information

MDPI

Review

Text Classification Algorithms: A Survey

Kamran Kowsari [1,2,*], Kiana Jafari Meimandi [1], Mojtaba Heidarysafa [1], Sanjana Mendu [1], Laura Barnes [1,2,3] and Donald Brown [1,3]

[1] Department of Systems and Information Engineering, University of Virginia, Charlottesville, VA 22904, USA; kj6vd@virginia.edu (K.J.M.); mh4pk@virginia.edu (M.H.); sm7gc@virginia.edu (S.M.); lb3dp@virginia.edu (L.B.); deb@virginia.edu (D.B.)

[2] Sensing Systems for Health Lab, University of Virginia, Charlottesville, VA 22911, USA

[3] School of Data Science, University of Virginia, Charlottesville, VA 22904, USA

* Correspondence: kk7nc@virginia.edu; Tel.: +1-202-812-3013

Received: 22 March 2019; Accepted: 17 April 2019; Published: 23 April 2019

Abstract: In recent years, there has been an exponential growth in the number of complex documents and texts that require a deeper understanding of machine learning methods to be able to accurately classify texts in many applications. Many machine learning approaches have achieved surpassing results in natural language processing. The success of these learning algorithms relies on their capacity to understand complex models and non-linear relationships within data. However, finding suitable structures, architectures, and techniques for text classification is a challenge for researchers. In this paper, a brief overview of text classification algorithms is discussed. This overview covers different text feature extractions, dimensionality reduction methods, existing algorithms and techniques, and evaluations methods. Finally, the limitations of each technique and their application in real-world problems are discussed.

Keywords: text classification; text mining; text representation; text categorization; text analysis; document classification

1. Introduction

Text classification problems have been widely studied and addressed in many real applications [1–8] over the last few decades. Especially with recent breakthroughs in Natural Language Processing (NLP) and text mining, many researchers are now interested in developing applications that leverage text classification methods. Most text classification and document categorization systems can be deconstructed into the following four phases: Feature extraction, dimension reductions, classifier selection, and evaluations. In this paper, we discuss the structure and technical implementations of text classification systems in terms of the pipeline illustrated in Figure 1 (The source code and the results are shared as free tools at https://github.com/kk7nc/Text_Classification).

The initial pipeline input consists of some raw text data set. In general, text data sets contain sequences of text in documents as $D = \{X_1, X_2, \ldots, X_N\}$ where X_i refers to a data point (i.e., document, text segment) with s number of sentences such that each sentence includes w_s words with l_w letters. Each point is labeled with a class value from a set of k different discrete value indices [7].

Then, we should create a structured set for our training purposes which call this section Feature Extraction. The dimensionality reduction step is an optional part of the pipeline which could be part of the classification system (e.g., if we use Term Frequency-Inverse Document Frequency (TF-IDF) as our feature extraction and in train set we have 200k unique words, computational time is very expensive, so we could reduce this option by bringing feature space in other dimensional space). The most significant step in document categorization is choosing the best classification algorithm. The other part of the pipeline is the evaluation step which is divided into two parts (prediction the test set and

evaluating the model). In general, the text classification system contains four different levels of scope that can be applied:

1. **Document level:** In the document level, the algorithm obtains the relevant categories of a full document.
2. **Paragraph level:** In the paragraph level, the algorithm obtains the relevant categories of a single paragraph (a portion of a document).
3. **Sentence level:** In the sentence level, obtains the relevant categories of a single sentence (a portion of a paragraph).
4. **Sub-sentence level:** In the sub-sentence level, the algorithm obtains the relevant categories of sub-expressions within a sentence (a portion of a sentence)).

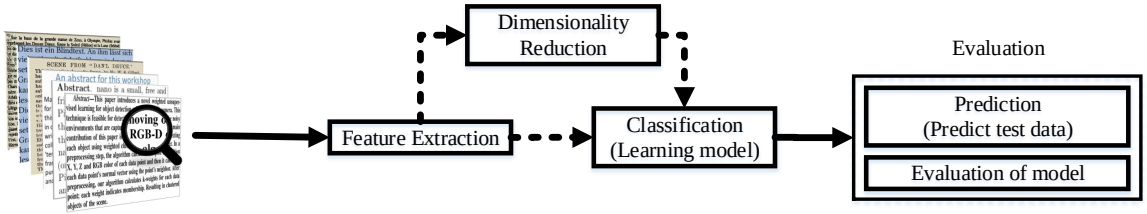

Figure 1. Overview of text classification pipeline.

(I) **Feature Extraction:** In general, texts and documents are unstructured data sets. However, these unstructured text sequences must be converted into a structured feature space when using mathematical modeling as part of a classifier. First, the data needs to be cleaned to omit unnecessary characters and words. After the data has been cleaned, formal feature extraction methods can be applied. The common techniques of feature extractions are Term Frequency-Inverse Document Frequency (TF-IDF), Term Frequency (TF) [9], Word2Vec [10], and Global Vectors for Word Representation (GloVe) [11]. In Section 2, we categorize these methods as either word embedding or weighted word techniques and discuss the technical implementation details.

(II) **Dimensionality Reduction:** As text or document data sets often contain many unique words, data pre-processing steps can be lagged by high time and memory complexity. A common solution to this problem is simply using inexpensive algorithms. However, in some data sets, these kinds of cheap algorithms do not perform as well as expected. In order to avoid the decrease in performance, many researchers prefer to use dimensionality reduction to reduce the time and memory complexity for their applications. Using dimensionality reduction for pre-processing could be more efficient than developing inexpensive classifiers.

In Section 3, we outline the most common techniques of dimensionality reduction, including Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), and non-negative matrix factorization (NMF). We also discuss novel techniques for unsupervised feature extraction dimensionality reduction, such as random projection, autoencoders, and t-distributed stochastic neighbor embedding (t-SNE).

(III) **Classification Techniques:** The most important step of the text classification pipeline is choosing the best classifier. Without a complete conceptual understanding of each algorithm, we cannot effectively determine the most efficient model for a text classification application. In Section 4, we discuss the most popular techniques of text classification. First, we cover traditional methods of text classification, such as Rocchio classification. Next, we talk about ensemble-based learning techniques such as boosting and bagging, which have been used mainly for query learning strategies and text analysis [12–14]. One of the simplest classification algorithms is logistic regression (LR) which has been addressed in most data mining domains [15–18]. In the earliest history of information retrieval as

a feasible application, The Naïve Bayes Classifier (NBC) was very popular. We have a brief overview of Naïve Bayes Classifier which is computationally inexpensive and also needs a very low amount of memory [19].

Non-parametric techniques have been studied and used as classification tasks such as k-nearest neighbor (KNN) [20]. Support Vector Machine (SVM) [21,22] is another popular technique which employs a discriminative classifier for document categorization. This technique can also be used in all domains of data mining such as bioinformatics, image, video, human activity classification, safety and security, etc. This model is also used as a baseline for many researchers to compare against their own works to highlight novelty and contributions.

Tree-based classifiers such as decision tree and random forest have also been studied with respect to document categorization [23]. Each tree-based algorithm will be covered in a separate sub-section. In recent years, graphical classifications have been considered [24] as a classification task such as conditional random fields (CRFs). However, these techniques are mostly used for document summarization [25] and automatic keyword extraction [26].

Lately, deep learning approaches have achieved surpassing results in comparison to previous machine learning algorithms on tasks such as image classification, natural language processing, face recognition, etc. The success of these deep learning algorithms relies on their capacity to model complex and non-linear relationships within data [27].

(IV) **Evaluation:** The final part of the text classification pipeline is evaluation. Understanding how a model performs is essential to the use and development of text classification methods. There are many methods available for evaluating supervised techniques. Accuracy calculation is the simplest method of evaluation but does not work for unbalanced data sets [28]. In Section 5, we outline the following evaluation methods for text classification algorithms: F_β Score [29], Matthews Correlation Coefficient (MCC) [30], receiver operating characteristics (ROC) [31], and area under the ROC curve (AUC) [32].

In Section 6, we talk about the limitations and drawbacks of the methods mentioned above. We also briefly compare the steps of pipeline including feature extractions, dimensionality reduction, classification techniques, and evaluation methods. The state-of-the-art techniques are compared in this section by many criteria such as architecture of their model, novelty of the work, feature extraction technique, corpus (the data set/s used), validation measure, and limitation of each work. Finding the best system for an application requires choosing a feature extraction method. This choice completely depends on the goal and data set of an application as some feature extraction techniques are not efficient for a specific application. For example, since GloVe is trained on Wikipedia and when used for short text messages like short message service (SMS), this technique does not perform as well as TF-IDF. Additionally, limited data points that this model cannot be trained as well as other techniques due to the small amount of data. The next step or in this pipeline, is a classification technique, where we briefly talk about the limitation and drawbacks of each technique.

In Section 7, we describe the text and document classification applications. Text classification is a major challenge in many domains and fields for researchers. Information retrieval systems [33] and search engine [34,35] applications commonly make use of text classification methods. Extending from these applications, text classification could also be used for applications such as information filtering (e.g., email and text message spam filtering) [36]. Next, we talk about adoption of document categorization in public health [37] and human behavior [38]. Another area that has been helped by text classification is document organization and knowledge management. Finally, we will discuss recommender systems which are extensively used in marketing and advertising.

