stochastic Gradient Descent (SGD) Classifiers

SGD classifier [149] is a discriminative learning approach for training linear models such as linear SVM and logistic regression. The algorithm is a simplification of the gradient descent algorithm. It focuses on the online learning and efficient large-scale dataset. Given a training set of n examples, as in the example of SVM, and also a loss function l to measure the gap between the predicted and the observed variable. SGD try to learn a linear function $f(\vec{x}) = \vec{w}.\vec{x} - b$. Learning the linear function amounts to minimizing the empirical risk (also called training error):

$$E(\vec{w}, b) = \frac{1}{n}\sum_{i=1}^{n} l(y_i, f(\vec{x}_i)) + \alpha R(\vec{w}) \tag{2.14}$$

Here, l is the loss function, R is the regularization term and α is a positive parameter. In SGD, the training error is minimized and the parameters are updated by iteratively computing the gradient of the loss function on a single example:

$$\vec{w}_{i+1} = \vec{w}_i - \eta \nabla [l(y_i, f(\vec{x}_i)) + \alpha R(\vec{w})] \tag{2.15}$$

Where η stands for the learning rate.

2.5.7 Nearest Centroid (NC)

NC classifier is a simple neighborhood-based classification model that labels test samples by the class of the closest group (distance to the centroid/mean) of training samples.

Given a labeled training set D of m-dimensional input vectors $D = \{\vec{x}_i, y_i | i = 1, ..., n\}$ where $x_i \in \Re^m$ denotes the i^{th} training point and $y_i \in \Re$ denotes the class labels, the algorithm first computes the centroid for each class:

$$\vec{m}_l = \frac{1}{|S_l|} \sum_{i \in S_l} \vec{x}_i. \tag{2.16}$$

Where S_l represents the indices of points having label l. By computing the per-class centroids, the training phase is complete. To classify an unlabeled point, the algorithm assigns to an unlabeled point $\vec{x}$ the output label y which is the label of the NC:

$$y = \arg\min_l ||\vec{m}_l - \vec{x}|| \tag{2.17}$$

2.6 Evaluation

After training a classification model, an evaluation step to check how well the model will perform on unseen data. Therefore, it is important not to train the model on the entire dataset. Generally, an evaluation procedure is performed that splits data into splits for training, testing, and validation. In the training phase, a model is built. The model parameters could then be optimized in the validation phase. And finally, the model is evaluated in the testing phase.

2.6.1 Evaluation Procedures

A key element in the model evaluation is to always train and test on different data. Training the model on all data available would cause overfitting. In order to avoid this issue, it is common to partition the dataset into at least two splits, one subset for training and another subset to be kept for testing. This technique is called hold out. In some benchmarks (i.e., Reuters-21578, Webkb), datasets are already prepared and partitioned into two splits using the hold-out technique. It is therefore advisable to use the proposed split for the sake of comparison. The hold-out technique is a simple cross-validation technique to evaluate the performance of a model on future/unseen data. There is a number of other cross-validation techniques such as the Leave P Out Cross-Validation (LpOCV) with its particular case, the Leave One Out Cross-Validation (LOOCV) when $p = 1$, the holdout method seen above and the k-fold cross-validation. LpOCV uses p instances as the validation set and the remaining for training. The process is repeated C_p^n where n is the size of the dataset. This technique is almost always computationally infeasible in TC. Even using the LOOCV (LpOCV with $p = 1$), the computational cost will still be very high. A small dataset of a thousand document would need $C_1^{1000} = 1000$ training/testing rounds. It will rapidly jump to about 500000 when $p = 2$. In k-fold cross-validation (see Figure 2.17), k disjoint subsets of the same size are generated from the original dataset. One partition is used for testing the model and the remaining subsets ($k-1$ subsets) are used for training the model. This process is repeated k times, using each time, a different subset as the validation data for testing the model. In classification tasks, it is typical to use the stratified k-fold cross-validation, so that the proportion of class labels in each fold (subset) is roughly equal to the proportion in the original dataset.

2.6.2 Evaluation Metrics

The evaluation of classification models concerns mostly the performance (predictive ability) of those models. Other important evaluation criteria include computation cost, robustness, and scalability.

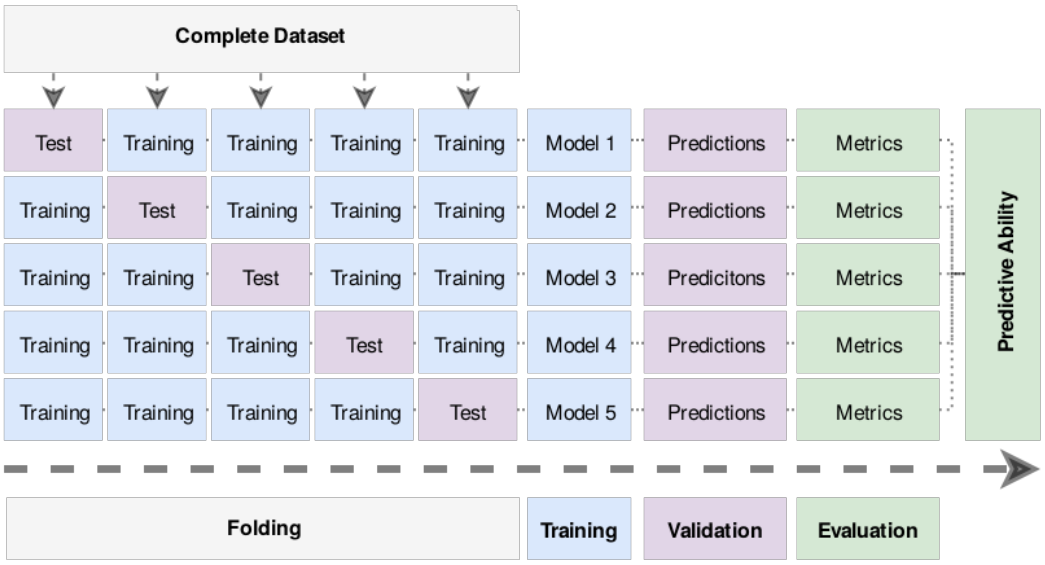

Figure 2.17 k-fold cross-validation process ($k = 5$).

Generally, the performance of a classification model is visualized using a confusion matrix (also called a contingency table) which is a 2×2 table containing four statistical information:

- The number of True Positives TP (i.e., when you correctly predicts the instance positive class).
- The number of True Negatives TN (i.e., when you correctly predicts the instance negative class).
- The number of False Positives FP (i.e., when you faulty predicts the instance belongs to the positive class).
- The number of False Negatives FN (i.e., when you faulty predicts the instance belongs to the negative class).

The confusion matrix shown in Figure 2.18 is fit for binary classifications. For multi-class classification tasks, the confusion matrix may be extended as shown in Figure 2.19 so that it takes the multiple labels into consideration. In this case, the four statistical information TP, TN, FP, and FN are not directly available, and they need to be computed for each label independently of the others as shown in the Figure 2.19.

		Predicted Labels	
		Positives	**Negatives**
Actual Labels	**Positives**	***TP*** True Positives Hits	***FN*** False Negatives Misses, Type II errors
	Negatives	***FP*** False Positives False alarms, type I errors	***TN*** True Negatives Correct rejections

Figure 2.18 Confusion matrix

		Predicted Labels				
		L_1	L_2	L_i	L_n	FN
Actual Labels	L_1	$TP_1 = V_{11}$ $TN_1 = \Sigma_{\substack{1\le j\le n \\ j\neq 1}} V_{jj}$	V_{12}	V_{13}	V_{1n}	$\Sigma_{\substack{1\le j\le n \\ j\neq 1}} V_{1j}$
	L_2	V_{21}	$TP_2 = V_{22}$ $TN_2 = \Sigma_{\substack{1\le j\le n \\ j\neq 2}} V_{jj}$	V_{2i}	V_{2n}	$\Sigma_{\substack{1\le j\le n \\ j\neq 2}} V_{2j}$
	L_i	V_{i1}	V_{i2}	$TP_i = V_{ii}$ $Tn_i = \Sigma_{\substack{1\le j\le n \\ j\neq i}} V_{jj}$	V_{in}	$\Sigma_{\substack{1\le j\le n \\ j\neq i}} V_{ij}$
	L_n	V_{n1}	V_{n2}	V_{n3}	$TP_n = V_{nn}$ $Tn_n = \Sigma_{\substack{1\le j\le n \\ j\neq n}} V_{jj}$	$\Sigma_{\substack{1\le j\le n \\ j\neq n}} V_{nj}$
	FP	$\Sigma_{\substack{1\le j\le n \\ j\neq 1}} V_{j1}$	$\Sigma_{\substack{1\le j\le n \\ j\neq 2}} V_{j2}$	$\Sigma_{\substack{1\le j\le n \\ j\neq i}} V_{ji}$	$\Sigma_{\substack{1\le j\le n \\ j\neq n}} V_{jn}$	

Figure 2.19 A multi-class confusion matrix

Various metrics could be defined using the values in the confusion matrix. Accuracy acc (e.g., the percentage of correct classification) is an obvious choice to evaluate the classification performance of the model. Using the confusion matrix, it could be defined as follows:

$$acc = \frac{TP + TN}{TP + TN + FP + FN}.$$

However, the accuracy metric has two major problems. First, it assumes relatively uniform class distribution. And secondly, it assumes equal cost for incorrect predictions. In order to show the weakness of the accuracy metric in expressing the model true predictive ability, let us consider the problem of predicting the winner in a contest with a hundred participants. In such problem (the portion of true positives is very small compared to that of true negatives), a model that predicts loosing for every participant would be 99% accurate. In order to overcome such issues, other metrics have been introduced, including error rate, likelihood ratios, the Area Under the Curve (AUC), precision and recall, sensitivity and specificity and others [150; 151; 152].

Precision is defined to be the fraction of relevant instances among retrieved ones, while recall is the fraction of relevant instances that have been retrieved over all relevant instances.

$$precision = p = \frac{TP}{TP + FP} \tag{2.18}$$

$$recall = r = \frac{TP}{TP + FN} \tag{2.19}$$

Recall ensures that positive instances are not overlooked, while precision ensures that negative instances are not misclassified as positive cases.

A perfect classifier would have precision and recall equal to one. However, a negative correlation exists between them. Hence, increasing one is generally done at the expense of the other. For this reason, precision and recall are usually evaluated in relation to each other. To this end, many metrics have been proposed with the most popular are the precision-recall Break-Even Point (BEP), Precision-Recall (PR) curve and the F-measure.

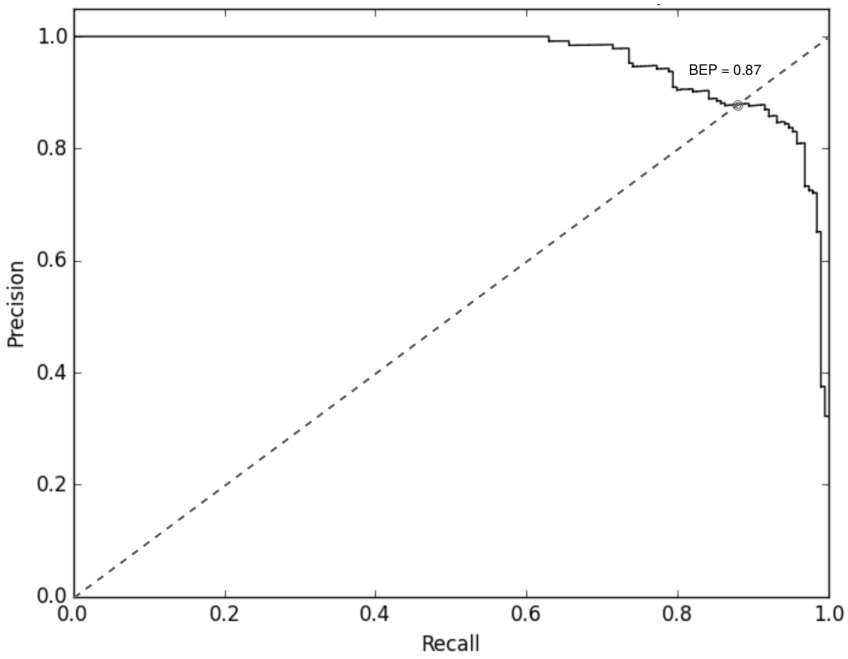

Figure 2.20 The precision recall curve and the break-even point.

The BEP represents the value where precision and recall are equal (see Figure 2.20). In order to obtain the BEP, precision and recall are computed for a number of different decision function thresholds. However, the BEP value is not always achievable. In such cases, an approximate value is considered.

The PR curve (Figure 2.20) is a plot of precision as a function of recall. Similarly to BEP, the PR curve is obtained computing precision and recall at various decision threshold values. Another curve used to analyze the binary classification is the Receiver Operating Characteristic (ROC) curve which is a plot of the True Postive Rate (recall) ($TPR = \frac{TP}{TP+FN}$) as a function of the False Positive Rate ($FPR = \frac{FP}{FP+TN}$). The curve with a larger area under the curve (AUC) usually represents a better model.

F-measure combines precision and recall into a single measure. It is the harmonic mean of precision and recall.

$$F = \frac{TP}{TP + (FP + FN)/2} = 2 \times \frac{p \times r}{p + r}$$

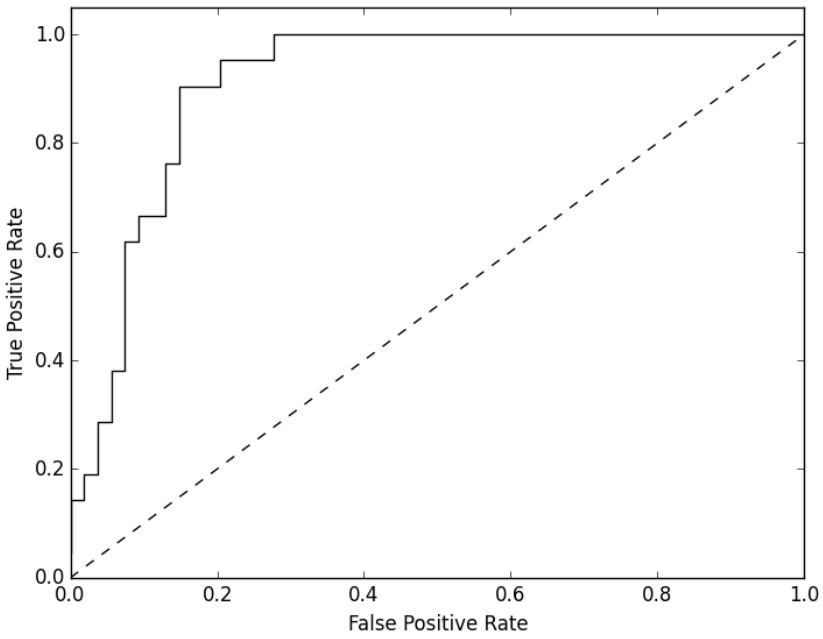

Figure 2.21 Receiver Operating Characteristic (ROC) curve

The above definition of F-measure gives equal importance to precision and recall. A weighted version of F-measure does however exist.

$$F_\beta = (1+\beta^2)\frac{p \times r}{\beta^2 \times p + r} \tag{2.20}$$

The β is a parameter that controls the tradeoff between precision and recall. A $\beta < 1$ focuses on precision, while $\beta > 1$ on recall. The F-measure is very popular for model evaluation in TC [10; 91; 153; 154; 155].