2. Text Preprocessing

Feature extraction and pre-processing are crucial steps for text classification applications. In this section, we introduce methods for cleaning text data sets, thus removing implicit noise and allowing

for informative featurization. Furthermore, we discuss two common methods of text feature extraction: Weighted word and word embedding techniques.

2.1. Text Cleaning and Pre-processing

Most text and document data sets contain many unnecessary words such as stopwords, misspelling, slang, etc. In many algorithms, especially statistical and probabilistic learning algorithms, noise and unnecessary features can have adverse effects on system performance. In this section, we briefly explain some techniques and methods for text cleaning and pre-processing text data sets.

2.1.1. Tokenization

Tokenization is a pre-processing method which breaks a stream of text into words, phrases, symbols, or other meaningful elements called tokens [39,40]. The main goal of this step is the investigation of the words in a sentence [40]. Both text classification and text mining require a parser which processes the tokenization of the documents, for example:
sentence [41]:

After sleeping for four hours, he decided to sleep for another four.

In this case, the tokens are as follows:

{ *"After" "sleeping" "for" "four" "hours" "he" "decided" "to" "sleep" "for" "another" "four"* }.

2.1.2. Stop Words

Text and document classification includes many words which do not contain important significance to be used in classification algorithms, such as {*"a", "about", "above", "across", "after", "afterwards", "again",*. . .}. The most common technique to deal with these words is to remove them from the texts and documents [42].

2.1.3. Capitalization

Text and document data points have a diversity of capitalization to form a sentence. Since documents consist of many sentences, diverse capitalization can be hugely problematic when classifying large documents. The most common approach for dealing with inconsistent capitalization is to reduce every letter to lower case. This technique projects all words in text and document into the same feature space, but it causes a significant problem for the interpretation of some words (e.g., "*US*" (United States of America) to "*us*" (pronoun)) [43]. Slang and abbreviation converters can help account for these exceptions [44].

2.1.4. Slang and Abbreviation

Slang and abbreviation are other forms of text anomalies that are handled in the pre-processing step. An abbreviation [45] is a shortened form of a word or phrase which contain mostly first letters form the words, such as SVM which stands for Support Vector Machine.

Slang is a subset of the language used in informal talk or text that has different meanings such as "lost the plot", which essentially means that they've gone mad [46]. A common method for dealing with these words is converting them into formal language [47].

2.1.5. Noise Removal

Most of the text and document data sets contain many unnecessary characters such as punctuation and special characters. Critical punctuation and special characters are important for human understanding of documents, but it can be detrimental for classification algorithms [48].

2.1.6. Spelling Correction

Spelling correction is an optional pre-processing step. Typos (short for typographical errors) are commonly present in texts and documents, especially in social media text data sets (e.g., Twitter). Many algorithms, techniques, and methods have addressed this problem in NLP [49]. Many techniques and methods are available for researchers including hashing-based and context-sensitive spelling correction techniques [50], as well as spelling correction using Trie and Damerau–Levenshtein distance bigram [51].

2.1.7. Stemming

In NLP, one word could appear in different forms (i.e., singular and plural noun form) while the semantic meaning of each form is the same [52]. One method for consolidating different forms of a word into the same feature space is stemming. Text stemming modifies words to obtain variant word forms using different linguistic processes such as affixation (addition of affixes) [53,54]. For example, the stem of the word "studying" is "study".

2.1.8. Lemmatization

Lemmatization is a NLP process that replaces the suffix of a word with a different one or removes the suffix of a word completely to get the basic word form (lemma) [54–56].

2.2. Syntactic Word Representation

Many researchers have worked on this text feature extraction technique to solve the loosing syntactic and semantic relation between words. Many researchers addressed novel techniques for solving this problem, but many of these techniques still have limitations. In [57], a model was introduced in which the usefulness of including syntactic and semantic knowledge in the text representation for the selection of sentences comes from technical genomic texts. The other solution for syntactic problem is using the n-gram technique for feature extraction.

2.2.1. N-Gram

The n-gram technique is a set of *n-word* which occurs "in that order" in a text set. This is not a representation of a text, but it could be used as a feature to represent a text.

BOW is a representation of a text using its words $(1\text{-}gram)$ which loses their order (syntactic). This model is very easy to obtain and the text can be represented through a vector, generally of a manageable size of the text. On the other hand, *n-gram* is a feature of BOW for a representation of a text using 1-*gram*. It is very common to use 2-*gram* and 3-*gram*. In this way, the text feature extracted could detect more information in comparison to 1-*gram*.

An Example of 2-*Gram*

After sleeping for four hours, he decided to sleep for another four.

In this case, the tokens are as follows:

{ *"After sleeping", "sleeping for", "for four", "four hours", "four he" "he decided", "decided to", "to sleep", "sleep for", "for another", "another four"* }.

An Example of 3-*Gram*

After sleeping for four hours, he decided to sleep for another four.

In this case, the tokens are as follows:

{ *"After sleeping for", "sleeping for four", "four hours he", " hours he decided", "he decided to", "to sleep for", "sleep for another", "for another four"* }.

2.2.2. Syntactic N-Gram

In [58], syntactic n-grams are discussed which is defined by paths in syntactic dependency or constituent trees rather than the linear structure of the text.

2.3. Weighted Words

The most basic form of weighted word feature extraction is TF, where each word is mapped to a number corresponding to the number of occurrences of that word in the whole corpora. Methods that extend the results of TF generally use word frequency as a boolean or logarithmically scaled weighting. In all weight words methods, each document is translated to a vector (with length equal to that of the document) containing the frequency of the words in that document. Although this approach is intuitive, it is limited by the fact that particular words that are commonly used in the language may dominate such representations.

2.3.1. Bag of Words (BoW)

The bag-of-words model (BoW model) is a reduced and simplified representation of a text document from selected parts of the text, based on specific criteria, such as word frequency.

The BoW technique is used in several domains such as computer vision, NLP, Bayesian spam filters, as well as document classification and information retrieval by Machine Learning.

In a BoW, a body of text, such as a document or a sentence, is thought of like a bag of words. Lists of words are created in the BoW process. These words in a matrix are not sentences which structure sentences and grammar, and the semantic relationship between these words are ignored in their collection and construction. The words are often representative of the content of a sentence. While grammar and order of appearance are ignored, multiplicity is counted and may be used later to determine the focus points of the documents.

Here is an example of BoW:

Document

> *"As the home to UVA's recognized undergraduate and graduate degree programs in systems engineering. In the UVA Department of Systems and Information Engineering, our students are exposed to a wide range of range"*

Bag-of-Words (BoW)

> *{"As", "the", "home", "to", "UVA's", "recognized", "undergraduate", "and", "graduate", "degree", "program", "in", "systems", "engineering", "in", "Department", "Information","students", " ","are", "exposed", "wide", "range" }*

Bag-of-Feature (BoF)

> Feature = {1,1,1,3,2,1,2,1,2,3,1,1,1,2,1,1,1,1,1,1}

2.3.2. Limitation of Bag-of-Words

Bag0of-words models encode every word in the vocabulary as one-hot-encoded vector e.g., for the vocabulary of size $|\Sigma|$, each word is represented by a $|\Sigma|$ dimensional sparse vector with 1 at index corresponding to the word and 0 at every other index. As vocabulary may potentially run into millions, bag-of-word models face scalability challenges (e.g., "This is good" and "Is this good" have exactly the same vector representation). The technical problem of the bag-of-word is also the main challenge for the computer science and data science community.

Term frequency, also called bag-of-words, is the simplest technique of text feature extraction. This method is based on counting the number of words in each document and assigning it to the feature space.

2.3.3. Term Frequency-Inverse Document Frequency

K. Sparck Jones [59] proposed Inverse Document Frequency (IDF) as a method to be used in conjunction with term frequency in order to lessen the effect of implicitly common words in the corpus. IDF assigns a higher weight to words with either high or low frequencies term in the document. This combination of TF and IDF is well known as Term Frequency-Inverse document frequency (TF-IDF). The mathematical representation of the weight of a term in a document by TF-IDF is given in Equation (1).

$$W(d,t) = TF(d,t) * log(\frac{N}{df(t)}) \tag{1}$$

Here N is the number of documents and $df(t)$ is the number of documents containing the term t in the corpus. The first term in Equation (1) improves the recall while the second term improves the precision of the word embedding [60]. Although TF-IDF tries to overcome the problem of common terms in the document, it still suffers from some other descriptive limitations. Namely, TF-IDF cannot account for the similarity between the words in the document since each word is independently presented as an index. However, with the development of more complex models in recent years, new methods, such as word embedding, have been presented that can incorporate concepts such as similarity of words and part of speech tagging.

2.4. Word Embedding

Even though we have syntactic word representations, it does not mean that the model captures the semantics meaning of the words. On the other hand, bag-of-word models do not respect the semantics of the word. For example, words *"airplane"*, *"aeroplane"*, *"plane"*, and *"aircraft"* are often used in the same context. However, the vectors corresponding to these words are orthogonal in the bag-of-words model. This issue presents a serious problem to understanding sentences within the model. The other problem in the bag-of-word is that the order of words in the phrase is not respected. The *n-gram* does not solve this problem so a similarity needs to be found for each word in the sentence. Many researchers worked on word embedding to solve this problem. The Skip-gram and continuous bag-of-words (CBOW) models of [61] propose a simple single-layer architecture based on the inner product between two word vectors.

Word embedding is a feature learning technique in which each word or phrase from the vocabulary is mapped to a N dimension vector of real numbers. Various word embedding methods have been proposed to translate unigrams into understandable input for machine learning algorithms. This work focuses on Word2Vec, GloVe, and FastText, three of the most common methods that have been successfully used for deep learning techniques. Recently, the Novel technique of word representation was introduced where word vectors depend on the context of the word called *"Contextualized Word Representations"* or *"Deep Contextualized Word Representations"*.

2.4.1. Word2Vec

T. Mikolov et al. [61,62] presented *"word to vector"* representation as an improved word embedding architecture. The Word2Vec approach uses shallow neural networks with two hidden layers, continuous bag-of-words (CBOW), and the Skip-gram model to create a high dimension vector for each word. The Skip-gram model dives a corpus of words w and context c [10]. The goal is to maximize the probability:

$$arg\max_{\theta} \prod_{w \in T} \left[\prod_{c \in c(w)} p(c \mid w; \theta) \right] \tag{2}$$

where T refers to Text, and θ is parameter of $p(c \mid w; \theta)$.

Figure 2 shows a simple CBOW model which tries to find the word based on previous words, while Skip-gram tries to find words that might come in the vicinity of each word. The weights between the input layer and output layer represent $v \times N$ [63] as a matrix of w.

$$h = W^T c = W^T_{k,\cdot} := v^T_{wI} \tag{3}$$

This method provides a very powerful tool for discovering relationships in the text corpus as well as similarity between words. For example, this embedding would consider the two words such as "big" and "bigger" close to each other in the vector space it assigns them.

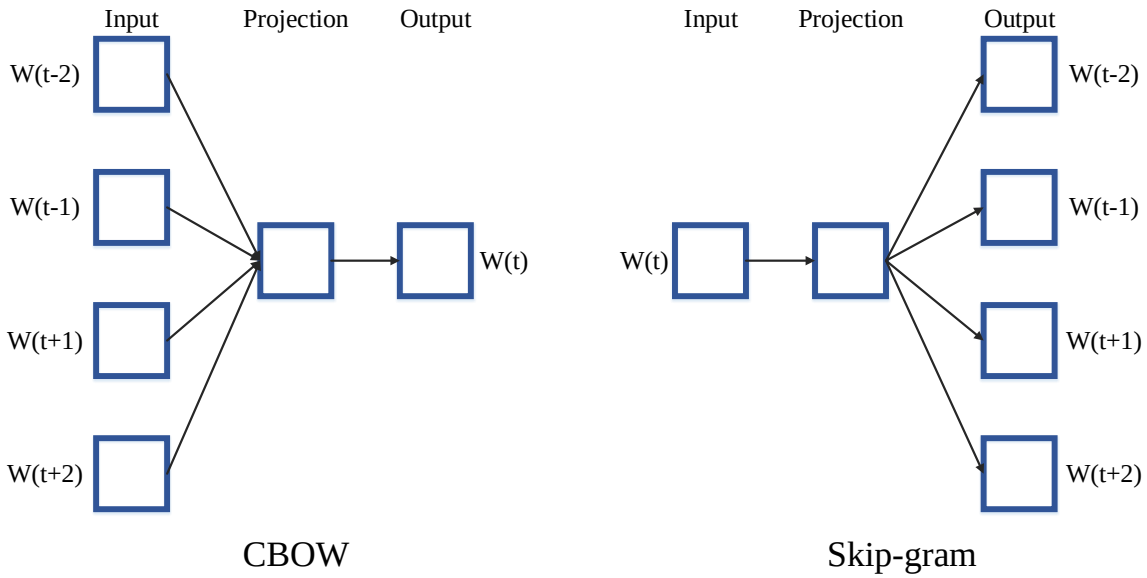

Figure 2. The continuous bag-of-words (CBOW) architecture predicts the current word based on the context, and the Skip-gram predicts surrounding words based on the given current word [61].

Continuous Bag-of-Words Model

The continuous bag-of-words model is represented by multiple words for a given target of words. For example, the word "airplane" and "military" as context words for "air-force" as the target word. This consists of replicating the input to hidden layer connections β times which is the number of context words [61]. Thus, the bag-of-words model is mostly used to represent an unordered collection of words as a vector. The first thing to do is create a vocabulary, which means all the unique words in the corpus. The output of the shallow neural network will be w_i that the task as *"predicting the word given its context"*. The number of words used depends on the setting for the window size (common size is 4–5 words).

Continuous Skip-Gram Model

Another model architecture which is very similar to CBOW [61] is the continuous Skip-gram model, however this model, instead of predicting the current word based on the context, tries to maximize classification of a word based on another word in the same sentence. The continuous bag-of-words model and continuous Skip-gram model are used to keep syntactic and semantic information of sentences for machine learning algorithms.

2.4.2. Global Vectors for Word Representation (GloVe)

Another powerful word embedding technique that has been used for text classification is Global Vectors (GloVe) [11]. The approach is very similar to the Word2Vec method, where each word is

presented by a high dimension vector and trained based on the surrounding words over a huge corpus. The pre-trained word embedding used in many works is based on 400,000 vocabularies trained over Wikipedia 2014 and Gigaword 5 as the corpus and 50 dimensions for word presentation. GloVe also provides other pre-trained word vectorizations with 100, 200, 300 dimensions which are trained over even bigger corpora, including Twitter content. Figure 3 shows a visualization of the word distances over a sample data set using the same t-SNE technique [64]. The objective function is as follows:

$$f(w_i - w_j, \widetilde{w}_k) = \frac{P_{ik}}{P_{jk}} \tag{4}$$

where w_i refers to the word vector of word i, and P_{ik} denotes to the probability of word k to occur in the context of word i.

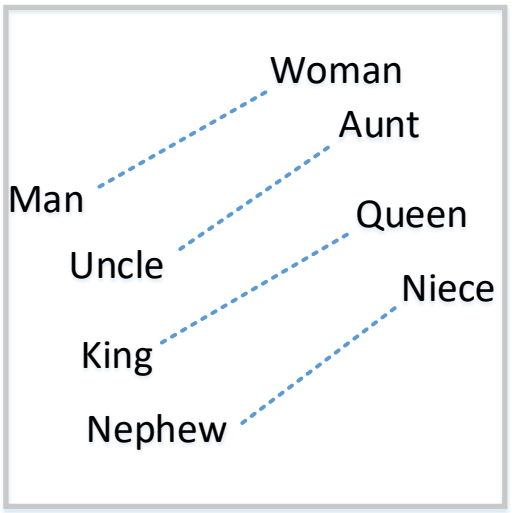

Figure 3. GloVe: Global Vectors for Word Representation.

2.4.3. FastText

Many other word embedding representations ignore the morphology of words by assigning a distinct vector to each word [65]. Facebook AI Research lab released a novel technique to solve this issue by introducing a new word embedding method called FastText. Each word, w, is represented as a bag of character n-gram. For example, given the word "*introduce*" and $n = 3$, FastText will produce the following representation composed of character tri-grams:

$$< in, int, ntr, tro, rod, odu, duc, uce, ce >$$

Note that the sequence <*int*>, corresponding to the word here is different from the tri-gram "*int*" from the word introduce.

Suppose we have a dictionary of n-grams of size G, and given a word w which is associated as a vector representation z_g to each n-gram g. The obtained scoring function [65] in this case is:

$$s(w, c) = \sum_{g \in g_w} z_g^T v_c \tag{5}$$

where $g_w \in \{1, 2, \ldots, G\}$.

Facebook published pre-trained word vectors for 294 languages which are trained on Wikipedia using FastText based on 300 dimension. The FastText used the Skip-gram model [65] with default parameters.

2.4.4. Contextualized Word Representations

Contextualized word representations are another word embedding technique which is based on the context2vec [66] technique introduced by B. McCann et al. The context2vec method uses bidirectional long short-term memory (LSTM). M.E. Peters et al. [67] built upon this technique to create the deep contextualized word representations technique. This technique contains both the main feature of word representation: (I) complex characteristics of word use (e.g., syntax and semantics) and (II) how these uses vary across linguistic contexts (e.g., to model polysemy) [67].

The main idea behind these word embedding techniques is that the resulting word vectors are learned from a bidirectional language model (biLM), which consist of both forward and backward LMs.

The forward LMs are as follows:

$$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k | t_1, t_2, \ldots, t_{k-1}) \tag{6}$$

The backward LMs are as follows:

$$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k | t_{k+1}, t_{k+2}, \ldots, t_N) \tag{7}$$

This formulation jointly maximizes the log-likelihood of the forward and backward directions as follows:

$$\sum_{k=1}^{N} \begin{pmatrix} \log p(t_k | t_1, \ldots, t_{k-1}; \Theta_x, \overrightarrow{\Theta}_{LSTM}, \Theta_s) + \\ \log p(t_k | t_{k+1}, \ldots, t_N; \Theta_x, \overleftarrow{\Theta}_{LSTM}, \Theta_s) \end{pmatrix} \tag{8}$$

where Θ_x is the token representation and Θ_x refers to the softmax layer. Then, ELMo is computed as a task-specific weighting for all biLM layers as follows:

$$ELMo_k^{task} = E(R_k; \Theta^{task}) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} h_{k,j}^{LM} \tag{9}$$

where $h_{k,j}^{LM}$ is calculated by:

$$h_{k,j}^{LM} = \left[\overrightarrow{h}_{k,j}^{LM}, \overleftarrow{h}_{k,j}^{LM} \right] \tag{10}$$

where s^{task} stands for softmax-normalized weights, and γ^{task} is the scalar parameter.