3

Term-Weighting

Term-weighting is the process of assigning weight to terms. Typically, it occurs after the term selection phase. A Term Weighting Scheme (TWS) is used to compute a weight for each term, that illustrates, roughly speaking, how many bits of information it contributes to the semantics of the document. Besides the term-weighting task, weighting methods are used in many other tasks such as Feature Selection (FS), automatic detection of stop-words, etc.

In Text Classification (TC), i.e., the task of automatically assigning a set of predefined categories to a text document, a classifier is learned from a training set of text documents.

Achieving the construction of a classifier involves a number of fundamental steps. Text documents are very complex to be processed directly in their raw format, due to the lack of structured data. Therefore, a critical step in text mining, in general, is to represent text documents in a suitable format compatible with learning algorithms. The most common text representation model in TC is the Vector Space Model (VSM). VSM could be seen as a generalization of the bag-of-words model.

A document in VSM is represented as a vector of index terms. Initially, in the VSM, each term is associated with its number of occurrences. In the term-weighting process, a weighting method is applied that modifies the vector space by assigning a different weight for each term. TWSs, typically, use statistical information from the training set of documents to measure the importance of terms.
TC is a supervised learning task such that statistical information on the membership of documents in categories are known in advance. Depending on whether the method includes such information, we classify the TWSs into two groups, Supervised Term-Weighting (STW) scheme (it uses information about the membership of documents in categories) and unsupervised TWS (the method does not use any prior information on the membership of documents).
Well-known unsupervised TWS are typically derived from text retrieval such as Term Frequency (TF) (term-frequency), the famous Term Frequency-Inverse Document Frequency (TF-IDF) and other variants.
STW methods can be grouped into four categories:

- Term Frequency-Collection Frequency (TF-CF) System.
- Other statistical based methods.
- Schemes evolved via Genetic Programming (GP).
- Methods based on text classifiers.

The most common approach is to combine information theory functions or statistic metrics to weight terms. FS metrics [5; 6] such as Chi-squared (χ^2), Information Gain (*IG*), Gain Ratio (*GR*), Odds Ratio (*OR*) are typical in this category of weighting methods. The effectiveness of these methods in FS made them natural candidates for term-weighting. Other methods from this category also include Relevance Frequency (*RF*) [9], Inverse Category Frequency (*ICF*) [86], ConfWeight which is based on statistical confidence intervals [156], etc.
The second approach uses statistical information to measure the importance of terms however it does not respect the TF-CF system. This category includes Term Class Relevance Measure proposed in [157] and a measure using the Bayes posterior probability appeared in [158].

Another approach appeared in [159] and called Weight Adjusted k-Nearest Neighbor Classification Algorithm (WAKNN). The method tries to maximize an objective function by adjusting vector weights first initialized with values computed using mutual information measure.
All these methods will be detailed further in Section 3.3, except for methods based on GP which will be discussed thoroughly in Chapter 5.

The rest of this chapter is organized as follows: Section 3.1 presents the TWSs in general. Section 3.2 presents the unsupervised group of weighting schemes. Finally, Section 3.3 presents and discusses the different types of supervised TWSs.

3.1 Term-Weighting Schemes (TWS)

In a binary classification task, given a term t_i and a category c_k, CF factor could be expressed using statistical information a, b, c, d and others obtained from the training data:

- a is the number of documents that contain the term t_i and belong to the positive category c_k
- b is the number of documents that do not contain t_i and belong to the positive category c_k
- c is the number of documents that contain t_i and do not belong to c_k
- d is the number of documents that do not contain t_i and do not belong to c_k
- N which stands for the total number of documents
- C is the total number of categories
- C_i is the number of categories that contains documents containing the term t_i

In machine learning and text retrieval, TWSs have been used for more than fifty years. The first trace of scoring using statistical information, precisely the number of occurrences, is due to Luhn and appeared in [160] for keyword indexing. Later,

very fruitful works were done on feature scoring, however, for information retrieval. Salton et al. used the frequency of occurrences in [161]. Jones in [162] introduces the notion of collection frequency and proposes an important new scheme which is known today as Inverse Document Frequency (IDF) noting that non-frequent terms should be treated as more valuable than frequent terms. The basic idea is that frequent terms tend to be general (not specific), and thus, they do not have discrimination ability. Given a term i that occurs n times in a collection of N documents, the weight of the term is formally defined as follows:

$$w_i = f(N) - f(n) + 1 \mid 2^{f(n)-1} < n \leqq 2^{f(n)} \ .$$

Later, in [163], Salton combines the TF with the collection frequency (IDF) introducing the most famous TWS, i.e., TF-IDF. The weight for a term i in a document k could be formulated as follows:

$$w_{ik} = tf_{ik} * idf_k \ .$$

Where $idf_k = \log(\frac{N}{n_k})$.
In [8], Salton et al. propose different variants for TF-IDF pointing out three primary considerations for a text retrieval system that are believed to improve both recall and precision.

- *Term frequency factor (TF)*: The TF factor is used to capture the relative importance of terms in a document. Table 3.1 lists a number of different TF factors. They include the simple binary representation where a score of 1 is assigned to present terms and a score of 0 for missing terms, the popular raw count which represents the number of times the term occurs in a document and different normalized variants ranged from the TF which is normalized by the total number of tokens (instance of terms) in a document to the log normalized variants to the inverse TF.

- *Term discrimination factor/Collection frequency factor (CF)*: The importance of words in a document (TF factor) does not provide enough discrimination ability. A common word like 'The' is frequent in almost all documents, and

Table 3.1 TF Factors

TF factor	**TF Weight**	**Description**
binary	1,0	1 for terms present, 0 otherwise
raw count (f)	f	the number of occurrences
TF	$\frac{f}{\sum f'}$	f normalized by the total number of tokens in the document
log normalization (nlf)	$\log(1+f)$	
double normalization 0.5 (nnf)	$0.5+0.5\frac{f}{\max f}$	f first normalized by maximum TF and further normalized to lie between 0.5 and 1
Inverse term frequency (itf)	$1-\frac{1}{1+f}$	

then it could not separate a group of documents from the remainder of the collection. Hence a discrimination factor is needed to favor those terms that are concentrated in a few documents of a collection. Table 3.2 lists six Collection Frequency (CF) factors.

- *Normalization factor*: In text retrieval, all documents are considered equally important. However, the TF factor will favor large documents over shorter ones, as large documents contain more unique terms and/or greater occurrence values. The normalization factor is helpful to eliminate this length effect. Assuming that w_{ij} is the weight of term t_i in document d_j produced by the multiplication of the TF factor by the CF factor. The normalized weight may then be defined as $\frac{w_{ij}}{\sqrt{\sum_i [w_{ij}^2]}}$.

Based on the discussion above, the weight of a term i in a document j could be expressed as:

$$w_{ik} = TF_{i,j} \times CF_i \ . \tag{3.1}$$

where $TF_{i,j}$ is the TF component, and CF_i is the CF component. Tables 3.1 and 3.2 show the commonly used frequency components.

Table 3.2 CF components

CF factor	Denoted	Description
1.0	1.0	no change in original weight
IDF	IDF	multiply by an inverse document frequency
Chi-squared	χ^2	multiply by χ^2 function
Information Gain	IG	multiply by mutual information function
Gain ratio	GR	multiply by a gain ratio function
Odds Ratio	OR	multiply by an odds ratio function
Relevance Frequency	RF	multiply by a relevance frequency function
Inverse Category Frequency	ICF	multiply by an inverse category frequency

3.2 Traditional Term-Weighting Schemes (TWS)

As mentioned before, unsupervised TWS [8; 162; 164; 165] are generally borrowed from Information Retrieval domain [8; 162] and adopted for TC [5; 87; 88]. TF-IDF is the most famous TWS, proposed in [163]. The TF-IDF is a combination of two components as discussed earlier. Both components have a number of variants. The TF component has multiple normalized variants such as $logf$, $logf+1$, itf, etc (see 3.1). The IDF component also has many variants such as $\log(\frac{N}{n_i+1})$, etc. However, the original variant $\log(\frac{N}{n_i})$ is still the most used.
Besides the raw count (f_ij) representation of TF, there exist numerous other variants such as binary representation ($w_{ij}=1$ if the term t_i occurs in the document d_j and 0 otherwise), $log(f_{ij})+1$, $f_{ij}/\sum_{t'\in d} f_{t',d}$. All these variants are also used as TWSs on their own [5; 6; 8; 87]. The IDF also has a number of variants such as $\log(N/N_i)+1$, $\log((N-N_i)/N_i)$ [8].

3.3 Supervised Term-Weighting Schemes (STW)

As mentioned earlier, supervised methods could be grouped into multiple categories depending on the approach used to compute weights.

- Based on the TF-CF system.

- Based on statistical information.
- Evolved via GP.
- Based on text classifiers.

3.3.1 Term Frequency-Collection Frequency (TF-CF) System

Schemes based on TF-CF system make use of available information on the membership of training documents by replacing the unsupervised IDF component in TF-IDF with another supervised component. Debole et al. and Deng et al. in [5; 6] are the first to take advantage of such information by combining the unsupervised TF component with different supervised CF component. Several comparative studies on these TWSs for both term-weighting and FS has been reported in [6; 10; 87; 91; 118; 166].

Chi-Squared (χ^2)

In the context of TC, χ^2 is a test of independence between a term t and a category c. χ^2 is a popular FS method. χ^2 alongside with other supervised FS metrics has been tested in several papers, as a TWS for TC. For example, Deng et al. in [6], replaced the IDF factor with χ^2 factor, claiming that χ^2 is more efficient than TF-IDF . In contrast, in a similar test, Debole et al. in [5], compare TF-IDF with three supervised TWSs, namely, χ^2, *IG* and Gain Ratio (GR). The authors have found no consistent superiority of these new TWSs over TF-IDF . Given a term t_i and a category c_k, The χ^2 of t_i and c_k is given by:

$$\chi^2 = N \times \frac{(a \times d - b \times c)^2}{(a+c)(b+d)(a+b)(c+d)} \tag{3.2}$$

Information Gain (*IG*)

IG [101] measures the amount of information obtained for category prediction by knowing the presence or absence of a term in a document. *IG* is widely used in FS for TC [5; 6; 10], and in Decision Tree (DT) [138].

$$IG = \left(\frac{a}{N} \times \log \frac{a \times N}{(a+b)(a+c)}\right) + \left(\frac{c}{N} \times \log \frac{c \times N}{(c+d)(a+c)}\right) + \left(\frac{b}{N} \times \log \frac{b \times N}{(a+b)(b+d)}\right) + \left(\frac{d}{N} \times \log \frac{d \times N}{(c+d)(b+d)}\right) \quad (3.3)$$

Gain Ratio (*GR*)

GR was first used in TC for both term-weighting and FS in [5]. It is defined as the ratio between the *IG* of two variables and the entropy of one of them. The authors claim that *GR* is a better term evaluation functions than the *IG*. In their TC test, they confirmed the superiority of *GR* over *IG* and χ^2. Statistically, the definition of the *GR* is given by:

$$GR = \frac{ig}{-\frac{a+c}{N}\log\left(\frac{a+c}{N}\right) - \frac{b+d}{N}\log\left(\frac{b+d}{N}\right)} \quad (3.4)$$

Odds Ratio (*OR*)

OR is a measure that describes the strength of association between two random variables. It was first used as a FS methods by Mladeni'c et al. [118] who found that *OR* outperforms 5 other scoring methods studied in a text-classification experiments. Another comparative study on feature weight in TC is done by Deng et al. in [6]. The study shows good performance of *OR*, however, it is still outperformed by χ^2. The formal definition of *OR* in statistical terms could be expressed as shown below:

$$OR = \log\left(2 + \frac{a * d}{b * c}\right) \quad (3.5)$$

Relevance Frequency (*RF*)

RF is a new supervised weight scheme proposed in [9]. *RF* measures the distribution of term t_i between positive and negative categories and favors those terms that are more concentrated in the positive category than in the negative category. Its formula is defined as

$$RF = \log\left(2 + \frac{a}{\max(1, c)}\right) \quad (3.6)$$

Inverse Category Frequency (*ICF*)

ICF is a new supervised TWS proposed by Wang et al. in [86]. *ICF* stands for inverse category frequency and aims to favor those terms that appear in fewer categories.

$$ICF = \log\left(\frac{C}{C_i}\right) \tag{3.7}$$

ConfWeight

ConfWeight is a TWS proposed by [156] based on statistical confidence intervals. Authors use the Wilson proportion estimate [167] to compute the proportion of documents containing a term t. The Wilson proportion estimate is computed as follows:

$$\tilde{p} = \frac{N_t + 1.96}{N + 3.86} \tag{3.8}$$

Where N is the size of the text collection and N_t is the number of documents that contain the given term. Moreover, its confidence interval at 95% is:

$$\tilde{p} \pm 1.96\sqrt{\frac{\tilde{p}(1-\tilde{p})}{N + 3.84}} \tag{3.9}$$

For a given label, two values $\tilde{p}_+$ and $\tilde{p}_-$ are then computed by applying Equation 3.8 to the positive and negative category respectively. Now, they define the strength of the term t for the positive category as:

$$str_{t,+} = \begin{cases} \log_2(2 \times \frac{MinPos}{MinPos+MaxNeg}) & \text{if MinPos} > \text{MaxNeg} \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

Where MinPos stands for the lower range of the confidence interval of $\tilde{p}_+$, and MaxNeg for the higher range of $\tilde{p}_-$ based on Equation 3.9.

Finally, by using the global policy technique [5], the ConfWeight of term t in a document d is defined as:

$$ConfWeight_{t,d} = \log(tf_{t,d} + 1) * max(str_t^2) \tag{3.11}$$

3.3.2 Based on Statistical Information

As mentioned above, methods in this category include two measures, the first uses the Bayes posterior probability [158], and the second is called Term Class Relevance Measure [157].
Like methods from the TF-CF category, these methods use statistical information in weighting terms. The difference, however, is that they do not adopt the TF-CF system.

Bayes Posterior Probability

In [158], authors use the Bayes formula to compute the posterior probability *Pr(Category | Word)* of each word in the input document. Then, they assign the probability as the weight in the vector space of the VSM. The posterior probability *Pr(Category | Word)* is defined as:

$$Pr(Category \mid Word) = \frac{Occurrence\ of\ word\ in\ Category}{\sum occurrence\ of\ all\ words\ in\ Category} \tag{3.12}$$

Term Class Relevance Measure

Term Class Relevance Measure [157] is a relatively new scheme proposed by Guru and Suhil in 2015. The measure is defined as the ability of a term t_i in classifying a document d_j as a member of class c_k. The measure combines three terms, class weight (c), class term weight (w) and class term density (d). The measure is formally defined as below:

$$r_{ik} = c \times w \times d$$

Where $c = \frac{N_k}{N}$, $w = \frac{a}{N_t}$ and $d = \frac{tf}{\sum tf}$

3.3.3 Term-weighting based on classifiers

Interacting with learning algorithms in order to evaluate terms is not new to machine learning tasks. Wrapper-based FS methods assess multiple classifiers trained on different features subset in order to find an optimal set of features. In embedded FS methods, the weighting process is part of the classifier. A similar method to

the embedded FS was proposed in [168]. The weighting scheme measures the classifier information rate, and then it backpropagates this rate in order to assign weights to features according to how much information they contributed. This information rate measure is based on the performance of the SVM classifier.