2.5. Limitations

Although the continuous bag-of-words model and continuous Skip-gram model are used to keep syntactic and semantic information of per-sentences for machine learning algorithms, there remains the issue how to keep full meaning of coherent documents for machine learning.

Example:

Document: {"*Maryam went to Paris on July 4th, 2018. She missed the independence day fireworks and celebrations. This day is a federal holiday in the United States commemorating the Declaration of Independence of the United States on July 4, 1776. The Continental Congress declared that the thirteen American colonies were no longer subject to the monarch of Britain and were now united, free, and independent states. She wants to stay in the country for next year and celebrate with her friends.*"}

Sentence level of this document:

S1: {"*Maryam went to Paris on July 4th, 2018.*"}
S2: {"*She missed the independence day fireworks and celebrations.*"}
S3: {"*This day is a federal holiday in the United States commemorating the Declaration of Independence of the*

United States on July 4, 1776."}
S4: {*"The Continental Congress declared that the thirteen American colonies were no longer subject to the monarch of Britain and were now united, free, and independent states."}*
S5: {*"She has a plan for next year to stay in the country and celebrate with her friends."}*

Limitation:

Figure 4 shows how the feature extraction fails for per-sentence level. The purple color shown in figure is the brief history of "This day". Furthermore, "This day" refers to "July 4th". In S5, "She" refers to the S1 "Maryam".

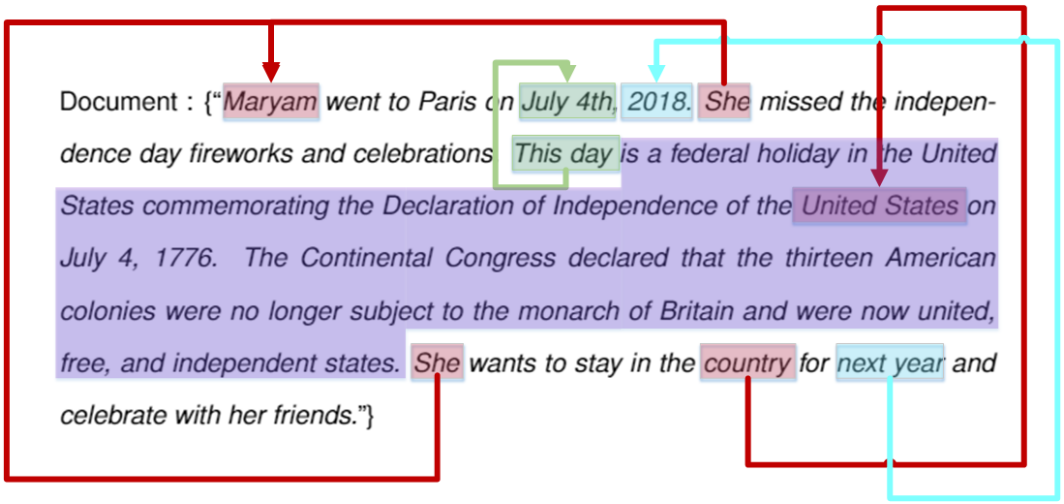

Figure 4. Limitation of document feature extraction by per-sentence level.

3. Dimensionality Reduction

Text sequences in term-based vector models consist of many features. Thus, time complexity and memory consumption are very expensive for these methods. To address this issue, many researchers use dimensionality reduction to reduce the size of feature space. In this section, existing dimensionality reduction algorithms are discussed in detail.

3.1. Component Analysis

3.1.1. Principal Component Analysis (PCA)

Principal component analysis (PCA) is the most popular technique in multivariate analysis and dimensionality reduction. PCA is a method to identify a subspace in which the data approximately lies [68]. This means finding new variables that are uncorrelated and maximizing the variance to "preserve as much variability as possible" [69].

Suppose a data set $x^{(i)}; i = 1, ..., m$ is given and $x^{(i)} \in \mathbb{R}^n$ for each i ($n \ll m$). The jth column of matrix X is vector, x_j that is the observations on the jth variable. The linear combination of x_js can be written as:

$$\sum_{j=1}^{m} a_j x_j = Xa \tag{11}$$

where a is a vector of constants $a_1, a_2, ..., a_m$. The variance of this linear combination can be given as:

$$var(Xa) = a^T Sa \tag{12}$$

where S is the sample co-variance matrix. The goal is to find the linear combination with maximum variance. This translates into maximizing $a^T Sa - \lambda(a^T a - 1)$, where λ is a Lagrange multiplier.

PCA can be used as a pre-processing tool to reduce the dimension of a data set before running a supervised learning algorithm on it ($x^{(i)}s$ as inputs). PCA is also a valuable tool as a noise reduction algorithm and can be helpful in avoiding the over-fitting problem [70]. kernel principal component analysis (KPCA) is another dimensionality reduction method that generalizes linear PCA into the nonlinear case by using the kernel method [71].

3.1.2. Independent Component Analysis (ICA)

Independent component analysis (ICA) was introduced by H. Jeanny [72]. This technique was then further developed by C. Jutten and J. Herault [73]. ICA is a statistical modeling method where the observed data are expressed as a linear transformation [74]. Assume that $4n$ linear mixtures $(x_1, x_2, \ldots, x_n)$ are observed where independent components:

$$x_j = a_{j1}s_1 + a_{j2}s_2 + \ldots + a_{jn}s_n \ \forall j \tag{13}$$

The vector-matrix notation is written as:

$$X = As \tag{14}$$

Denoting them by a_i, the model can also be written [75] as follows:

$$X = \sum_{i=1}^{n} a_i s_i \tag{15}$$

3.2. Linear Discriminant Analysis (LDA)

LDA is a commonly used technique for data classification and dimensionality reduction [76]. LDA is particularly helpful where the within-class frequencies are unequal and their performances have been evaluated on randomly generated test data. Class-dependent and class-independent transformation are two approaches to LDA in which the ratio of between class variance to within class variance and the ratio of the overall variance to within class variance are used respectively [77].

Let $x_i \in \mathbb{R}^d$ which be d-dimensional samples and $y_i \in \{1, 2, ..., c\}$ be associated target or output [76], where n is the number of documents and c is the number of categories. The number of samples in each class is calculated as follows:

$$S_w = \sum_{l=1}^{c} s_l \tag{16}$$

where

$$S_i = \sum_{x \in w_i} (x - \mu_i)(x - \mu_i)^T, \ \mu_i = \frac{1}{N_i} \sum_{x \in w_i} x \tag{17}$$

The generalization between the class scatter matrix is defined as follows:

$$S_B = \sum_{i=1}^{c} N_i (\mu_i - \mu)(\mu_i - \mu)^T \tag{18}$$

where

$$\mu = \frac{1}{N} \sum_{\forall x} x \tag{19}$$

Respect to $c - 1$ projection vector of w_i that can be projected into W matrix:

$$W = [w_1 | w_2 | \ldots | w_{c-1}] \tag{20}$$

$$y_i = w_i^T x \tag{21}$$

Thus, the μ (mean) vector and S matrices (scatter matrices) for the projected to lower dimension as follows:

$$\widetilde{S}_w = \sum_{i=1}^{c} \sum_{y \in w_i} (y - \widetilde{\mu}_i)(y - \widetilde{\mu}_i)^T \tag{22}$$

$$\widetilde{S}_B = \sum_{i=1}^{c} (\widetilde{\mu}_i - \widetilde{\mu})(\widetilde{\mu}_i - \widetilde{\mu})^T \tag{23}$$

If the projection is not scalar ($c - 1$ dimensions), the determinant of the scatter matrices will be used as follows:

$$J(W) = \frac{|\widetilde{S}_B|}{|\widetilde{S}_W|} \tag{24}$$

From the fisher discriminant analysis (FDA) [76,78], we can re-write the equation as:

$$J(W) = \frac{|W^T S_B W|}{|W^T S_W W|} \tag{25}$$

3.3. Non-Negative Matrix Factorization (NMF)

Non-negative matrix factorization (NMF) or non-negative matrix approximation has been shown to be a very powerful technique for very high dimensional data such as text and sequences analysis [79]. This technique is a promising method for dimension reduction [80]. In this section, a brief overview of NMF is discussed for text and document data sets. Given a non-negative $n \times m$ in matrix V is an approximation of:

$$V \approx WH \tag{26}$$

where $W = \mathbb{R}^{n \times r}$ and $H = \mathbb{R}^{r \times m}$. Suppose $(n + m)\ r < nm$, then the product WH can be regarded as a compressed form of the data in V. Then v_i and h_i are the corresponding columns of V and H. The computation of each corresponding column can be re-written as follows:

$$u_i \approx Wh_i \tag{27}$$

The computational time of each iteration, as introduced by S. Tsuge et al. [80], can be written as follows:

$$\overline{H}_{ij} = H_{ij} \frac{(W^T V)_{ij}}{(W^T WH)_{ij}} \tag{28}$$

$$\overline{W}_{ij} = W_{ij} \frac{(VH^T)_{ij}}{(WHH^T)_{ij}} \tag{29}$$

Thus, the local minimum of the objective function is calculated as follows:

$$F = \sum_i \sum_j (V_{ij} - (WH)_{ij})^2 \tag{30}$$

The maximization of the objective function can be re-written as follows:

$$F = \sum_i \sum_j (V_{ij} \log((WH)_{ij}) - (WH)_{ij}) \tag{31}$$

The objective function, given by the Kullback–Leibler [81,82] divergence, is defined as follows:

$$\overline{H}_{ij} = H_{ij} \sum_k W_{kj} \frac{V_{kj}}{(WH)_{kj}} \tag{32}$$

$$\hat{W}_{ij} = W_{ij} \sum_k \frac{V_{ik}}{(WH)_{ik}} H_{jk} \tag{33}$$

$$\overline{W}_{ij} = \frac{\hat{W}_{ij}}{\sum_k \hat{W}_{kj}} \tag{34}$$

This NMF-based dimensionality reduction contains the following 5 steps [80] (step VI is optional but commonly used in information retrieval:

(I) Extract index term after pre-processing stem like feature extraction and text cleaning as discussed in Section 2. Then we have n documents with m features;
(II) Create n documents ($d \in \{d_1, d_2, \ldots, d_n\}$), where vector $a_{ij} = L_{ij} \times G_i$ where L_{ij} refers to local weights of i_{-th} term in document j, and G_i is global weights for document i;
(III) Apply NMF to all terms in all documents one by one;
(IV) Project the trained document vector into r-dimensional space;
(V) Using the same transformation, map the test set into the r-dimensional space;
(VI) Calculate the similarity between the transformed document vectors and a query vector.

3.4. Random Projection

Random projection is a novel technique for dimensionality reduction which is mostly used for high volume data set or high dimension feature spaces. Texts and documents, especially with weighted feature extraction, generate a huge number of features. Many researchers have applied random projection to text data [83,84] for text mining, text classification, and dimensionality reduction. In this section, we review some basic random projection techniques. As shown in Figure 5, the overview of random projection is shown.

3.4.1. Random Kitchen Sinks

The key idea of random kitchen sinks [85] is sampling via monte carlo integration [86] to approximate the kernel as part of dimensionality reduction. This technique works only for shift-invariant kernel:

$$K(x, x') =< \phi(x), \phi(x') > \approx K(x - x') \tag{35}$$

where shift-invariant kernel, which is an approximation kernel of:

$$K(x - x') = z(x)z(x') \tag{36}$$

$$K(x, x') = \int_{R^D} P(w) e^{iw^T(x - x')} \tag{37}$$

where D is the target number of samples, $P(w)$ is a probability distribution, w stands for random direction, and $w \in \mathbb{R}^{F \times D}$ where F is the number of features and D is the target.

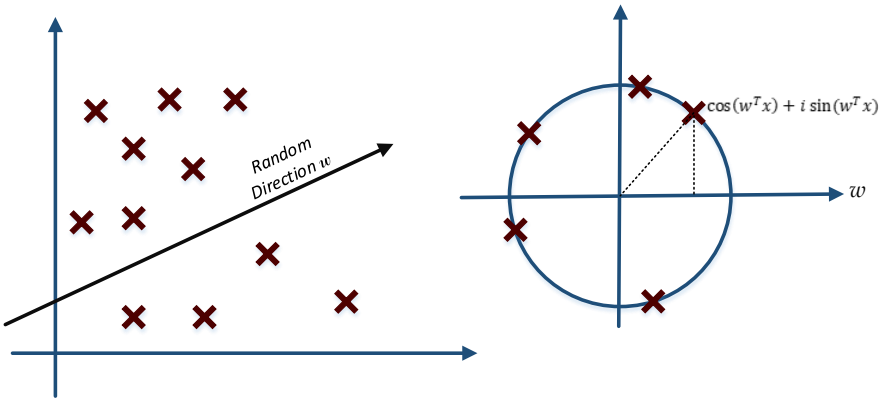

Figure 5. The plot on the left shows how we generate random direction, and the plot on the right shows how we project the data set into the new space using complex numbers.

$$K(x, x') = K(x - x') \approx \frac{1}{D} \sum_{j=1}^{D} e^{iw_j^T (x - x')} \tag{38}$$

$$\begin{aligned} \frac{1}{D} \sum_{j=1}^{D} e^{iw_j^T (x - x')} = & \frac{1}{D} \sum_{j=1}^{D} e^{iw_j^T x} e^{iw_j^T x'} = \\ & \frac{1}{\sqrt{D}} \sum_{j=1}^{D} e^{iw_j^T x} \frac{1}{\sqrt{D}} \sum_{j=1}^{D} e^{iw_j^T x'} \end{aligned} \tag{39}$$

$$k(x - x') \approx \phi(x) \phi(x') \tag{40}$$

$$\phi(x) = \cos(w^T x + b_i) \tag{41}$$

where the b_i is uniform random variable ($b_i \in [0, \pi]$).

3.4.2. Johnson Lindenstrauss Lemma

William B. Johnson and Joram Lindenstrauss [87,88] proved that for any n point Euclidean space can be bounded in $k = O(\frac{logn}{\epsilon^2})$ for any u and $v \in n$ and $n \in \mathbb{R}^d$: $\exists f : \mathbb{R}^d \to \mathbb{R}^k | \epsilon \in [0, 1]$. With $x = u - v$ to the lower bound of the success probability.

$$(i - \epsilon)||u - v||^2 \leq ||f(u) - f(v)||^2 \leq (i + \epsilon)||u - v||^2 \tag{42}$$

Johnson Lindenstrauss Lemma Proof [89]:

For any V sets of data point from n where $V \in n$ and random variable $w \in R^{k \times d}$:

$$Pr[success] \geq 1 - 2m^2 e^{\frac{-k(\epsilon^2 - \epsilon^3)}{4}} \tag{43}$$

If we let $k = \frac{16logn}{\epsilon^2}$:

$$\begin{aligned} 1 - 2m^2 e^{\frac{-k(\epsilon^3 - \epsilon^3)}{4}} \geq & 1 - 2m^2 e^{\frac{(-\frac{8logn}{\epsilon^2})(\epsilon^2 - \epsilon^3)}{4}} \\ & 1 - 2m^2 e^{\frac{-\frac{16logn}{\epsilon^2})(\epsilon^3 - \epsilon^3)}{4}} = \\ & 1 - 2m^{4\epsilon - 2} > 1 - 2m^{-\frac{1}{2}} > 0 \end{aligned} \tag{44}$$

Lemma 1 Proof [89]:

Let Ψ be a random variable with k degrees of freedom, then for $\epsilon \in [0,1]$

$$Pr[(1-\epsilon)k \leq \Psi \leq (1+\epsilon)k] \geq 1 - 2e^{\frac{-k(\epsilon^2-\epsilon^3)}{4}} \tag{45}$$

We start with Markov's inequality [90]:

$$Pr[(\Psi \geq (1-\epsilon)k)] \leq \frac{E[\Psi]}{(1-\epsilon)k} \tag{46}$$

$$\begin{aligned} Pr[e^{\lambda\Psi} \geq e^{\lambda(1-\epsilon)k)}] &\leq \frac{E[e^{\lambda\Psi}]}{e^{\lambda(1-\epsilon)k}} \\ E[e^{\lambda\Psi}] &= (1-2\lambda)^{-\frac{k}{2}} \end{aligned} \tag{47}$$

where $\lambda < 0.5$ and using the fact of $(1-\epsilon) \leq e^{\epsilon - \frac{(\epsilon^2-\epsilon^3)}{2}}$; thus, we can proof $Pr[(\Psi \geq (1-\epsilon)k)] \leq \frac{E[\Psi]}{(1-\epsilon)k}$ and the $Pr[(\Psi \leq (1+\epsilon)k)] \leq \frac{E[\Psi]}{(1+\epsilon)F}$ is similar.

$$\frac{(1+\epsilon)}{e^{\epsilon}} \leq \left(\frac{e^{\epsilon - \frac{(\epsilon^2-\epsilon^3)}{2}}}{e^{\epsilon}}\right)^{\frac{k}{2}} = e^{\frac{-k(\epsilon^3-\epsilon^3)}{4}} \tag{48}$$

$$\begin{aligned} Pr[\overline{(1-\epsilon)k \leq \Psi \leq (1+\epsilon)k}] \leq \\ Pr[(1-\epsilon)k \geq \Psi \cup \Psi \leq (1+\epsilon)k] = \\ 2e^{\frac{-k(\epsilon^3-\epsilon^3)}{4}} \end{aligned} \tag{49}$$

Lemma 2 Proof [89]:

Let w be a random variable of $w \in \mathbb{R}^{k \times d}$ and $k < d$, x is data points $x \in \mathbb{R}^d$ then for any $\epsilon \in [0,1]$:

$$\begin{aligned} Pr[(1-\epsilon)||x||^2 \leq ||\frac{1}{\sqrt{k}}wx||^2 \leq \\ (1+\epsilon)||x||^2] \geq 1 - 2e^{\frac{-k(\epsilon^3-\epsilon^3)}{4}} \end{aligned} \tag{50}$$

In Equation (50), $\frac{1}{\sqrt{k}}wx$ is the random approximation value and $\widehat{x} = wx$, so we can rewrite the Equation (50) by $Pr[(1-\epsilon)||x||^2 \leq ||\frac{1}{\sqrt{k}}\widehat{x}||^2 \leq (1+\epsilon)||x||^2] \geq 1 - 2e^{\frac{-k(\epsilon^3-\epsilon^3)}{4}}$.