Another embedded approach to the classifier-based term-weighting could be found in [159]. In this paper, an iterative algorithm adjusts the feature weights according to a defined measure based on the performance of a k-nearest neighbor classifier. In each iteration, the algorithm slightly modifies the weight of each feature and then checks for classifier performance improvement. The algorithm finally updates the weight of the feature which has the greater impact on the classification accuracy. Although, this method should achieve optimal weights at convergence, the computational cost of the adjustment step is $O(cn^2)$ where c is the number of iterations and n is the number of data points.

4

Contribution: Information Gain Based Term-Weighting Scheme

This chapter is based on my contributions [87; 88]. Traditional classification algorithms are not adapted for multi-label classification tasks. A multi-label classification task is therefore transformed into multiple single-label binary tasks. Though this transformation is the most popular in Text Classification (TC), it introduces two main negative points. First, the strategy does not consider the label dependancy, and secondly, terms are only considered in terms of positive and negative categories. In this chapter, we propose a new Term Weighting Scheme (TWS) based on the Information Gain (*IG*) scheme that, to some extent, solves the second issue, by considering the importance of terms in relevance to all categories without impacting the complexity of the weighting task.

4.1 Information Gain Based Term-Weighting Scheme (IGB)

In TC, terms are given weights using TWSs in order to improve classification performance. A multi-label classification task is generally simplified into several single-label binary tasks. Thus, the term distribution is considered only in terms of the positive category and the negative category. In this paper, we propose a new TWS based on the *IG* measure for multi-label classification tasks. The method tries to overcome this shortness without affecting the complexity of the problem.
Moreover, we examine the impact of our proposed scheme alongside eight well-known TWSs on the performance of five learning algorithms on two popular problems in term of micro- and macro-averaging F_1 measure. We find from the experimental results that the new proposed method outperforms other methods, especially regarding the macro-averaging measure. The basic idea of our proposed *IG* based method comes in the form of a question: how much *IG* a term t_k holds about one category after subtracting the *IG* of the same term t_k in relevance to the other categories. It is to say that the higher the difference between a term *IG* of one category and the average of other categories, the more the term helps in separating the positive from the negative category. As mentioned earlier, a multi-label classification task is transformed into multiple binary single-label classification tasks, therefore, a term has multiple Collection Frequency (CF) weights, one for each binary task. Each weight only considers the distribution of a feature/term in terms of the positive category and the negative category (all documents that do not belong to the positive category). We think that using these weights could be helpful for more effective TWSs.
Considering this idea, we propose a new TWS based on *IG*. Its formula is defined by:

$$w'_{t,c} = w_{t,c} - (\mu_{c' \in C} w_{t,c'} + \sigma_{c' \in C} w_{t,c'})$$

Where $w'_{t,c}$ is the new weight of a term t and a category c, $w'_{t,c}$ is the *IG* score of a term t and a category c, $\mu_{c' \in C} w_{t,c'}$ is the mean of weights on all other categories, and $\sigma_{c' \in C} w_{t,c'}$ is the standard deviation of weights on all other categories.
To evaluate the differences between the *IG* measure and our IGB method, let us

consider the weights for the three terms in Table 4.1. The comparison aims to show that our IGB values describe better the discriminative power of terms compared to the standard TWS by including a score calculated in relevance to all categories.
First, let us clarify some points:

- When $\mu + \sigma > ig$, the term contributes more to the negative categories than to the positive category.
- When $\mu + \sigma < ig$, the term contributes more information to the positive category.
- When $\mu + \sigma = ig$, the term has about the same amount of information about both positive and negative categories.

First, considering the term t_1 in Table 4.1, $\mu + \sigma$ (0.5) is higher than ig value (0.3), which means that the negative categories have higher weights than the positive category. However, the ig value of t_1 is a positive value, in contrary to our new method. That said, the difference does not have a significant impact on scores especially when the number of categories in the corpus is important, as $\mu + \sigma$ will have about the same value.
Now, if we consider terms t_1 and t_3, they both have the same ig value (0.3), which means that they both contribute the same amount of information to the positive category, however by looking at the values of $\mu + \sigma$, t_1 has a value of $0.5 > 0.3$ and t_3 has a value of $0.1 < 0.3$. In this case, we think that t_3 should have a higher value than t_1 as it has the same *IG* in the positive category but smaller *IG* in the negative categories.
Finally, t_2 has the same *IG* value both in the positive category and the negative categories $ig = \mu + \sigma = 0.2$. Thus, the IGB value is equal to 0.

4.2 Benchmark

In this study, we conduct two experiments. The primary purpose of the first experiment (see 4.2.4) is to find the best TWS (if it exists) and to explore the superiority of supervised TWSs. In order to achieve it, we compare eight unsupervised and supervised methods on three well-known benchmark dataset, Reuters-21578,

Table 4.1 Comparison of the weighting values of ig and the proposed method. $\mu + \sigma$ is the average plus the standard deviation of scores of categories which are not the positive category. The values were hand-chosen.

Feature	ig	$\mu + \sigma$	IGB
t_1	0.3	0.5	-0.2
t_2	0.2	0.2	0
t_3	0.3	0.1	0.2

20Newsgroup, and Oshumed, using five different classification algorithms in terms of micro-/macro-averaged F_1 measure [87].
The objective of the second experiment is to validate the effectiveness of our proposed method using the same datasets, classification algorithms and evaluation metrics [88].

4.2.1 Datasets

The Reuters-21578 dataset is one of the most used test collection for TC research. We use the well-known "ApteMod" split [85]. This split includes 10788 documents from the Reuters financial service, divided into a training set of 7769 documents, and a test set of 3019 documents. The dataset is highly skewed, the smallest category contains only two documents, and the biggest contains 3964 documents. Documents in this dataset belongs to one or more categories. This version of the dataset contains ninety categories, and, in our experiments, we report results for the ten most significant categories. Oshumed dataset is extracted from the Oshumed collection compiled by William Hersh[1]. It includes 13,929 medical abstracts (6,286 for training and 7,643 for testing) from the MeSH categories of the year 1991. Each document in this dataset belongs to one or more categories from 23 cardiovascular diseases categories.
The last test collection used in our experiment is the 20 Newsgroups. The dataset "20news-bydate"[2] is sorted by date and split into a training set (about 60%) and a test set (about 40%). Duplicates are removed. Newsgroup-identifying headers

[1] http://disi.unitn.it/moschitti/corpora.htm
[2] http://qwone.com/~jason/20Newsgroups/

(Xref, Newsgroups, Path, Followup-To, Date) are also removed.
In all three test collections, we applied a lowercase transformation, word stemming and stop word removal. No additional preprocessing steps or Feature Selection (FS) is performed.
Reuters-21578 and Oshumed are multi-labeled datasets. 20Newsgroups is a multi-class dataset. In all cases, we transform the task into multiple binary single label tasks using the one-vs.-all transformation strategy aka one-vs.-rest.
Table 5.3 shows some statistics on the three collections.

Table 4.2 Statistics on the three test collections (train data/test data).

	Reuters	Oshumed	Newsgroups
# documents	7769/3019	6286/7643	11314/7532
# terms	26000	30198	101322
# categories	90	23	20
size of the smallest category	1/1	65/70	377/251
size of the largest category	2877/1087	1799/2153	600/399

4.2.2 Classifiers

Generally, the performance of a TWS is assessed on known benchmarks by evaluating a classification model on Vector Space Model (VSM) representation of this TWS. In order to build the classification models, we experiment five different algorithms, namely: Passive-Aggressive (PA), C4.5, Support Vector Machine (SVM), Stochastic Gradient Descent (SGD), and Nearest Centroid (NC).

SVMs are a set of supervised machine learning methods introduced by Boser *et al.* Developed from statistical learning theory, SVMs have shown good performance in many fields. In TC, Joachims used SVM in which he demonstrates the better efficiency of SVM over other learning algorithms namely Naive Bayes, Rocchio, C4.5 and k-NN [85]. PA proposed by Crammer *et al.* is a learning algorithm focused on online learning and large-scale dataset [148]. The method treats a flow of documents and outputs a prediction once a document is received. Later at any time a document true-label is discovered, the method redefines its prediction

function. SGD classifier [149] is a linear classifier such as linear SVM, PA that uses SGD for training. This classifier is also used for large-scale categorization problem. NC [169] is a neighborhood-based classification algorithm and C4.5 [139] is state of the art supervised learning algorithm based on the Decision Tree (DT).

4.2.3 Evaluation

As mentioned in Subsection 2.6.2, numerous evaluation metrics exist to evaluate the classification models such as F_1 measure. The F_1 measure can be considered as a weighted average of the precision (the fraction of positive predictions that is correct) and recall (the fraction of actual positives that have been correctly classified) and can be formally defined as:

$$F_1 = \frac{2 * recall * precision}{recall + precision} \quad .$$

Generally, the F_1 measure is computed in two ways, micro-averaged and macro-averaged. In micro-averaged, big categories are emphasized while in macro-averaged, all categories have the same importance.
Underlined results represent the highest score over a column, and the bolded results are the best pair of micro-/macro-averaged F_1 scores when all the classifiers and all the TWSs are considered. The pair having the highest mean is chosen as the best.

4.2.4 Experiments

A Comparative Study on Term-Weighting Schemes (TWS) for Text Classification (TC)

This section is based on the article that appeared in [87]. In this experiment, we are interested in finding a proper TWS by studying the impact of eight TWSs on five classification algorithms.

In tables 4.3, 4.4 and 4.5, we present the micro- and macro-averaged precision, recall, and f-score, respectively, for the Reuters-21578 dataset. In Table 4.5, NC shows the lowest performance, considering both micro- and macro-averaged

scores. PA has the highest micro-averaged score (87.22%) and the second highest macro-averaged score (48.51%) preceeded only by C4.5 with an macro-averaged score of 54.24%. Regarding TWS, even though, Term Frequency-Inverse Document Frequency (TF-IDF) shows higher scores, the results are very close.

Tables 4.6, 4.7 and 4.8 shows the micro-/macro-averaged precision, recall, and f-score, respectively, for Oshumed dataset. Considering both precision and recall scores in Table 4.8, PA shows the best performance, followed by SGD, SVM. Strangely C4.5 shows the lowest performance.
Regarding TWS, Term Frequency-Odds Ratio (TF-OR) outperforms all other methods except when used in conjunction NC. Term Frequency-Relevance Frequency (TF-RF), Term Frequency-Gain Ratio (TF-GR) and Term Frequency-Information Gain (TF-IG) have close results, and come second, followed by Term Frequency (TF). TF-IDF and Term Frequency-Inverse Category Frequency (TF-ICF) perform poorly.
For these two datasets, we can note that, in comparison with the other algorithms, NC has very high recall scores (see Tables 4.4 and 4.7). However, NC reports the lowest precision scores (see Tables 4.3, 4.6).
Scores for Newsgroups dataset are presented in Tables 4.9, 4.10 and 4.11. TF-IG and TF-GR record the best scores (70%/69%) in conjunction with both SVM and SGD. Overall, in this dataset, SVM performs the best, followed by SGD and PA. C4.5 records very low scores. As for TWS, TF-IG and TF-GR give the best results, followed closely by TF-OR, TF-IDF and TF-ICF. TF-RF shows the lowest scores.
In contrast to the high recall scores and low precision scores registered by NC algorithm on Reuters-21578 and Oshumed datasets, NC registered approximately equal results on both precision and recall.
Concerning C4.5, we can note that precision and recall results are approximately equal on the three datasets.
Overall, in our study, we find that TF-OR is the best TWS. TF-IDF, TF-GR and TF-IG are also good choices for weighting features. Term Frequency-Chi Squared (TF-χ^2), TF-ICF and TF-RF are the worst methods.

Table 4.3 micro-/macro-averaged precision results (%) on Reuters-21578 dataset using different weighting methods.

	PA	SVM	SGD	NC	C4.5
TF	91.50/62.69	94.37/56.75	94.40/56.64	39.22/30.28	82.17/57.17
TF-χ^2	91.35/63.23	94.37/56.75	94.46/55.54	39.22/30.28	82.18/56.23
TF-GR	91.26/61.37	94.37/56.75	94.48/57.21	39.22/30.28	82.44/55.26
TF-ICF	93.21/64.03	94.95/57.31	94.69/61.25	48.87/50.00	81.64/55.34
TF-IDF	**93.12/64.14**	95.17/56.95	94.45/58.85	63.40/47.57	81.82/56.65
TF-IG	91.56/62.63	94.37/56.75	94.48/56.68	39.22/30.28	82.45/58.53
TF-OR	91.73/63.42	94.37/56.75	94.47/56.62	39.22/30.28	82.07/56.24
TF-RF	91.51/60.75	94.37/56.75	94.45/55.52	39.22/30.28	81.93/55.63

Table 4.4 micro-/macro-averaged recall results (%) on Reuters-21578 dataset using different weighting methods.

	PA	SVM	SGD	NC	C4.5
TF	82.27/42.74	78.85/33.51	79.73/35.08	**89.93/61.76**	81.62/53.79
TF-χ^2	81.76/42.64	78.85/33.51	79.62/34.82	**89.93/61.76**	81.41/52.91
TF-GR	82.27/41.55	78.85/33.51	79.54/34.81	**89.93/61.76**	81.62/51.81
TF-ICF	79.59/39.44	75.27/30.64	77.19/33.20	86.75/52.96	80.26/51.91
TF-IDF	82.02/41.81	78.37/33.60	80.02/36.29	87.55/55.60	80.80/53.65
TF-IG	82.61/41.93	78.85/33.51	79.51/34.81	**89.93/61.76**	81.70/53.51
TF-OR	82.10/42.59	78.85/33.51	79.46/34.66	**89.93/61.76**	81.68/53.32
TF-RF	82.00/41.15	78.85/33.51	79.57/34.75	**89.93/61.76**	81.97/52.97

Results for Information Gain-Based Method (IGB)

Table 4.12 and Table 4.13 show the micro-/macro-averaged F_1 performances of different TWSs using linear SVM for the two datasets Reuters and Oshumed, respectively.

Considering the Reuters dataset, the best micro-averaged F_1 score 88.68% is achieved by using our IGB method using the SVM classifier. Regarding the macro-averaged F_1 score, using IGB gives the best score 57.70%. The best micro-/macro-

Table 4.5 micro-/macro-averaged f-score results (%) on Reuters-21578 dataset using different weighting methods.

	PA	SVM	SGD	NC	C4.5
TF	86.64/48.48	85.91/39.74	86.45/41.15	54.61/34.75	81.90/53.63
TF-χ^2	86.29/48.51	85.91/39.74	86.41/40.80	54.61/34.75	81.79/53.24
TF-GR	86.53/47.14	85.91/39.74	86.37/41.14	54.61/34.75	82.03/51.79
TF-ICF	85.87/46.42	83.97/37.77	85.05/40.28	62.52/46.43	80.94/52.05
TF-IDF	**87.22/48.20**	85.95/40.32	86.64/42.73	73.55/47.05	81.31/53.36
TF-IG	86.86/47.76	85.91/39.74	86.35/40.97	54.61/34.75	**82.08/54.24**
TF-OR	86.65/48.48	85.91/39.74	86.32/40.85	54.61/34.75	81.87/52.82
TF-RF	86.49/46.74	85.91/39.74	86.37/40.76	54.61/34.75	81.95/52.82

Table 4.6 micro-/macro-averaged precision results (%) on Oshumed dataset using different weighting methods.

	PA	SVM	SGD	NC	C4.5
TF	71.13/73.55	78.77/81.13	79.85/82.42	39.22/35.64	57.09/53.40
TF-χ^2	64.72/61.40	72.81/71.56	71.57/69.70	47.34/45.23	58.22/56.02
TF-GR	76.17/78.21	**81.04/80.14**	80.67/79.39	58.65/58.85	56.72/52.89
TF-ICF	74.27/75.04	80.80/81.07	77.92/77.81	69.32/68.58	55.63/53.09
TF-IDF	75.76/78.26	80.83/80.36	80.48/79.11	54.40/53.06	57.54/53.61
TF-IG	76.14/77.84	**81.04/80.14**	80.81/79.49	58.65/58.85	56.84/53.65
TF-OR	74.25/76.45	79.74/81.91	79.44/81.19	53.58/53.61	57.34/54.62
TF-RF	74.08/76.38	80.29/83.20	80.39/82.11	52.24/52.12	57.64/54.63

averaged F_1 pair 87.29%/57.70% is also achieved by IGB. Compared to the second best pair (87.27%/48.59%) achieved by TF-OR, the proposed method records a boost of over 9% in terms of macro-averaged F_1.

In terms of learning algorithms, in this experiment, PA, SVM, and SGD show comparable performances. NC records the lowest results.

Considering the Oshumed dataset, the highest micro-averaged F_1 (67.45%) is achieved using our IGB method. The highest macro-averaged F_1 (62.37%) achieved

Table 4.7 micro-/macro-averaged recall results (%) on Oshumed dataset using different weighting methods.

	PA	SVM	SGD	NC	C4.5
TF	52.91/44.32	46.21/35.96	47.27/37.76	**68.04**/**66.55**	56.08/51.73
TF-χ^2	56.50/51.22	54.83/48.33	50.77/45.94	64.82/65.07	56.70/52.42
TF-GR	54.89/47.80	48.61/40.01	52.08/44.79	66.60/64.62	56.89/52.70
TF-ICF	45.57/40.16	35.50/29.46	42.02/36.32	51.58/47.07	57.43/52.71
TF-IDF	53.55/45.80	46.82/37.43	50.19/42.00	67.71/65.54	56.35/52.50
TF-IG	54.76/47.84	48.61/40.01	51.96/44.73	66.60/64.62	57.36/53.67
TF-OR	58.15/53.84	54.68/48.36	56.53/51.26	66.14/66.29	55.89/51.46
TF-RF	56.38/50.68	52.32/44.55	53.69/46.72	66.32/65.98	55.53/51.79

Table 4.8 micro-/macro-averaged f-score results (%) on Oshumed dataset using different weighting methods.

	PA	SVM	SGD	NC	C4.5
TF	60.68/53.95	58.25/47.02	59.39/48.78	49.76/44.48	56.58/52.42
TF-χ^2	60.33/55.51	62.55/55.27	59.40/52.01	54.72/51.83	57.45/53.88
TF-GR	63.80/58.11	60.77/51.71	63.29/56.05	62.37/60.16	56.80/52.65
TF-ICF	56.48/51.32	49.33/41.93	54.60/48.32	59.15/55.25	56.51/52.67
TF-IDF	62.75/56.42	59.30/49.08	61.83/53.41	60.33/57.43	56.94/52.88
TF-IG	63.71/58.12	60.77/51.71	63.25/56.02	62.37/60.16	57.10/53.47
TF-OR	**65.22**/**62.37**	64.87/58.78	66.05/60.57	59.20/57.43	56.60/52.76
TF-RF	64.03/60.08	63.36/55.52	64.38/57.19	58.44/56.05	56.56/53.00

by using TF-OR. As a pair of micro- and macro-averaged F_1, the proposed method has a slightly higher average.

Concerning the learning algorithms, SGD and SVM perform the best followed closely by PA and finally, NC and C4.5 show the lowest results.

Overall, in our study, we find that the proposed method gives good results, better than the standard TWSs. TF-OR, TF-RF, TF-IDF and TF-IG have also shown good results. $tf.\chi^2$ and TF-ICF give the worst results.

Table 4.9 micro-/macro-averaged precision results (%) on 20newsgroups dataset using different weighting methods.

	PA	SVM	SGD	NC	C4.5
TF	63.91/63.77	66.94/66.58	61.05/62.78	55.91/62.22	44.07/44.12
TF-χ^2	58.55/60.54	60.26/60.35	59.33/59.51	47.73/60.20	38.20/38.16
TF-GR	68.43/68.37	69.69/69.41	**70.19**/**70.06**	62.85/71.44	43.07/43.37
TF-ICF	68.14/68.23	69.15/69.23	69.24/68.85	59.43/71.87	49.19/51.77
TF-IDF	68.31/68.10	69.69/69.29	61.26/66.59	64.27/69.19	43.65/43.67
TF-IG	68.97/68.86	70.14/69.79	70.26/69.85	63.64/71.56	44.16/44.40
TF-OR	68.57/68.19	69.80/69.26	69.54/69.24	56.44/69.03	45.13/44.77
TF-RF	56.00/55.42	57.73/56.95	56.57/55.36	36.56/46.22	42.22/42.58

Table 4.10 micro-/macro-averaged recall results (%) on 20newsgroups dataset using different weighting methods.

	PA	SVM	SGD	NC	C4.5
TF	63.91/62.87	66.94/65.81	61.05/59.69	55.91/55.17	44.07/43.01
TF-χ^2	58.55/57.05	60.26/58.74	59.33/57.82	47.73/47.05	38.20/37.17
TF-GR	68.43/67.33	69.69/68.52	70.19/68.92	62.85/62.10	43.07/42.07
TF-ICF	68.14/66.96	69.15/67.90	69.24/67.95	59.43/58.62	49.19/48.15
TF-IDF	68.31/67.20	69.69/68.48	61.26/59.95	64.27/63.32	43.65/42.73
TF-IG	68.97/67.86	70.14/68.93	**70.26**/**68.94**	63.64/62.78	44.16/43.15
TF-OR	68.57/67.42	69.80/68.52	69.54/68.26	56.44/55.81	45.13/44.05
TF-RF	56.00/54.86	57.73/56.38	56.57/55.10	36.56/36.06	42.22/41.33

In this study, we experimented with a new TWS applied to multi-label TC based on the *IG* measure. The basic idea is that the *IG* weight of a feature in negative categories should affect the importance of this term in the positive category.

We studied the effectiveness of the IGB method in comparison to eight well-known TWSs applied to TC tasks.

Experimental results show that our method outperformed all other methods tested in this study, especially regarding the macro-averaged measure.

Table 4.11 micro-/macro-averaged f-score results (%) on 20newsgroups dataset using different weighting methods.

	PA	SVM	SGD	NC	C4.5
TF	63.91/63.06	66.94/65.85	61.05/60.38	55.91/56.97	44.07/43.18
TF-χ^2	58.55/56.89	60.26/58.18	59.33/57.21	47.73/50.62	38.20/36.91
TF-GR	68.43/67.55	69.69/68.60	70.19/68.98	62.85/64.45	43.07/42.36
TF-ICF	68.14/67.18	69.15/68.10	69.24/67.97	59.43/61.74	49.19/49.08
TF-IDF	68.31/67.37	69.69/68.49	61.26/62.38	64.27/64.90	43.65/42.86
TF-IG	68.97/68.05	70.14/68.96	**70.26/68.93**	63.64/65.09	44.16/43.39
TF-OR	68.57/67.52	69.80/68.51	69.54/68.30	56.44/59.19	45.13/43.97
TF-RF	56.00/54.69	57.73/56.18	56.57/54.47	36.56/38.41	42.22/41.47

Table 4.12 micro-/macro- averaged F_1 results (%) on Reuters-21578 corpus using eight standard TWSs and the proposed method.

	PA	SVM	SGD	NC	C4.5
TF	86.6/48.5	85.9/39.7	86.4/41.1	54.6/34.7	81.9/53.6
TF-χ^2	86.3/48.5	84.8/43.9	86.4/40.8	54.6/34.7	81.8/53.2
TF-IDF	87.2/48.2	85.7/40.3	86.6/42.7	73.5/47.0	81.3/53.4
TF-GR	86.5/47.1	86.5/42.4	86.4/41.1	54.6/34.7	82.0/51.8
TF-OR	86.6/48.5	87.3/48.6	86.3/40.8	54.6/34.7	81.9/52.8
TF-IG	86.9/47.7	86.5/42.4	86.3/41.0	54.6/34.7	82.1/54.2
TF-ICF	85.9/46.4	84.0/37.8	85.0/40.3	62.5/46.4	80.9/52.0
TF-RF	86.5/46.7	87.8/45.3	86.4/40.8	54.6/34.7	82.0/52.8
New	**87.3/57.7**	88.7/51.7	88.4/49.0	66.5/54.7	82.2/51.3

4.3 Conclusion

In this chapter, we first got an insight into eight different TWSs. These schemes are used in conjunction with five classifiers tested on Reuters-21578, Oshumed, and 20newsgroups datasets. The work aims at extending previous surveys and establishing a clean and fair basis for TC benchmarks. Secondly, we introduced

Table 4.13 micro-/macro- averaged F_1 results (%) on Oshumed using eight standard TWSs and the proposed method.

	PA	SVM	SGD	NC	C4.5
TF	60.7/54.0	58.2/47.0	59.4/48.8	49.8/44.5	56.6/52.4
TF-χ^2	60.3/55.5	62.5/55.3	59.4/52.0	54.7/51.8	57.4/53.9
TF-IDF	62.7/56.4	59.3/49.1	61.8/53.4	60.3/57.4	56.9/52.9
TF-GR	63.8/58.1	60.8/51.7	63.3/56.0	62.4/60.2	56.8/52.6
TF-OR	65.2/62.4	64.9/58.8	66.0/60.6	59.2/57.4	56.6/52.8
TF-IG	63.7/58.1	60.8/51.7	63.2/56.0	62.4/60.2	57.1/53.5
TF-ICF	56.5/51.3	49.3/41.9	54.6/48.3	59.1/55.2	56.5/52.7
TF-RF	64.0/60.1	63.4/55.5	64.4/57.2	58.4/56.0	56.6/53.0
New	64.4/60.8	67.0/60.8	**67.4/61.2**	59.4/57.0	56.7/52.5

a new TWS based on *IG* for the multi-label classification task. The basic idea is that the *IG* weight of a feature in negative categories should affect the importance of this term in the positive class. We studied the effectiveness of our method in comparison to the presented TWSs applied to TC tasks. To sum up, from the first experiment comparing eight different , we find that TF-OR gives slightly better results, however, the superiority of Supervised Term-Weighting (STW) methods over unsupervised methods is still not clear. Results from the second experiment show that the IGB method outperforms all other methods tested in this study, particularly regarding the macro-averaged measure. Therefore, we conclude that weighting a term by only considering its relevance to the positive category and the negative category has a real negative impact on the classification results and that even a partial solution to the problem has significantly improved the performance.

5

Contribution: Evolving Term-Weighting Scheme using Genetic Programming

This chapter is based on my contributions [170; 171; 172].

5.1 Introduction

Text Classification (TC) aims to automatically assign a set of predefined categories to a text document based on their content. TC is an important machine learning problem that has been applied to numerous applications such as spam filtering [173], language identification [174], authorship recognition [175], sentiment analysis [176], and so on. Generally, the TC approach is to learn an inductive classifier from a set of predefined categories. This approach requires that documents are rep-

resented in a suitable format such as the Vector Space Model (VSM) representation (Salton and Buckley, 1988).

In a VSM, a document d_j is represented by a term vector $d_j = (w_{1,j}, w_{2,j}, ..., w_{t,j})$ where each term is associated with a weight $w_{k,j}$.

The weight represents how much a term contributes to the semantics of a document. The method which assigns a weight to a term is called Term Weighting Scheme (TWS).

Numerous TWSs exist, and we introduce the most famous in Section 5.2. They are generated according to human a priori and mathematical rules. TWSs are usually simple mathematical expressions. Unfortunately, depending on the application, it is not easy to know a priori which TWS will be effective.

As expression discovery may naturally be addressed by Genetic Programming (GP) [177], we are interested in this chapter to study the effectiveness of GP generated formulas and their aspects. We are also interested to know if a stochastic evolutionary process with no information about the complexity, the shape and the size of the expression can find at least competitive discriminative TWS.

The chapter is organized as follows: Section 5.2 presents the TWSs and related works. In section 5.3 we present GP and how it is applied to TWS. Section 5.4 presents the experiments and the results, and then we conclude in section 5.5.

5.2 Term-Weighting Schemes (TWS)

TC is a supervised learning task. Hence, the training data consists of a set of labeled documents

$$D = ((d_1, l_1), ..., (d_N, l_N)),$$

such that d_j is the term vector of j-th document, l_j is its label and N is the total number of training documents. As in VSM representation, a document d_j is represented by a term vector $d_j = (w_{1,j}, w_{2,j}, ..., w_{k,j})$ where $w_{i,j}$ is a weight assigned to the term t_i of the document d_j and determined by the TWS.

5.2.1 Statistical Information

Generally, a multi-labeled classification task is turned into several distinct single-label binary task, one for each label, using the Binary Relevance (BR) transformation strategy. That is, given the list of labels $L = \{l_1, l_2, ..., l_m\}$, the original dataset is transformed into m different datasets $D = \{D_1, D_2, ..., D_m\}$. For each dataset D_k, documents having the label l_k will be tagged as the positive category c_k and the rest as the negative category $\overline{c_k}$. Weights are then computed independently for each binary dataset.

Based on the BR transformation, given a term t_i and a category c_k, TWS could be expressed using statistical information a, b, c, and d obtained from the training data:

- a is the number of documents that contain the term t_i and belong to the positive category c_k.
- b is the number of documents that do not contain t_i and belong to the positive category c_k.
- c is the number of documents that contain t_i and do not belong to c_k.
- d is the number of documents that do not contain t_i and do not belong to c_k.