Call $\zeta_i = \frac{\widehat{x}_i}{||x||} \sim N(0,1)$ and $\Psi = \sum_{i=1}^{k} \zeta_i^2$ thus:

$$\begin{aligned} Pr[(1-\epsilon)k \leq ||\sum_{i=0}^{k} \zeta_i||^2 \leq (1+\epsilon)k] = \\ Pr[(1-\epsilon)k \leq ||w||^2 \leq (1+\epsilon)k] \end{aligned} \tag{51}$$

where we can prove Equation (51) by using Equation (45):

$$Pr[(1-\epsilon)k \leq \Psi \leq (1+\epsilon)k] \geq 1 - 2e^{\frac{-k(\epsilon^3-\epsilon^3)}{4}} \tag{52}$$

3.5. Autoencoder

An autoencoder is a type of neural network that is trained to attempt to copy its input to its output [91]. The autoencoder has achieved great success as a dimensionality reduction method via the powerful reprehensibility of neural networks [92]. The first version of autoencoder was introduced

by D.E. Rumelhart et al. [93] in 1985. The main idea is that one hidden layer between input and output layers has fewer units [94] and could thus be used to reduce the dimensions of a feature space. Especially for texts, documents, and sequences that contain many features, using an autoencoder could help allow for faster, more efficient data processsing.

3.5.1. General Framework

As shown in Figure 6, the input and output layers of an autoencoder contain n units where $x = \mathbb{R}^n$, and hidden layer Z contains p units with respect to $p < n$ [95]. For this technique of dimensionality reduction, the dimensions of the final feature space are reduced from $n \rightarrow p$. The encoder representation involves a sum of the representation of all words (for bag-of-words), reflecting the relative frequency of each word [96]:

$$a(x) = c + \sum_{i=1}^{|x|} W_{.,x_i}, \phi(x) = h(a(x)) \tag{53}$$

where $h(.)$ is an element-wise non-linearity such as the sigmoid (Equation (79)).

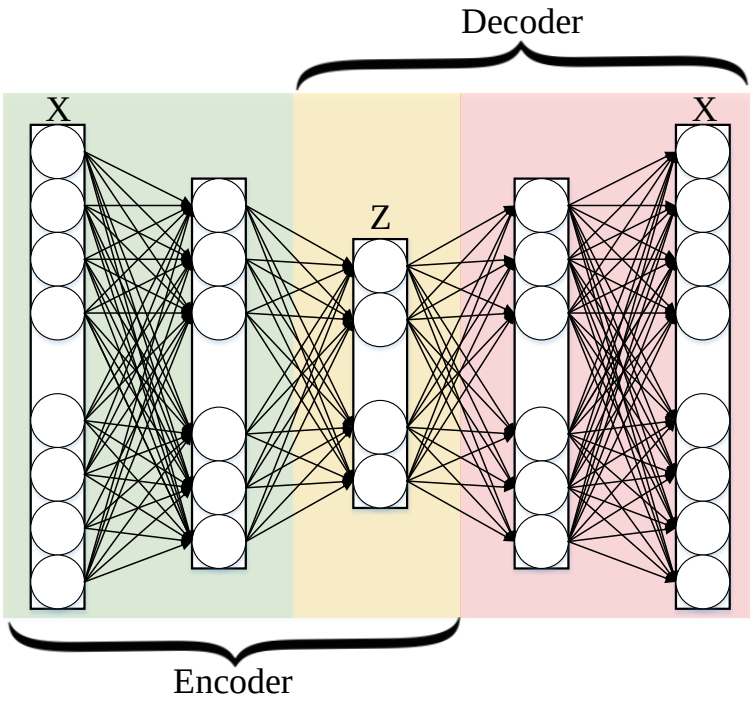

Figure 6. This figure shows how a simple autoencoder works. The model depicted contains the following layers: Z is code and two hidden layers are used for encoding and two are used for decoding.

3.5.2. Conventional Autoencoder Architecture

A convolutional neural networks (CNN)-based autoencoder can be divided into two main steps [97] (encoding and decoding).

$$O_m(i,j) = a\left(\sum_{d=1}^{D}\sum_{u=-2k-1}^{2k+1}\sum_{v=-2k-1}^{2k+1} F_{m_d}^{(1)}(u,v) I_d(i-u, j-v)\right) \quad \forall m = 1, \cdots, n \tag{54}$$

where $F \in \{F_1^{(1)}, F_2^{(1)}, \ldots, F_n^{(1)},\}$ which is a convolutional filter, with convolution among an input volume defined by $I = \{I_1, \cdots, I_D\}$ which learns to represent input combining non-linear functions:

$$z_m = O_m = a(I * F_m^{(1)} + b_m^{(1)}) \quad m = 1, \cdots, m \tag{55}$$

where $b_m^{(1)}$ is the bias, and the number of zeros we want to pad the input with is such that: dim(I) = dim(decode(encode(I))). Finally, the encoding convolution is equal to:

$$\begin{aligned} O_w = O_h &= (I_w + 2(2k+1) - 2) - (2k+1) + 1 \\ &= I_w + (2k+1) - 1 \end{aligned} \tag{56}$$

The decoding convolution step produces n feature maps $z_{m=1,\ldots,n}$. The reconstructed results $\hat{I}$ is the result of the convolution between the volume of feature maps $Z = \{z_{i=1}\}^n$ and this convolutional filters volume $F^{(2)}$ [97–99].

$$\tilde{I} = a(Z * F_m^{(2)} + b^{(2)}) \tag{57}$$

$$\begin{aligned} O_w = O_h =& (I_w + (2k+1) - 1) - \\ & (2k+1) + 1 = I_w = I_h \end{aligned} \tag{58}$$

where Equation (58) shows the decoding convolution with I dimensions. Input's dimensions are equal to the output's dimensions.

3.5.3. Recurrent Autoencoder Architecture

A recurrent neural network (RNN) is a natural generalization of feedforward neural networks to sequences [100]. Figure 7 illustrate recurrent autoencoder architecture. A standard RNN compute the econding as a sequences of output by iteration:

$$h_t = sigm(W^{hx} x_t + W^{hh} h_{t-1}) \tag{59}$$

$$y_t = W^{y^h} h_t \tag{60}$$

where x is inputs $(x_1, ..., x_T)$ and y refers to output $(y_1, ..., y_T)$. A multinomial distribution (1-of-K coding) can be output using a softmax activation function [101]:

$$p(x_{t,j} = 1 \mid x_{t-1,\ldots,x_1}) = \frac{\exp(w_j h_t)}{\sum_{j'=1}^{K} \exp(w_{j'} h_t)} \tag{61}$$

By combining these probabilities, we can compute the probability of the sequence x as:

$$p(x) = \prod_{t=1}^{T} p(x_t \mid x_{t-1}, ..., x_1) \tag{62}$$

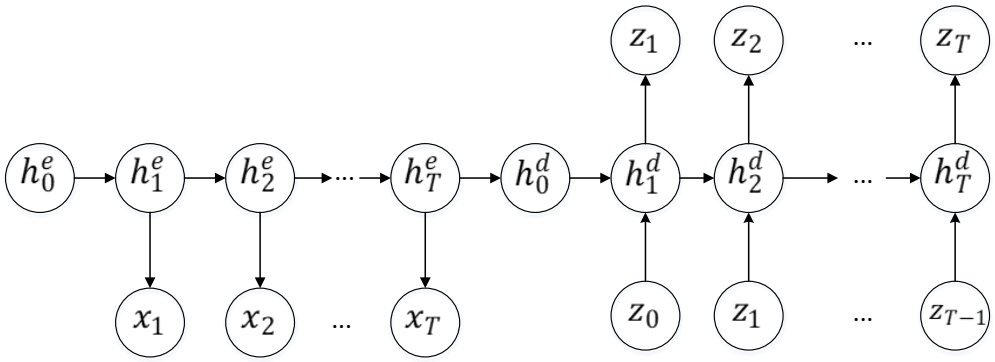

Figure 7. A recurrent autoencoder structure.