Using these statistics, the Inverse Document Frequency (IDF) is generally expressed as $idf(t_i, D) = \log \frac{N}{|d \in D : t \in d|}$ could also be expressed as $idf(t_i, D) = \log \frac{N}{a+c}$ where $N = a + b + c + d$ is the total number of documents in the training data. Besides the statistics described above, Table 5.1 shows different statistical information that could be extracted from the training data.

5.2.2 Short Review

As detailed in Chapter 3, term-weighting has been approached by a wide range of approaches - from the traditional unsupervised TWS such as Term Frequency-Inverse Document Frequency (TF-IDF), the supervised methods [6; 9; 86; 88; 156]

Table 5.1 Statistical information (Terminals) used to evolve a TWS.

Label	Description
N	Total number of documents
C	Number of categories
C_t	Number of categories that contain the term t
N_t	Number of documents that contain t
$\overline{Nt}$	Number of documents that do not contain t
N_{cat}	Number of documents in the positive category cat
$\overline{N_{cat}}$	Number of documents that do not belong to cat

and methods based on statistical information [157; 158] to methods based on classifier performance [159; 168].

Another approaches have also been proposed to learn TWSs via GP in [178; 179; 180; 181; 182; 183], however, these studies have focused on information retrieval problem. For TC, a similar approach proposed by Escalante et al. in [184]. However our study differs in two ways: first, Escalante et al. try to generate new TWSs by combining existing TWS, and secondly, they learn a single TWS for each dataset whereas we learn a TWS for each category in a dataset. In our work, we generate TWSs by combining statistical information at a microscopic level to evolve new TWSs. We also extend the study on the thematic TC. We hope this leads to more robust nonhuman based TWSs.

5.3 Genetic Programming (GP)

Evolutionary computing is based on Darwin's theory of "survival of the fittest". The main scheme of evolutionary algorithms is to evolve a population of individuals that are randomly generated. Each represents a candidate solution that undergoes a set of genetic operators that allow to mix and alter partial solutions. One of the key features of evolutionary algorithms is that they are stochastic schemes.

5.3.1 Introduction

GP belongs to the family of evolutionary algorithms. Crammer et al. first proposed it [185] and then popularized by Koza [186]. Unlike genetic algorithms where the aim is to discover a solution, the goal of GP is to find out a computer program that can solve a problem. Figure 5.5 outlines the flow of the GP algorithm.

In GP, a set of random expressions that usually represent computer programs are generated. As in all evolutionary computation algorithms, this set of programs will evolve and change dynamically during the evolution. What makes GP suitable for a number of different applications is that these computer programs can represent many different structures, such as mathematical expressions for symbolic regression [187], Decision Tree (DT) [188], programs that control a robot [189; 190] to fulfill a certain task or programs that are able to predict defibrillation success in patients and so on.

The quality of a candidate solution (i.e., a program) is usually assessed by confronting it with a set of fitness cases. This step is usually the most time-consuming step as the programs may get huge and several thousands of candidate programs are usually evaluated at each generation. These computer programs will undergo one or several evolutionary operators that will alter in a hopefully beneficial way. The most classical evolutionary operators are usually the crossover operator that allows the exchange of genetic material (in our case subtrees) and the mutation operator that allows a small alteration to the program.

In the most conventional GP approach, programs are usually depicted by trees. In GP terminology, the set of nodes are split into two sets, inner nodes of the tree are drawn from a set of functions while the terminal nodes (leaves) are drawn from a so-called terminal set. Depending on the problem, the set of functions can be mathematical functions, boolean functions, the control flow functions (if,...), or any functions that may be suitable to solve the given problem. The terminal set is usually the set of inputs of the problem, e.g., parameters and constants for symbolic regression problems, sensors for robot planning and so on.

When the stopping criterion is reached, the best individual is returned, otherwise, the loop continues. A set of individuals are selected (according to some criteria) to produce the next generation. Those individuals are called parents. A

number of different selection strategy exists such as Elitism, Tournament Selection (TS), Roulette Wheel Selection (RWS) and Random Selection (RS) in which parents are randomly selected. In Elitism, the best individuals of the current generation are carried to be part of the next generation without being modified. TS is a popular selection method in which the best individual from a fixed number of random individuals is selected. The algorithm loops to select all parents. In RWS, each individual is represented as a region proportional to its fitness value which could be done by normalizing fitness values to one. An individual is then selected by randomly choosing a number between zero and one.

Various methods were also proposed for the crossover (i.e., single-point crossover and two-point crossover) and the mutation (i.e., uniform mutation, gaussian mutation) operators. Crossover operator combines the genes of two individuals to produce new offsprings. The single-point crossover splits two individuals at a randomly chosen cut-off point and then swaps the tails. In a similar way, the

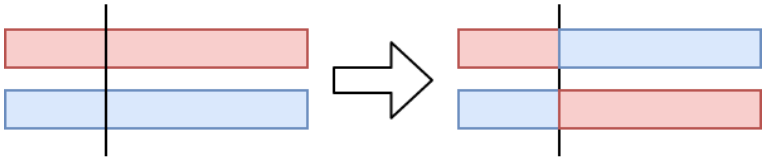

Figure 5.1 Single-point crossover.

two-point crossover trades parts between two different cut-off points. Figure 5.1 and Figure 5.2 show the process of the two approaches.

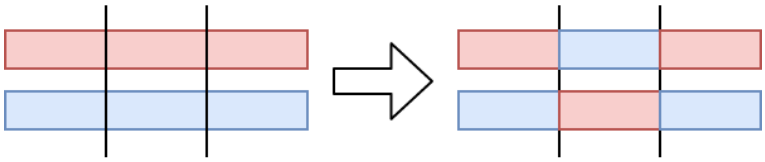

Figure 5.2 Two-point crossover.

Mutation aims to maintain diversity in the future generation by performing simple modifications to one or more genes according to some distribution. In uniform mutation, the value of a random gene is replaced by a random value selected uniformly from the input values. The reader can refer to [186; 191] for more information.

5.3.2 Evolving Term-Weighting Scheme (TWS) using Genetic Programming (GP)

A Collection Frequency (CF) factor is a combination of statistical information. It is intended to measure the discriminative power of a term, i.e., it tells how much a term is related to a certain category. These statistics are combined by means of mathematical operators and functions.

We are interested in automatically evolving a CF factor (an individual) using GP. In our approach, the learned CF factor combined with the Term Frequency (TF) factor forms a TWS.

In our context of automatically evolving TWSs, an individual is a combination of the function set that is built with simple arithmetical operators (+,-,*,/,log,...) and the terminal set (constant values and inputs to our problem).

Table 5.1 shows the statistical information used as a terminal set for generating formulas (the function set) which represent CF factors. As it can be seen, the function set is made of very simple arithmetical functions while the terminal set includes to the best of our knowledge all the statistical information used to build a TWS.

As previously mentioned, programs (generated TWS) are depicted as trees. In this problem, the terminal nodes consist of statistical information extracted from the training data, while the inner nodes are a set of defined operators that combines the statistical information to form a new TWS.

Figure 5.3 describes the representation graphically and Table 5.2 shows the parameters used in the GP algorithm.

Terminals and Function Set

In this study, we try to generate new TWS by evolving the CF factor and then combines it with the TF factor. The CF factor is a combination of constants, statistical information (N, N_t,...), and mathematical operators. Hence we define the terminals as the statistical information shown in Table 5.1. Regarding the mathematical operators, they are defined as one of the following ($+$, $-$, $/$, $*$, $\sqrt{x}$, $log1(x) = \log(1+x)$ and $log2(x) = \log(2+x)$).

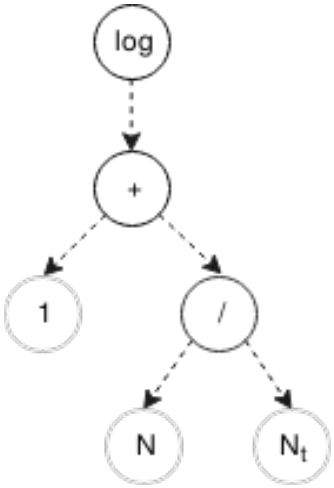

Figure 5.3 Representation for the IDF component $\log(1 + \frac{N}{N_t})$ in GP.

Table 5.2 Parameters used in our GP.

Parameter	Value	
Population Size	100	
Max Individual Size	20	
Number of generations	100	
Function set	$+, -, /, *, \sqrt{x}$,	
	$log1(x) = \log(1 + x)$,	
	$log2(x) = \log(2 + x)$	
Terminal set	$a, b, c, d, N, N_t, verline N_t$,	
	$N_{cat}, \overline{N_{cat}}, C, C_t$	
Mutation	Type	OnePointMutation
	Probability	1/individual size
CrossOver	Type	SubtreeCrossover
	Probability	0.85

We should note that the statistical information has different types (single value, vector, and matrix). For instance, the number of documents in the training data N is a constant (single value), the number of documents that contains a term t is a vector containing the number of documents for each term and finally, the number of documents that belongs to a category cat and contains a term t is a matrix. Operations on these different types of statistical information are taken care of by Eigen[1] library using element-wise transformations.

Genetic Operators

In GP, a set of individuals is initialized and then evolved according to a set of genetic operators. At first, we randomly generate a random size individuals with a max size of twenty genes (the max size could be overpassed during the cross-over operation). As for genetic operators, we use the elite selection and re-insertion, a subtree crossover with a probability of 0.85 and one point mutation with a probability of $1/$*size of the individual*.
Figure 5.4 shows the crossover and mutation operations for our expression discovery problem.

Fitness Function

Generally, the performance of a TWS is assessed on a known benchmark by evaluating a classification model on VSM representation of this TWS. Numerous evaluation metrics exist to evaluate the classification models such as f_1 measure. Evaluating the classification model is a vital step that affects the performance of the GP. However, it could be very time-consuming. Hence, it is important to choose a good and fast machine learning algorithm. LibLinear [192] is an open source library for large-scale linear classification. It supports logistic regression and linear Support Vector Machine (SVM)s.

In our study, once a new individual is generated, we perform a 3-fold cross-validation on the training data which generates three disjoint subsets. We use two subsets as the training set and one subset as the test set. The process is repeated three times, each time using a different subset for testing. The performance is

[1]http://eigen.tuxfamily.org/

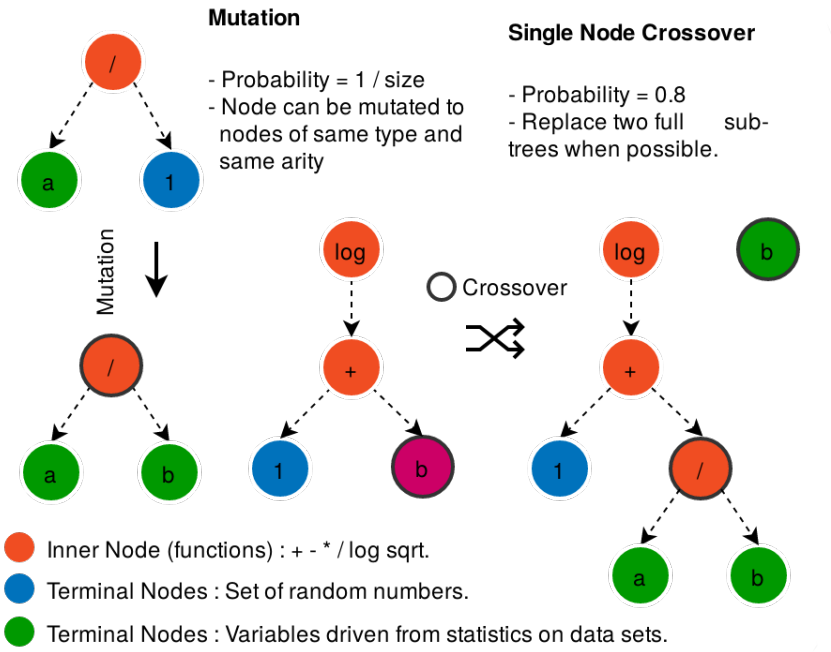

Figure 5.4 One node crossover and uniform mutation operations for expression discovery.

measured using the f_1 measure. The average classification performance is used as the fitness function. The f_1 measure considers both precision p (true positive over true positive plus false positive) and recall r (true positive over true positive plus false negative) and can be formally defined as $f_1(p,r) = \frac{2rp}{r+p}$.

Figure 5.5 describes the course of the evolution of individuals graphically.

5.4 Experiments and Results

This section presents an empirical evaluation of the proposed approach. The goal of this study is to assess the effectiveness of the generated TWSs and compare their performances to standard TWSs including TF-IDF, Term Frequency-Information Gain (TF-IG), Term Frequency-Chi Squared (TF-χ^2), Term Frequency-Odds Ratio (TF-OR), Term Frequency-Relevance Frequency (TF-RF) and Term Frequency-Inverse Category Frequency (TF-ICF).

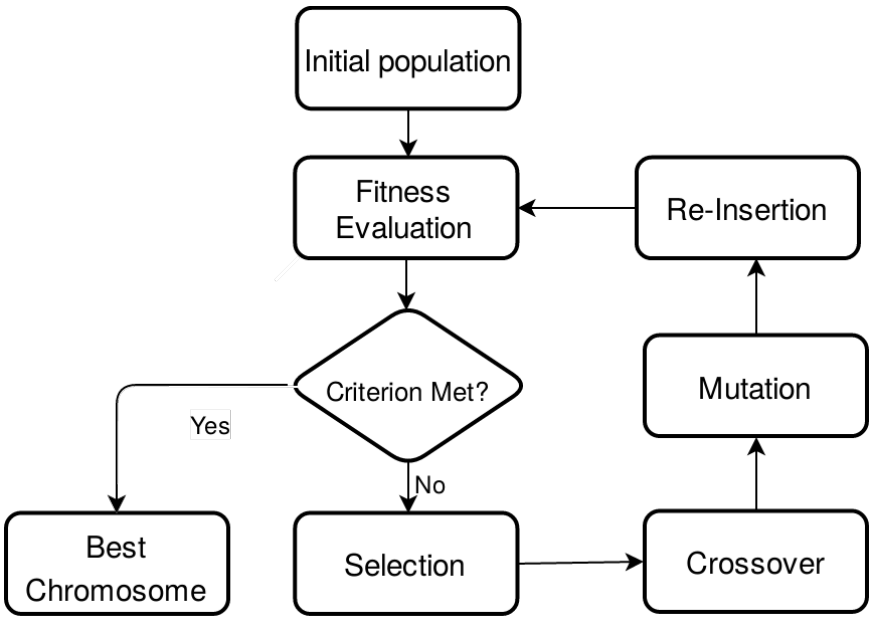

Figure 5.5 The generic diagram of evolutionary algorithm as applied in our study.

5.4.1 Experimental setup

In our experiments, we have used three widely well-known benchmarks in TC: Reuters-21578 Benchmark Corpus[2], Oshumed Benchmark Corpus[2], and the 4 Universities dataset also called Webkb[3]. The Reuters-21578 dataset is one of the most used test collection for TC research. We use the well-known "ApteMod" split [85]. This split includes 10788 documents from the Reuters financial service, divided into a training set of 7769 documents and a test set of 3019 documents. The dataset is highly skewed, the smallest category contains only two documents, and the biggest contains 3964 documents. Documents in this dataset belong to one or more categories. This version of the dataset contains ninety categories, however, in our experiments, we report results only for the largest ten categories. Oshumed dataset is extracted from the Oshumed[1] collection compiled by William Hersh. It includes 13,929 medical abstracts (6,286 for training and 7,643 for testing) from the MeSH categories of the year 1991. Each document in this dataset belongs to one or more categories from 23 cardiovascular diseases categories. Webkb dataset

[2]http://disi.unitn.it/moschitti/corpora.htm
[3]http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/

contains WWW-pages collected from computer science departments of various universities in January 1997 by the World Wide Knowledge Base (Webkb) project of the CMU text learning group. In this experiment, we kept only the four largest categories ("student", "faculty", "course" and "project"), and we split it into three random folds where two folds are used for the training set and one fold for the test set.

For all three datasets considered in the experiments:

- A default list of stop words, punctuation and numbers are removed.
- Lowercase transformation is applied.
- Porter's stemming is performed.

Furthermore, for each experiment, a binary transformation is applied. That leads to multiple distinct single-label binary task, one for each label (see Section 5.2.1). Each task could be treated as an independent experiment with its own dataset.

As mentioned above, each dataset has been split into training and test subsets. Table 5.3 shows, for each dataset, the number of documents in the training and test subsets, the number of classes, the number of terms, the size of smallest category and the size of the largest category.

TWSs are evolved using the training subset (see Section 5.3.2). Finally, the test subset is used to evaluate the performance of the generated TWS. And finally, for each dataset, we report the f_1 measure (see Section 5.3.2).

In order to obtain more reliable results, we have performed 20 runs on each task. After having evaluated the generated TWSs, we report the performance average and standard deviation over the 20 runs. In addition, we report the maximum and minimum f_1 score obtained across the 20 runs (for each run, only the last generated TWS is taken into account).

Tables 5.4, 5.5 and 5.6 show the results obtained by the generated TWSs and the best baseline using linear SVM.

Table 5.7 shows the average classification performance of the generated TWSs on the test subset of the training data (Validation) and the performance on the test

Table 5.3 Statistics on the selected datasets used for our experiments (training/test).

	Reuters	Oshumed	Webkb
# documents	7769/3019	6286/7643	2803/1396
# classes	90	23	4
# terms	26000	30198	7890
smallest category	1/1	65/70	336/168
largest category	2877/1087	1799/2153	1097/544

data (Test). The goal of this experiment is to show us if further learning is possible or to warn us of eventual overfitting.

Table 5.8 shows the average classification performance of a random learned TWS for a single-label binary task on the complete dataset. These results are important, in order to know whether our GP-Based TWS has good generalization performances.

5.4.2 Results

First, a fast study of the Tables 5.4, 5.5 and 5.6 shows that the best baseline TWS is different for each binary task. Therefore, a multi-labeled task requires different TWSs for each category. Using different TWSs could lead to better results. However, the problem is to recognize the best TWS for a specific task. Finding the TWS by cross-validation does not obligatorily return the best TWS.

Regarding Reuters-21578, the generated TWSs and the baseline schemes have similar performances. However, on Oshumed and Webkb datasets, the GP-Based TWSs outperforms the best baseline schemes.

From Table 5.7 , we can see that the performance of generated TWSs on the test subset of the training data during the cross-validation (See Section 5.3.2) are very similar to the performance on the test data. In addition, the standard TWSs have different results. That is interesting as it suggests that there is no overfitting and that further learning can improve the performance.

Table 5.4 Classification performance on top 10 categories of Reuters-21578 obtained with the generated TWSs and the best baseline. We put in bold the best results between our generated TWSs and the best baseline of the standard TWSs.

	GP			Best Baseline	
Label	f1	Min	Max	f1	Baseline
earn	98.34±0.09	98.24	98.54	**98.38**	TF-IDF
acq	96.93±0.23	96.55	97.54	**97.10**	TF-IDF
money-fx	**79.60**±0.50	78.16	80.45	78.63	TF-IDF
grain	**94.25**±0.63	93.10	95.22	93.43	TF-RF
crude	**90.01**±0.81	88.27	90.94	88.24	TF-RF
trade	**79.10**±1.21	77.69	80.18	78.03	TF-RF
interest	75.16±0.50	74.45	76.19	**76.19**	TF-IDF
ship	**80.52**±1.54	77.84	82.93	78.95	TF-OR
wheat	88.11±1.26	86.12	90.96	**90.20**	TF-χ^2
corn	92.80±0.27	90.83	93.94	**93.91**	TF-χ^2
Average	**87.48**±0.70	86.13	88.69	87.30	

Table 5.5 Classification performance on Oshumed dataset obtained with the generated TWSs and the best baseline.

	GP			Best Baseline	
Label	f1	Min	Max	f1	Baseline
C01	**68.19**±1.00	65.91	70.71	64.36	TF-OR
C02	**41.28**±1.20	38.45	43.51	36.38	TF-OR
C03	76.54±3.28	72.03	81.21	**78.23**	TF-OR
C04	80.06±1.48	77.67	81.72	**80.06**	TF-χ^2
C05	**59.48**±0.20	59.05	60.59	52.85	TF-OR
C06	**73.99**±1.29	71.49	75.76	71.44	TF-OR
C07	**41.40**±3.35	34.86	47.45	32.6	TF-OR
C08	**63.97**±2.51	59.13	67.69	61.34	TF-OR
C09	**53.75**±2.63	50.85	58.43	48.00	TF-OR
C10	**57.00**±2.33	51.05	59.53	50.2	TF-RF
C11	**67.78**±1.06	65.52	69.23	66.67	TF-OR
C12	**76.72**±1.10	73.52	78.25	72.86	TF-OR
C13	**66.48**±0.47	64.72	67.92	63.70	TF-OR
C14	**80.08**±0.39	79.22	80.55	77.11	TF-IDF
C15	**65.98**±0.71	64.16	67.20	61.53	TF-χ^2
C16	**33.54**±0.89	31.14	35.41	28.00	TF-OR
C17	**64.85**±0.90	61.87	66.87	59.24	TF-χ^2
C18	61.21±1.50	57.50	65.12	**61.22**	TF-OR
C19	**41.60**±2.04	38.23	45.01	39.84	TF-OR
C20	**71.61**±0.28	70.96	72.07	69.62	TF-OR
C21	**65.55**±0.32	64.18	67.56	64.37	TF-χ^2
C22	**10.31**±0.12	8.33	14.37	4.21	TF-OR
C23	**46.77**±0.08	45.59	47.20	46.15	TF-IDF
Average	**59.48**±1.26	56.76	61.89	56.08	

Table 5.6 Classification performance on Webkb dataset obtained with the generated TWSs and the best baseline.

	GP			Best Baseline	
Label	f1	Min	Max	f1	Baseline
student	**90.29**±0.50	89.05	90.90	90.11	TF-RF
faculty	**86.62**±0.15	85.69	87.81	86.21	TF-RF
project	**80.82**±0.64	77.48	81.76	80.25	TF-RF
course	**94.47**±0.34	93.86	96.08	93.56	TF-RF
Average	**88.05**±0.41	86.52	89.14	87.53	

Table 5.7 Average classification performance for validation phase and test phase.

	Reuters	Oshumed	Webkb
Validation	**89.15**±0.42	**59.74**±0.9	**87.74**±0.31
Test	**87.48**±0.70	**59.48**±1.26	**88.05**±0.41

From Table 5.8, we can see that the average performance (macro-f_1) of the generated TWSs outperforms the best baseline on the three corpora which means that the three learned TWS have good generalization performance.

Finally, compared to the results obtained in [184] on Reuters-21578 and Webkb, we have similar results. Note that, in [184], they used Reuters-10 dataset which contains, only documents from the top 10 categories of the Reuters-21578 dataset, whereas we use Reuters-21578 “ModApte” split which contains documents from 90 categories.

5.5 Conclusion

In this chapter, we have studied the benefits of using GP for generating TWS for TC. Unlike previous studies, we generate formulas by combining statistical information at a microscopic level. This kind of generation is new, and we can conclude that :

- Different datasets require different formulas. This means that having a good generic scheme is hard to find.

Table 5.8 Average classification performance of random TWS learned for a single-label task on its corresponding dataset and the best baseline. The selected TWS is randomly chosen between the best generated TWSs for each category.

	GP-Based			Baseline	
dataset	Prefixed formula	TWS	f_1	f_1	Best
Reuters	$**C*//acN\log_2 cC$	$C*C*(\frac{a}{c*N}*\log(2+c))$	86.9	85.9	TF-RF
Oshumed	$/d/+N_t log 2 C_t a$	$\frac{a}{d*(N_t+log(2+C_t))}$	60.3	57.1	TF-χ^2
Webkb	$\log_1 \log_2 a$	$\log(1+log(2+a))$	88.4	87.5	TF-RF

- Within a corpus, it is even better to use different schemes for different categories. The hard task is to find out the best for each one.
- GP can find very good formulas which outperform standard formulas given by experts in the literature.
- Eventually, even if the generated formula is specific to a given category, results show that the best formula for one category is generic enough to be good (but not best) for other categories.

6

Conclusion

In this thesis, we deal with TC problems. The literature related to the TC techniques was considered in Chapter 2. Term-weighting was considered in more details in Chapter 3.

Three main contributions to the TC are presented in Chapter 4 and Chapter 5. First, we analyzed and compared the impact of eight different TWS on the classification model for the multi-label task. These schemes were used in conjunction with five classifiers tested on three well-known benchmarks (i.e., Reuters-21578, Oshumed, and 20newsgroups datasets). Results could be found in Table 4. We find that the superiority of supervised TWS over unsupervised TWS is not clear. The aim was to have a comparative study for future work. In addition, we are currently working on providing a complete survey on the impact of the different preprocessing techniques on the performance of classification models. Secondly, we introduced a new TWS based on Information Gain (*IG*) for the multi-label classification task. The basic idea is that the *IG* weight of a feature in negative categories should affect the importance of this term in the postive category. In the

proposed method, the distribution of terms are not only considered in terms of the positive and negative categories.

Finally, we studied the benefits of using GP for generating TWS for TC. Unlike previous studies, we generate formulas by combining statistical information at a microscopic level.

These works are promising. They can lead to new perspectives. It would be interesting to experiment bagging and boosting techniques on very large datasets to avoid local optima. It would also be possible to replace the GP by other tree construction algorithms such as Monte Carlo Tree Search (MCTS) [193; 194; 195; 196; 197; 198] or Nested Monte Carlo (NMC) [177].

Bibliography

[1] Andreas Hotho, Andreas Nürnberger, and Gerhard Paaß. A brief survey of text mining. Citeseer, 2005.

[2] Ian H Witten, Eibe Frank, Mark A Hall, and Christopher J Pal. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[3] Mehdi Allahyari, Seyedamin Pouriyeh, Mehdi Assefi, Saied Safaei, Elizabeth D Trippe, Juan B Gutierrez, and Krys Kochut. A brief survey of text mining: Classification, clustering and extraction techniques. *arXiv preprint arXiv:1707.02919*, 2017.

[4] Daniel Jurafsky. Speech and language processing: An introduction to natural language processing. *Computational linguistics, and speech recognition*, 2000.

[5] Franca Debole and Fabrizio Sebastiani. Supervised term weighting for automated text categorization. In *Text mining and its applications*, pages 81–97. Springer, 2004.

[6] Zhi-Hong Deng, Shi-Wei Tang, Dong-Qing Yang, Ming Zhang Li-Yu Li, and Kun-Qing Xie. A comparative study on feature weight in text categorization. In *Advanced Web Technologies and Applications*, pages 588–597. Springer, 2004.

[7] Hassan Saif, Miriam Fernandez, Yulan He, and Harith Alani. On stopwords, filtering and data sparsity for sentiment analysis of twitter. *Filtering and Data Sparsity for Sentiment Analysis of Twitter*, 2014.

[8] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

[9] Man Lan, Chew Lim Tan, Jian Su, and Yue Lu. Supervised and traditional term weighting methods for automatic text categorization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(4):721–735, 2009.

[10] Yiming Yang and Jan O Pedersen. A comparative study on feature selection in text categorization. In *ICML*, volume 97, pages 412–420, 1997.

[11] Alper Kursat Uysal and Serkan Gunal. The impact of preprocessing on text classification. *Information Processing & Management*, 50(1):104–112, 2014.

[12] Ravi Parikh and Matin Movassate. Sentiment analysis of user-generated twitter updates using various classification techniques. *CS224N Final Report*, 118, 2009.

[13] William B Cavnar, John M Trenkle, et al. N-gram-based text categorization. *Ann arbor mi*, 48113(2):161–175, 1994.

[14] Johannes Furnkranz. A study using n-gram features for text categorization. 1998.

[15] Marc Damashek. Gauging similarity with n-grams: Language-independent categorization of text. *Science*, 267(5199):843, 1995.

[16] Ralf Steinberger, Bruno Pouliquen, Mijail Kabadjov, and Erik Van der Goot. Jrc-names: A freely available, highly multilingual named entity resource. *arXiv preprint arXiv:1309.6162*, 2013.

[17] José Ramon Méndez, Eva Lorenzo Iglesias, Florentino Fdez-Riverola, Fernando Díaz, and Juan M Corchado. Tokenising, stemming and stopword removal on anti-spam filtering domain. In *Conference of the Spanish Association for Artificial Intelligence*, pages 449–458. Springer, 2005.

[18] Jan Pomikálek and Radim Rehurek. The influence of preprocessing parameters on text categorization. *International Journal of Applied Science, Engineering and Technology*, 1:430–434, 2007.

[19] Richard William Sproat. *Morphology and computation*. MIT press, 1992.

[20] Julie Beth Lovins. Development of a stemming algorithm. *Mech. Translat. & Comp. Linguistics*, 11(1-2):22–31, 1968.

[21] Gerard Salton and Michael E Lesk. The smart automatic document retrieval systems—an illustration. *Communications of the ACM*, 8(6):391–398, 1965.

[22] Anjali Ganesh Jivani et al. A comparative study of stemming algorithms. *Int. J. Comp. Tech. Appl*, 2(6):1930–1938, 2011.

[23] Cornelis J Van Rijsbergen, Stephen Edward Robertson, and Martin F Porter. *New models in probabilistic information retrieval*. British Library Research and Development Department London, 1980.

[24] Martin F Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[25] Karen Sparck Jones. *Readings in information retrieval*. Morgan Kaufmann, 1997.

[26] Martin F Porter. Snowball: A language for stemming algorithms. http://snowballstem.org/algorithms/porter/stemmer.html, 2001. [Online; accessed 22-June-2018].

[27] Martin F Porter, Richard Boulton, and Andrew Macfarlane. The english (porter2) stemming algorithm. http://snowballstem.org/algorithms/english/stemmer.html, 2002. [Online; accessed 22-June-2018].