3.6. T-distributed Stochastic Neighbor Embedding (t-SNE)

T-SNE is a nonlinear dimensionality reduction method for embedding high-dimensional data. This method is mostly commonly used for visualization in a low-dimensional feature space [64], as shown in Figure 8. This approach is based on G. Hinton and S. T. Roweis [102]. SNE works by converting the high dimensional Euclidean distances into conditional probabilities which represent similarities [64]. The conditional probability $p_{j|i}$ is calculated by:

$$p_{j|i} = \frac{\exp\left(-\frac{||x_i - x_j||^2}{2\sigma_i^2}\right)}{\sum_{k \neq i} \exp\left(-\frac{||x_i - x_k||^2}{2\sigma_i^2}\right)} \tag{63}$$

where σ_i is the variance of the centered on data point x_i. The similarity of y_j to y_i is calculated as follows:

$$q_{j|i} = \frac{\exp\left(-||y_i - y_j||^2\right)}{\sum_{k \neq i} \exp\left(-||y_i - y_k||^2\right)} \tag{64}$$

The cost function C is as follows:

$$C = \sum_i KL(p_i|Q_i) \tag{65}$$

where $KL(P_i|Q_i)$ is the Kullback–Leibler divergence [103], which is calculated as:

$$KL(P_i|Q_i) = \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}} \tag{66}$$

The gradient update with a momentum term is as follows:

$$\gamma^{(t)} = \gamma^{(t-1)} + \eta \frac{\delta C}{\delta \gamma} + \alpha(t)\left(\gamma^{(t-1)} - \gamma^{(t-2)}\right) \tag{67}$$

where η is the learning rate, $\gamma^{(t)}$ refers to the solution at iteration t, and $\alpha(t)$ indicates momentum at iteration t. Now we can re-write symmetric SNE in the high-dimensional space and a joint probability distribution, Q, in the low-dimensional space as follows [64]:

$$C = KL(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{68}$$

in the high-dimensional space p_{ij} is:

$$p_{ij} = \frac{\exp\left(-\frac{||x_i - x_j||^2}{2\sigma^2}\right)}{\sum_{k \neq l} \exp\left(-\frac{||x_i - x_l||^2}{2\sigma^2}\right)} \tag{69}$$

The gradient of symmetric S is as follows:

$$\frac{\delta C}{\delta y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j) \tag{70}$$

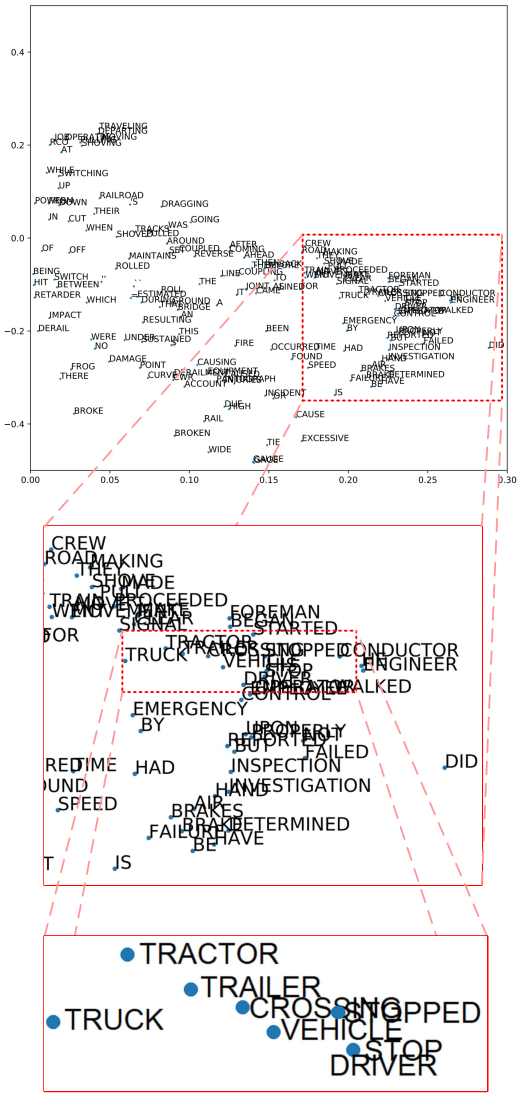

Figure 8. This figure presents the t-distributed stochastic neighbor embedding (t-SNE) visualization of Word2vec of the Federal Railroad Administration (FRA) data set.

4. Existing Classification Techniques

In this section, we outline existing text and document classification algorithms. First, we describe the Rocchio algorithm which is used for text classification. Then, we address two popular techniques in ensemble learning algorithms: Boosting and bagging. Some methods, such as logistic regression,

Naïve Bayes, and k-nearest neighbor, are more traditional but still commonly used in the scientific community. Support vector machines (SVMs), especially kernel SVMs, are also broadly used as a classification technique. Tree-based classification algorithms, such as decision tree and random forests are fast and accurate for document categorization. We also describe neural network based algorithms such as deep neural networks (DNN), CNN, RNN, deep belief network (DBN), hierarchical attention networks (HAN), and combination techniques.

4.1. Rocchio Classification

The Rocchio algorithm was first introduced by J.J. Rocchio [104] in 1971 as method of using relevance feedback to query full-text databases. Since then, many researchers have addressed and developed this technique for text and document classification [105,106]. This classification algorithm uses TF-IDF weights for each informative word instead of boolean features. Using a training set of documents, the Rocchio algorithm builds a prototype vector for each class. This prototype is an average vector over the training documents' vectors that belong to a certain class. It then assigns each test document to the class with the maximum similarity between the test document and each of the prototype vectors [107]. The average vector computes the centroid of a class c (center of mass of its members):

$$\vec{\mu}(c) = \frac{1}{|D_c|} \sum_{d \in D_c} \vec{v}_d \tag{71}$$

where D_c is the set of documents in D that belongs to class c and $\vec{v}_d$ is the weighted vector representation of document d. The predicted label of document d is the one with the smallest Euclidean distance between the document and the centroid:

$$c^* = \arg\min_c \|\vec{\mu}_c - \vec{v}_d\| \tag{72}$$

Centroids can be normalized to unit-length as follows:

$$\vec{\mu}_c = \frac{\sum_{d \in D_c} \vec{v}_d}{\|\sum_{d \in D_c} \vec{v}_d\|} \tag{73}$$

Therefore, the label of test documents can be obtained as follows:

$$c_* = \arg\min_c \vec{\mu}_c \cdot \vec{v}_d \tag{74}$$

Limitation of Rocchio Algorithm

The Rocchio algorithm for text classifcation contains many limitations such as the fact that the user can only retrieve a few relevant documents using this model [108]. Furthermore, this algorithms' results illustrate by taking semantics into consideration [109].

4.2. Boosting and Bagging

Voting classification techniques, such as bagging and boosting, have been successfully developed for document and text data set classification [110]. While boosting adaptively changes the distribution of the training set based on the performance of previous classifiers, bagging does not look at the previous classifier [111].

4.2.1. Boosting

The boosting algorithm was first introduced by R.E. Schapire [112] in 1990 as a technique for boosting the performance of a weak learning algorithm. This technique was further developed by Freund [113,114].

Figure 9 shows how a boosting algorithm works for 2D data sets, as shown we have labeled the data, then trained by multi-model architectures (ensemble learning). These developments resulted in the AdaBoost (Adaptive Boosting) [115]. Suppose we construct D_t such that $D_1(i) = \frac{1}{m}$ given D_t and h_t:

$$\begin{aligned} D_{t+1}(i) &= \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & if y_i = h_t(x_i) \\ e^{\alpha_t} & if y_i \neq h_t(x_i) \end{cases} \\ &= \frac{D_t(i)}{Z_t} exp(-\alpha y_i h_i(x_i)) \end{aligned} \tag{75}$$

where Z_t refers to the normalization factor and α_t is as follows:

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right) \tag{76}$$

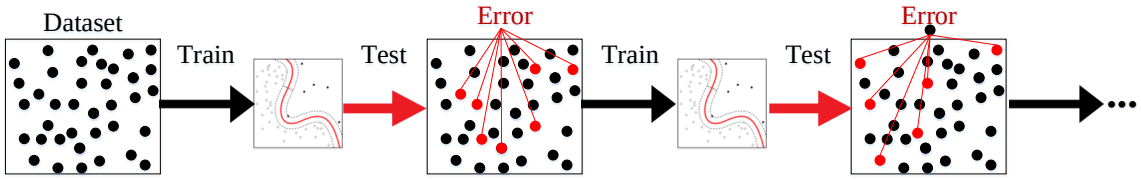

Figure 9. This figure is the boosting technique architecture.

As shown in Algorithm 1, training set S of size m, inducer τ and integer N as input. Then this algorithm find the weights of each x_j, and finally, the output is the optimal classifier (C^*).

Algorithm 1 The AdaBoost method

input : training set S of size m, inducer τ, integer N

for $i = 1$ *to* N **do**
  $C_i = \tau(S')$

  $$\epsilon_i = \frac{1}{m} \sum_{x_j \in S'; C_i(x_j) \notin y_i} weight(x)$$

  if $\epsilon_i > \frac{1}{2}$ **then**
    set S′ to a bootstrap sample from S with weight 1 for all instance and go top
  endif
  $\beta_i = \frac{\epsilon_i}{1-\epsilon_i}$
  for $x_i \in S'$ **do**
    if $C_i(x_j) = y_i$ **then**
      $weight(x_j) = weight(x_j).\beta_i$
    endif
  endfor
  Normalize weights of instances
endfor

$$C^*(x) = arg \max_{y \in Y} \sum_{i, C_i(x) = y} \log \frac{1}{\beta_i}$$

output: Classifier C^*

The final classifier formulation can be written as:

$$H_f(x) = sign\left(\sum_t \alpha_t h_t(x)\right) \tag{77}$$

4.2.2. Bagging

The bagging algorithm was introduced by L. Breiman [116] in 1996 as a voting classifier method. The algorithm is generated by different bootstrap samples [111]. A bootstrap generates a uniform sample from the training set. If N bootstrap samples $B_1, B_2, \ldots, B_N$ have been generated, then we have N classifiers (C) which C_i is built from each bootstrap sample B_i. Finally, our classifier C contain or generated from $C_1, C_2, ..., C_N$ whose output is the class predicted most often by its sub-classifiers, with ties broken arbitrarily [111,116]. Figure 10 shows a simple bagging algorithm which trained N models. As shown in Algorithm 2, We have training set S which is trained and find the best classifier C.