[28] Chris D. Paice. Another stemmer. *SIGIR Forum*, 24(3):56–61, Nov 1990.

[29] Chris D Paice. An evaluation method for stemming algorithms. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 42–50. Springer-Verlag New York, Inc., 1994.

[30] John Dawson. Suffix removal and word conflation. *ALLC bulletin*, 2(3):33–46, 1974.

[31] Jiaul H Paik, Mandar Mitra, Swapan K Parui, and Kalervo Järvelin. Gras: An effective and efficient stemming algorithm for information retrieval. *ACM Transactions on Information Systems (TOIS)*, 29(4):19, 2011.

[32] Mohamad Ababneh, Riyad Al-Shalabi, Ghassan Kanaan, and Alaa Al-Nobani. Building an effective rule-based light stemmer for arabic language to improve search effectiveness. *International Arab Journal of Information Technology (IAJIT)*, 9(4), 2012.

[33] Haidar M Harmanani, Walid T Keirouz, and Saeed Raheel. A rule-based extensible stemmer for information retrieval with application to arabic. 2006.

[34] Ghassan Kanaan, Riyad Al-Shalabi, Mohamad Ababneh, and Alaa Al-Nobani. Building an effective rule-based light stemmer for arabic language to inprove search effectiveness. In *Innovations in Information Technology, 2008. IIT 2008. International Conference on*, pages 312–316. IEEE, 2008.

[35] Nikola Ljubešić, Damir Boras, and Ozren Kubelka. Retrieving information in croatian: Building a simple and efficient rule-based stemmer. *Digital information and heritage/Seljan, Sanja*, pages 313–320, 2007.

[36] Jacques Savoy. Stemming of french words based on grammatical categories. *Journal of the American Society for Information Science*, 44(1):1, 1993.

[37] Jacques Savoy. Light stemming approaches for the french, portuguese, german and hungarian languages. In *Proceedings of the 2006 ACM symposium on Applied computing*, pages 1031–1035. ACM, 2006.

[38] Vishal Gupta. Hindi rule based stemmer for nouns. *International Journal of Advanced Research in Computer Science and Software Engineering*, 4(1), 2014.

[39] Niraj Aswani and Robert J Gaizauskas. Developing morphological analysers for south asian languages: Experimenting with the hindi and gujarati languages. In *LREC*, pages 811–815, 2010.

[40] Jelita Asian, Hugh E Williams, and Seyed MM Tahaghoghi. Stemming indonesian. In *Proceedings of the Twenty-eighth Australasian conference on Computer Science-Volume 38*, pages 307–314. Australian Computer Society, Inc., 2005.

[41] Martin F Porter. Stemming algorithms for various european languages. http://snowball.tartarus.org/texts/stemmersoverview.html, 2002. [Online; accessed 22-June-2018].

[42] James Mayfield and Paul McNamee. Single n-gram stemming. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 415–416. ACM, 2003.

[43] Massimo Melucci and Nicola Orio. A novel method for stemmer generation based on hidden markov models. In *Proceedings of the twelfth international conference on Information and knowledge management*, pages 131–138. ACM, 2003.

[44] Prasenjit Majumder, Mandar Mitra, Swapan K Parui, Gobinda Kole, Pabitra Mitra, and Kalyankumar Datta. Yass: Yet another suffix stripper. *ACM transactions on information systems (TOIS)*, 25(4):18, 2007.

[45] Deepika Sharma. Stemming algorithms: a comparative study and their analysis. *International Journal of Applied Information Systems*, 4(3):7–12, 2012.

[46] William B Frakes and Christopher J Fox. Strength and similarity of affix removal stemming algorithms. In *ACM SIGIR Forum*, volume 37, pages 26–30. ACM, 2003.

[47] Sandeep R Sirsat, Vinay Chavan, and Hemant S Mahalle. Strength and accuracy analysis of affix removal stemming algorithms. *International Journal of Computer Science and Information Technologies*, 4(2):265–269, 2013.

[48] Maurice Gross. Lemmatization of compound tenses in english. *Lingvisticae Investigationes*, 22(1):71–122, 1998.

[49] Nizar Habash, Owen Rambow, and Ryan Roth. Mada+ tokan: A toolkit for arabic tokenization, diacritization, morphological disambiguation, pos tagging, stemming and lemmatization. In *Proceedings of the 2nd international conference on Arabic language resources and tools (MEDAR), Cairo, Egypt*, volume 41, page 62, 2009.

[50] Djamé Seddah, Grzegorz Chrupała, Özlem Çetinoğlu, Josef Van Genabith, and Marie Candito. Lemmatization and lexicalized statistical parsing of morphologically rich languages: the case of french. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 85–93. Association for Computational Linguistics, 2010.

[51] Derwin Suhartono. Lemmatization technique in bahasa: Indonesian. *Journal of Software*, 9(5):1203, 2014.

[52] Staffan Hellberg. Computerized lemmatization without the use of a dictionary: a case study from swedish lexicology. *Computers and the Humanities*, 6(4):209–212, 1972.

[53] Tuomo Korenius, Jorma Laurikkala, Kalervo Järvelin, and Martti Juhola. Stemming and lemmatization in the clustering of finnish text documents. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, page 625–633. ACM, 2004.

[54] Praharshana Perera and René Witte. A self-learning context-aware lemmatizer for german. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, 2005.

[55] Joël Plisson, Nada Lavrac, Dr Mladenić, et al. A rule based approach to word lemmatization. 2004.

[56] Pieter Theron and Ian Cloete. Automatic acquisition of two-level morphological rules. page 103–110. Association for Computational Linguistics, 1997.

[57] Bart Jongejan and Hercules Dalianis. Automatic training of lemmatization rules that handle morphological changes in pre-, in-and suffixes alike. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 145–153. Association for Computational Linguistics, 2009.

[58] Adwait Ratnaparkhi. A simple introduction to maximum entropy models for natural language processing. *IRCS Technical Reports Series*, page 81, 1997.

[59] Hans Peter Luhn. Key word-in-context index for technical literature (kwic index). *Journal of the Association for Information Science and Technology*, 11(4):288–295, 1960.

[60] W. John Wilbur and Karl Sirotkin. The automatic identification of stop words. *J. Inf. Sci.*, 18(1):45–55, January 1992.

[61] Yiming Yang and John Wilbur. Using corpus statistics to remove redundant words in text categorization. *J. Am. Soc. Inf. Sci.*, 47(5):357–369, May 1996.

[62] Catarina Silva and Bemardete Ribeiro. The importance of stop word removal on recall values in text categorization. In *Neural Networks, 2003. Proceedings of the International Joint Conference on*, volume 3, pages 1661–1666. IEEE, 2003.

[63] Masoud Makrehchi and Mohamed S. Kamel. Automatic extraction of domain-specific stopwords from labeled documents. In *Proceedings of the IR Research, 30th European Conference on Advances in Information Retrieval*, ECIR '08, pages 222–233, Berlin, Heidelberg, 2008. Springer-Verlag.

[64] CJ Van Rijsbergen. Information retrieval. dept. of computer science, university of glasgow. *URL: citeseer. ist. psu. edu/vanrijsbergen79information. html*, 14, 1979.

[65] Christopher Fox. Information retrieval data structures and algorithms. *Lexical Analysis and Stoplists*, pages 102–130, 1992.

[66] Rachel Tsz-Wai Lo, Ben He, and Iadh Ounis. Automatically building a stopword list for an information retrieval system. In *Journal on Digital Information Management: Special Issue on the 5th Dutch-Belgian Information Retrieval Workshop (DIR)*, volume 5, pages 17–24. Citeseer, 2005.

[67] George K Zipf. Human behavior and the principle of least effort, 1950.

[68] Jinxi Xu and W Bruce Croft. Query expansion using local and global document analysis. In *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 4–11. ACM, 1996.

[69] Thomas M Cover and Joy A Thomas. Entropy, relative entropy and mutual information. *Elements of information theory*, 2:1–55, 1991.

[70] S Vijayarani, Ms J Ilamathi, and Ms Nithya. Preprocessing techniques for text mining-an overview. *International Journal of Computer Science & Communication Networks*, 5(1):7–16, 2015.

[71] Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330, 1993.

[72] W. Nelson Francis. A tagged corpus – problems and prospects. In Sidney Greenbaum, Geoffrey Leech, and Jan Svartvik, editors, *Studies in English linguistics for Randolph Quirk*, pages 192–209. Longman, London and New York, 1979.

[73] Roger Garside, Geoffrey N Leech, and Tony McEnery. *Corpus annotation: linguistic information from computer text corpora*. Taylor & Francis, 1997.

[74] Sonam Tripathi and Tripti Sharma. Document classification using part of speech in text mining. *International Journal of Science and Research (IJSR)*, 4(12):2004–2008, Dec 2015.

[75] Stephanie Chua. The role of parts-of-speech in feature selection. *Hong Kong*, page 5, 2008.

[76] Andrew McCallum and Wei Li. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. volume 4, page 188–191. Association for Computational Linguistics, 2003.

[77] Burr Settles. Biomedical named entity recognition using conditional random fields and rich feature sets. page 104. Association for Computational Linguistics, 2004.

[78] Jun'ichi Kazama, Takaki Makino, Yoshihiro Ohta, and Jun'ichi Tsujii. Tuning support vector machines for biomedical named entity recognition. In *Proceedings of the ACL-02 workshop on Natural language processing in the biomedical domain-Volume 3*, page 1–8. Association for Computational Linguistics, 2002.

[79] Hideki Isozaki and Hideto Kazawa. Efficient support vector classifiers for named entity recognition. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, page 1–7. Association for Computational Linguistics, 2002.

[80] Daniel M. Bikel, Scott Miller, Richard Schwartz, and Ralph Weischedel. Nymble: A high-performance learning name-finder. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, ANLC '97, page 194–201. Association for Computational Linguistics, 1997.

[81] Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence*, 165(1):91–134, Jun 2005.

[82] David Nadeau, Peter D. Turney, and Stan Matwin. Unsupervised named-entity recognition: Generating gazetteers and resolving ambiguity. In *Advances in Artificial Intelligence*, Lecture Notes in Computer Science, page 266–277. Springer, Berlin, Heidelberg, Jun 2006.

[83] Bin Liu and Chunping Li. An efficient feature selection method using named entity recognition for chinese text categorization. In *2009 International Conference on Machine Learning and Cybernetics*, volume 6, page 3527–3531, Jul 2009.

[84] Harper Collins. For the american heritage dictionary definition. *natural language. (n.d.) American Heritage® Dictionary of the English Language, Fifth Edition. (2011). Retrieved June 12 2018 from https://www.thefreedictionary.com/natural+language*, 2011.

[85] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer, 1998.

[86] Deqing Wang and Hui Zhang. Inverse category frequency based supervised term weighting scheme for text categorization. *preprint arXiv:1012.2609v4*, 2013.

[87] Ahmad Mazyad, Fabien Teytaud, and Cyril Fonlupt. A comparative study on term weighting schemes for text classification. In *International Workshop on Machine Learning, Optimization, and Big Data*, pages 100–108. Springer, 2017.

[88] Ahmad Mazyad, Fabien Teytaud, and Cyril Fonlupt. Information gain based term weighting method for multi-label text classification task. In *IntelliSys 2018*, 2018.

[89] Vlado Kešelj, Fuchun Peng, Nick Cercone, and Calvin Thomas. N-gram-based author profiles for authorship attribution. In *Proceedings of the conference pacific association for computational linguistics, PACLING*, volume 3, pages 255–264, 2003.

[90] Fuchun Peng and Dale Schuurmans. Combining naive bayes and n-gram language models for text classification. In FabrizioEditor Sebastiani, editor, *Advances in Information Retrieval*, Lecture Notes in Computer Science, page 335–350. Springer Berlin Heidelberg, 2003.

[91] George Forman. An extensive empirical study of feature selection metrics for text classification. *Journal of machine learning research*, 3(Mar):1289–1305, 2003.

[92] Kenji Kira and Larry A Rendell. The feature selection problem: Traditional methods and a new algorithm. In *Aaai*, volume 2, pages 129–134, 1992.

[93] Mark A. Hall. *Correlation-based feature selection of discrete and numeric class machine learning*. May 2000.

[94] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3(Mar):1157–1182, 2003.

[95] Claude Elwood Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.

[96] N. Kwak and Chong-Ho Choi. Input feature selection by mutual information based on parzen window. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(12):1667–1671, Dec 2002.

[97] Zhaohui Zheng, Xiaoyun Wu, and Rohini Srihari. Feature selection for text categorization on imbalanced data. *ACM SIGKDD Explorations Newsletter*, 6(1):80, Jun 2004.

[98] François Fleuret. Fast binary feature selection with conditional mutual information. *Journal of Machine Learning Research*, 5(Nov):1531–1555, 2004.

[99] Hanchuan Peng, Fuhui Long, and C. Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1226–1238, Aug 2005.

[100] P. A. Estevez, M. Tesmer, C. A. Perez, and J. M. Zurada. Normalized mutual information feature selection. *IEEE Transactions on Neural Networks*, 20(2):189–201, Feb 2009.

[101] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[102] Karl Pearson. X. on the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 50(302):157–175, 1900.

[103] Karl Pearson. *On the Criterion that a Given System of Deviations from the Probable in the Case of a Correlated System of Variables is Such that it Can be Reasonably Supposed to have Arisen from Random Sampling*, page 11–28. Springer Series in Statistics. Springer, New York, NY, 1992.

[104] C. Spearman. The proof and measurement of association between two things. *The American Journal of Psychology*, 15(1):72, Jan 1904.

[105] Yvan Saeys, Thomas Abeel, and Yves Van de Peer. *Robust Feature Selection Using Ensemble Feature Selection Techniques*, volume 5212, page 313–325. Springer Berlin Heidelberg, 2008.

[106] M. G. Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.

[107] Xiubo Geng, Tie-Yan Liu, Tao Qin, and Hang Li. Feature selection for ranking. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, page 407–414. ACM, 2007.

[108] Cosmin Lazar, Jonatan Taminau, Stijn Meganck, David Steenhoff, Alain Coletta, Colin Molter, Virginie de Schaetzen, Robin Duque, Hugues Bersini, and Ann Nowe. A survey on filter techniques for feature selection in gene expression microarray analysis. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 9(4):1106–1119, Jul 2012.

[109] Quanquan Gu, Zhenhui Li, and Jiawei Han. Generalized fisher score for feature selection. page 8.

[110] B. L. Welch. The generalization of 'student's' problem when several different population variances are involved. *Biometrika*, 34(1/2):28–35, 1947.

[111] Yudong Zhang, Zhengchao Dong, Preetha Phillips, Shuihua Wang, Genlin Ji, Jiquan Yang, and Ti-Fei Yuan. Detection of subjects and brain regions related to alzheimer's disease using 3d mri scans based on eigenbrain and machine learning. *Frontiers in Computational Neuroscience*, 9, Jun 2015.

[112] Eric W Weisstein. Eigenvector. 2002.

[113] G. Roffo, S. Melzi, and M. Cristani. Infinite feature selection. In *2015 IEEE International Conference on Computer Vision (ICCV)*, page 4202–4210, Dec 2015.

[114] T. M. Phuong, Z. Lin, and R. B. Altman. Choosing snps using feature selection. In *2005 IEEE Computational Systems Bioinformatics Conference (CSB'05)*, page 301–309, Aug 2005.

[115] Li-Ping Jing, Hou-Kuan Huang, and Hong-Bo Shi. Improved feature selection approach tfidf in text mining. In *Machine Learning and Cybernetics, 2002. Proceedings. 2002 International Conference on*, volume 2, pages 944–946. IEEE, 2002.

[116] Bong Chih How and Kulathuramaiyer Narayanan. An empirical study of feature selection for text categorization based on term weightage. In *Web Intelligence, 2004. WI 2004. Proceedings. IEEE/WIC/ACM International Conference on*, pages 599–602. IEEE, 2004.

[117] Tim O'Keefe and Irena Koprinska. Feature selection and weighting methods in sentiment analysis. In *Proceedings of the 14th Australasian document computing symposium, Sydney*, pages 67–74. Citeseer, 2009.

[118] Dunja Mladeni'c and Marko Grobelnik. Feature selection for classification based on text hierarchy. In *Text and the Web, Conference on Automated Learning and Discovery CONALD-98*. Citeseer, 1998.

[119] David Broadhurst, Royston Goodacre, Alun Jones, Jem J Rowland, and Douglas B Kell. Genetic algorithms as a method for variable selection in multiple linear regression and partial least squares regression, with applications to pyrolysis mass spectrometry. *Analytica Chimica Acta*, 348(1-3):71–86, 1997.

[120] Sihua Peng, Qianghua Xu, Xuefeng Bruce Ling, Xiaoning Peng, Wei Du, and Liangbiao Chen. Molecular classification of cancer types from microarray data using the combination of genetic algorithms and support vector machines. *FEBS Letters*, 555(2):358–362, 2003.

[121] Shital C. Shah and Andrew Kusiak. Data mining and genetic algorithm based gene/snp selection. *Artificial Intelligence in Medicine*, 31(3):183–196, Jul 2004.

[122] L. Vermeulen-Jourdan, C. Dhaenens, and E-G. Talbi. Linkage disequilibrium study with a parallel adaptive ga. *International Journal of Foundations of Computer Science*, 16(02):241–260, Apr 2005.

[123] Thanyaluk Jirapech-Umpai and Stuart Aitken. Feature selection and classification for microarray data analysis: Evolutionary methods for identifying predictive genes. *BMC Bioinformatics*, 6:148, Jun 2005.

[124] Durga Prasad Muni, Nikhil R Pal, and Jyotirmay Das. Genetic programming for simultaneous feature selection and classifier design. 2006.

[125] Chris Hans, Adrian Dobra, and Mike West. Shotgun stochastic search for "large p" regression. *Journal of the American Statistical Association*, 102(478):507–516, Jun 2007.

[126] Béatrice Duval, Jin-Kao Hao, and Jose Crispin Hernandez Hernandez. A memetic algorithm for gene selection and molecular classification of cancer. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*, GECCO '09, page 201–208. ACM, 2009.

[127] P. Xuan, M.Z. Guo, J. Wang, C.Y. Wang, X.Y. Liu, and Y. Liu. Genetic algorithm-based efficient feature selection for classification of pre-mirnas. *Genetics and Molecular Research*, 10(2):588–603, 2011.

[128] Rahul Karthik Sivagaminathan and Sreeram Ramakrishnan. A hybrid approach for feature subset selection using neural networks and ant colony optimization. *Expert systems with applications*, 33(1):49–60, 2007.

[129] Mehdi Hosseinzadeh Aghdam, Nasser Ghasem-Aghaee, and Mohammad Ehsan Basiri. Text feature selection using ant colony optimization. *Expert systems with applications*, 36(3):6843–6853, 2009.

[130] Yumin Chen, Duoqian Miao, and Ruizhi Wang. A rough set approach to feature selection based on ant colony optimization. *Pattern Recognition Letters*, 31(3):226–233, 2010.

[131] David B Skalak. Prototype and feature selection by sampling and random mutation hill climbing algorithms. In *Machine Learning Proceedings 1994*, pages 293–301. Elsevier, 1994.

[132] Ron Kohavi and George H John. Wrappers for feature subset selection. *Artificial intelligence*, 97(1-2):273–324, 1997.

[133] N Long, D Gianola, GJM Rosa, and KA Weigel. Dimension reduction and variable selection for genomic selection: application to predicting milk yield in holsteins. *Journal of Animal Breeding and Genetics*, 128(4):247–257, 2011.

[134] E. Alba, J. Garcia-Nieto, L. Jourdan, and E. G. Talbi. Gene selection in cancer classification using pso/svm and ga/svm hybrid algorithms. In *2007 IEEE Congress on Evolutionary Computation*, page 284–290, Sep 2007.

[135] Li-Yeh Chuang, Cheng-Huei Yang, and Cheng-Hong Yang. Tabu search and binary particle swarm optimization for feature selection using microarray data. *Journal of Computational Biology*, 16(12):1689–1703, Dec 2009.

[136] RM Sharkawy, K Ibrahim, MMA Salama, and R Bartnikas. Particle swarm optimization feature selection for the classification of conducting particles in transformer oil. *IEEE Transactions on Dielectrics and Electrical Insulation*, 18(6), 2011.

[137] Norshafarina Omar, Mohd Shahizan bin Othman, et al. Particle swarm optimization feature selection for classification of survival analysis in cancer. *International Journal of Innovative Computing*, 2(1), 2013.

[138] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, Mar 1986.

[139] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[140] Jerome H Friedman, Richard A Olshen, Charles J Stone, et al. Classification and regression trees. *Belmont, CA: Wadsworth & Brooks*, 1984.

[141] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[142] Lorenzo Rosasco, Ernesto De Vito, Andrea Caponnetto, Michele Piana, and Alessandro Verri. Are loss functions all the same? *Neural Computation*, 16(5):1063–1076, May 2004.

[143] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, Aug 1996.

[144] Scott Fortmann-Roe. Understanding the bias-variance tradeoff. 2012.

[145] Zhi-Hua Zhou. *Ensemble methods: foundations and algorithms*. Chapman and Hall/CRC, 2012.

[146] Tin Kam Ho. Random decision forests. In *Document analysis and recognition, 1995., proceedings of the third international conference on*, volume 1, pages 278–282. IEEE, 1995.

[147] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct 2001.

[148] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7(Mar):551–585, 2006.

[149] Tong Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *ICML 2004: PROCEEDINGS OF THE TWENTY-FIRST INTERNATIONAL CONFERENCE ON MACHINE LEARNING. OMNIPRESS*, pages 919–926, 2004.

[150] Margaret Sullivan Pepe. *The statistical evaluation of medical tests for classification and prediction*. Medicine, 2003.

[151] Peter A Flach. The geometry of roc space: understanding machine learning metrics through roc isometrics. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 194–201, 2003.

[152] David J. Hand. Measuring classifier performance: a coherent alternative to the area under the roc curve. *Machine Learning*, 77(1):103–123, Oct 2009.

[153] George Hripcsak and Adam S Rothschild. Agreement, the f-measure, and reliability in information retrieval. *Journal of the American Medical Informatics Association*, 12(3):296–298, 2005.

[154] Alaa M El-Halees. Arabic text classification using maximum entropy. *IUG Journal of Natural Studies*, 15(1), 2015.

[155] B. Tang, H. He, P. M. Baggenstoss, and S. Kay. A bayesian classification approach using class-specific features for text categorization. *IEEE Transactions on Knowledge and Data Engineering*, 28(6):1602–1606, Jun 2016.

[156] Pascal Soucy and Guy W. Mineau. Beyond tfidf weighting for text categorization in the vector space model. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, IJCAI'05, page 1130–1135. Morgan Kaufmann Publishers Inc., 2005.

[157] D.S. Guru and Mahamad Suhil. A novel term-class relevance measure for text categorization. *Procedia Computer Science*, 45:13–22, 2015.

[158] D. Isa, L.H. Lee, V.P. Kallimani, and R. RajKumar. Text document preprocessing with the bayes formula for classification using the support vector machine. *IEEE Transactions on Knowledge and Data Engineering*, 20:1264–1272, Sep 2008.

[159] Eui-Hong Sam Han, George Karypis, and Vipin Kumar. Text categorization using weight adjusted k-nearest neighbor classification. In *Pacific-asia conference on knowledge discovery and data mining*, page 53–65. Springer, 2001.

[160] Hans Peter Luhn. *Potentialities of auto-encoding of scientific literature*. International Business Machines Corporation, Research Center, 1959.

[161] Gerard. Salton. *Automatic Information Organization and Retrieval.* McGraw Hill Text, 1968.

[162] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21, 1972.

[163] Gerard Salton, Edward A Fox, and Harry Wu. Extended boolean information retrieval. *Communications of the ACM*, 26(11):1022–1036, 1983.

[164] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47, 2002.

[165] Ronen Feldman and James Sanger. *The text mining handbook: advanced approaches in analyzing unstructured data*. Cambridge university press, 2007.

[166] Man Lan, Chew-Lim Tan, Hwee-Boon Low, and Sam-Yuan Sung. A comprehensive comparative study on term weighting schemes for text categorization with support vector machines. In *Special interest tracks and posters of the 14th international conference on World Wide Web*, pages 1032–1033. ACM, 2005.

[167] Edwin B. Wilson. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209–212, 1927.

[168] Vikas Sindhwani, Pushpak Bhattacharya, and Subrata Rakshit. Information theoretic feature crediting in multiclass support vector machines. In *Proceedings of the 2001 SIAM International Conference on Data Mining*, page 1–18. SIAM, 2001.

[169] Robert Tibshirani, Trevor Hastie, Balasubramanian Narasimhan, and Gilbert Chu. Diagnosis of multiple cancer types by shrunken centroids of gene expression. *Proceedings of the National Academy of Sciences*, 99(10):6567–6572, 2002.

[170] Ahmad Mazyad, Fabien Teytaud, and Cyril Fonlupt. Learning new term weighting schemes with genetic programming. In *The Biennial International Conference on Artificial Evolution*, 2017.

[171] Ahmad Mazyad, Fabien Teytaud, and Cyril Fonlupt. Generating term weighting schemes through genetic programming. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 268–269. ACM, 2018.

[172] Ahmad Mazyad, Fabien Teytaud, and Cyril Fonlupt. Generating term weighting schemes through genetic programming. In *The Fourth International Conference on Machine Learning, Optimization, and Data Science*. Springer, 2018.

[173] Konstantin Tretyakov. Machine learning techniques in spam filtering. In *Data Mining Problem-oriented Seminar, MTAT*, volume 3, pages 60–79, 2004.

[174] Marc A Zissman. Comparison of four approaches to automatic language identification of telephone speech. *IEEE Transactions on speech and audio processing*, 4(1):31, 1996.

[175] Efstathios Stamatatos. A survey of modern authorship attribution methods. *Journal of the Association for Information Science and Technology*, 60(3):538–556, 2009.

[176] Akshi Kumar and Teeja Mary Sebastian. Sentiment analysis on twitter. *IJCSI International Journal of Computer Science Issues*, 9(3):372–378, 2012.

[177] Tristan Cazenave. Nested monte-carlo expression discovery. In *ECAI*, pages 1057–1058, 2010.

[178] Ronan Cummins and Colm O'riordan. Evolving general term-weighting schemes for information retrieval: Tests on larger collections. *Artificial Intelligence Review*, 24(3-4):277–299, 2005.

[179] Ronan Cummins and Colm O'Riordan. Evolving local and global weighting schemes in information retrieval. *Information Retrieval*, 9(3):311–330, 2006.

[180] Ronan Cummins and Colm O'Riordan. Evolved term-weighting schemes in information retrieval: an analysis of the solution space. *Artificial Intelligence Review*, 26(1-2):35–47, 2006.

[181] Andrew Trotman. Learning to rank. *Information Retrieval*, 8(3):359–381, 2005.

[182] Nir Oren. Reexamining tf. idf based information retrieval with genetic programming. In *Proceedings of the 2002 annual research conference of the South African institute of computer scientists and information technologists on Enablement through technology*, pages 224–234. South African Institute for Computer Scientists and Information Technologists, 2002.

[183] Weiguo Fan, Edward A Fox, Praveen Pathak, and Harris Wu. The effects of fitness functions on genetic programming-based ranking discovery for web search. *Journal of the Association for Information Science and Technology*, 55(7):628–636, 2004.

[184] Hugo Jair Escalante, Mauricio A García-Limón, Alicia Morales-Reyes, Mario Graff, Manuel Montes-y Gómez, Eduardo F Morales, and José Martínez-Carranza. Term-weighting learning via genetic programming for text classification. *Knowledge-Based Systems*, 83:176–189, 2015.

[185] Nichael Lynn Cramer. A representation for the adaptive generation of simple sequential programs. In *Proceedings of the First International Conference on Genetic Algorithms*, pages 183–187, 1985.

[186] John R Koza. *Genetic programming: on the programming of computers by means of natural selection*, volume 1. MIT press, 1992.

[187] Dominic P Searson, David E Leahy, and Mark J Willis. Gptips: an open source genetic programming toolbox for multigene symbolic regression. In *Proceedings of the International multiconference of engineers and computer scientists*, volume 1, pages 77–80. Citeseer, 2010.

[188] John R Koza. Concept formation and decision tree induction using the genetic programming paradigm. In *International Conference on Parallel Problem Solving from Nature*, pages 124–128. Springer, 1990.

[189] John R Koza. *Genetic Programming II, Automatic Discovery of Reusable Subprograms*. MIT Press, Cambridge, MA, 1992.

[190] M Anthony Lewis, Andrew H Fagg, and Alan Solidum. Genetic programming approach to the construction of a neural network for control of a walking robot. In *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*, pages 2618–2623. IEEE, 1992.

[191] Murat Karakus. Function identification for the intrinsic strength and elastic properties of granitic rocks via genetic programming (gp). *Computers & geosciences*, 37(9):1318–1323, 2011.

[192] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *Journal of machine learning research*, 9(Aug):1871–1874, 2008.

[193] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

[194] GMJB Chaslot, Jahn-Takeshi Saito, Bruno Bouzy, JWHM Uiterwijk, and H Jaap Van Den Herik. Monte-carlo strategies for computer go. In *Proceedings of the 18th BeNeLux Conference on Artificial Intelligence, Namur, Belgium*, pages 83–91, 2006.

[195] Ahmad Mazyad, Fabien Teytaud, and Cyril Fonlupt. Monte-carlo tree search for the "mr jack" board game. *International Journal on Soft Computing, Artificial Intelligence and Applications (IJSCAI)*, 2015.

[196] Joris Duguépéroux, Ahmad Mazyad, Fabien Teytaud, and Julien Dehos. Pruning playouts in monte-carlo tree search for the game of havannah. In *International Conference on Computers and Games*, pages 47–57. Springer, 2016.

[197] Rémi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *International conference on computers and games*, pages 72–83. Springer, 2006.

[198] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer, 2006.