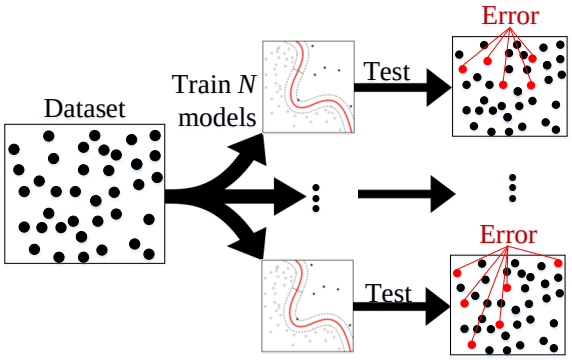

Figure 10. This figure shows a simple model of the bagging technique.

Algorithm 2 Bagging

input : training set S, inducer τ, integer N

for $i = 1$ *to* N **do**
  S' = bootstrap sample from S
  $C_i = \tau(S')$
endfor

$$C^*(x) = arg\max_{y \in Y} \sum_{i, C_i = y} 1$$

output: Classifier C^*

4.2.3. Limitation of Boosting and Bagging

Boosting and bagging methods also have many limitations and disadvantages, such as the computational complexity and loss of interpretability [117], which means that the feature importance could not be discovered by these models.

4.3. Logistic Regression

One of the earliest methods of classification is logistic regression (LR). LR was introduced and developed by statistician David Cox in 1958 [118]. LR is a linear classifier with decision boundary of $\theta^T x = 0$. LR predicts probabilities rather than classes [119,120].

4.3.1. Basic Framework

The goal of LR is to train from the probability of variable Y being 0 or 1 given x. Let us have text data which is $X \in \mathbb{R}^{n \times d}$. If we have binary classification problems, the Bernoulli mixture models function should be used [121] as follows:

$$\begin{aligned} L(\theta \mid x) = p(y \mid x; \theta) = \\ \prod_{i=1}^{n} \beta(y_i \mid sigm(x_i\theta)) = \\ \prod_{i=1}^{n} sigm(x_i)^{y_i}(1 - sigm(x_i))^{1-y_i} = \\ \prod_{i=1}^{n} \left[\frac{1}{1+e^{-x_i\theta}}\right]^{y_i} \left[1 - \frac{1}{1+e^{-x_i\theta}}\right]^{(1-y_i)} \end{aligned} \tag{78}$$

where $x_i\theta = \theta_0 + \sum_{j=1}^{d}(x_{ij}\theta_j)$, and $sigm(.)$ is a sigmoid function which is defined as shown in Equation (79).

$$sigm(\eta) = \frac{1}{1-e^{-\eta}} = \frac{e^{\eta}}{1-e^{\eta}} \tag{79}$$

4.3.2. Combining Instance-Based Learning and LR

The LR model specifies the probability of binary output $y_i = \{0, 1\}$ given the input x_i. we can consider posterior probability as:

$$\pi_0 = P(y_0 = +1 \mid y_i) \tag{80}$$

where:

$$\frac{\pi_0}{1-\pi_0} = \frac{P(y_i \mid y_0 = +1)}{P(y_i \mid y_0 = +1)} . \frac{p_0}{1-p_0} \tag{81}$$

where p is the likelihood ratio it could be re-written as:

$$\frac{\pi_0}{1-\pi_0} = p . \frac{p_0}{1-p_0} \tag{82}$$

$$\log\left(\frac{\pi_0}{1-\pi_0}\right) = \log(p) + w_0 \tag{83}$$

with respect to:

$$w_0 = \log(p_0) - \log(1-p_0) \tag{84}$$

To obey the basic principle underlying instance-based learning (IBL) [122], the classifier should be a function of the distance δ_i. p will be large if $\delta_i \to 0$ then $y_i = +1$, and small for $y_i = -1$. p should be close to 1 if $\delta_i \to \infty$; then, neither in favor of $y_0 = +1$ nor in favor of $y_0 = -1$, so the parameterized function is as follows:

$$p = p(\delta) = \exp\left(y_i . \frac{\alpha}{\delta}\right) \tag{85}$$

Finally,

$$\log\left(\frac{\pi_0}{1-\pi_0}\right) = w_0 + \alpha \sum_{x_i \in N(x_0)} k(x_0, x_i) . y_i \tag{86}$$

where $k(x_0, x_i)$ is similarity measure.

4.3.3. Multinomial Logistic Regression

Multinomial (or multilabeled) logistic classification [123] uses the probability of x belonging to class i (as defined in Equation (87))

$$p\left(y^{(i)} = 1 \mid x, \theta\right) = \frac{\exp\left({\theta^{(i)}}^T x\right)}{\sum_{j=1}^{m} \exp\left({\theta^{(i)}}^T x\right)} \tag{87}$$

where $\theta^{(i)}$ is the weight vector corresponding to class i.

For binary classification ($m = 2$) which is known as a basic LR, but for multinomial logistic regression ($m > 2$) is usually uses the $softmax$ function.

The normalization function is:

$$\sum_{i=1}^{m} p\left(y^{(i)} = 1 \mid x, \theta\right) = 1 \tag{88}$$

In a classification task as supervised learning context, the component of θ is calculated from the subset of the training data D which belongs to class i where $i \in \{1, \ldots, n\}$. To perform maximum likelihood (ML) estimation of θ, we need to maximize the log-likelihood function as follows:

$$\begin{aligned} \ell(\theta) &= \sum_{j=1}^{n} \log p\left(y_j = 1 \mid x_j, \theta\right) \\ &= \sum_{j=1}^{n} \left[\sum_{i=1}^{m} y_j^{(i)} {\theta^{(i)}}^T x_j - \log \sum_{i=1}^{m} \exp\left({\theta^{(i)}}^T x_j\right)\right] \end{aligned} \tag{89}$$

The adoption of a maximum a posteriori (MAP) estimates as follows:

$$\hat{\theta}_{MAP} = arg \max_{\theta} L(\theta) = arg \max_{\theta} \left[\ell(\theta) + \log p(\theta)\right] \tag{90}$$

4.3.4. Limitation of Logistic Regression

Logistic regression classifier works well for predicting categorical outcomes. However, this prediction requires that each data point be independent [124] which is attempting to predict outcomes based on a set of independent variables [125]

4.4. Naïve Bayes Classifier

Naïve Bayes text classification has been widely used for document categorization tasks since the 1950s [126,127]. The Naïve Bayes classifier method is theoretically based on Bayes theorem, which was formulated by Thomas Bayes between 1701–1761 [128,129]). Recent studies have widely addressed this technique in information retrieval [130]. This technique is a generative model, which is the most traditional method of text categorization. We start with the most basic version of NBC which was developed by using TF (bag-of-word), a feature extraction technique which counts the number of words in documents.

4.4.1. High-Level Description of Naïve Bayes Classifier

If the number of documents (n) fit into k categories where $k \in \{c_1, c_2, ..., c_k\}$, the predicted class as output is $c \in C$. The Naïve Bayes algorithm can be described as follows:

$$P(c \mid d) = \frac{P(d \mid c)P(c)}{P(d)} \tag{91}$$

where d is document and c indicates classes.

$$\begin{aligned} C_{MAP} &= \arg\max_{c \in C} P(d \mid c)P(c) \\ &= \arg\max_{c \in C} P(x_1, x_2, ..., x_n \mid c)p(c) \end{aligned} \tag{92}$$

This model is used as baseline of many papers which is word-level of Naïve Bayes classifier [3,131] as follows:

$$P(c_j \mid d_i; \hat{\theta}) = \frac{P(c_j \mid \hat{\theta})P(d_i \mid c_j; \hat{\theta}_j)}{P(d_i \mid \hat{\theta})} \tag{93}$$

4.4.2. Multinomial Naïve Bayes Classifier

If the number of documents (n) fit into k categories where $k \in \{c_1, c_2, ..., c_k\}$ the predicted class as output is $c \in C$. The Naïve Bayes algorithm can be written as:

$$P(c \mid d) = \frac{P(c) \prod_{w \in d} P(d \mid c)^{n_{wd}}}{P(d)} \tag{94}$$

where n_{wd} is denoted to the number of times word w occurs in document, and $P(w|c)$ is the probability of observing word w given class c [132].

$P(w|c)$ is calculated as:

$$P(w \mid c) = \frac{1 + \sum_{d \in D_c} n_{wd}}{k + \sum_{w'} \sum_{d \in D_c} n_{w'd}} \tag{95}$$

4.4.3. Naïve Bayes Classifier for Unbalanced Classes

One of the limitations of NBC is that the technique performs poorly on data sets with unbalanced classes [133]. Eibe Frank and Remco R. Bouckaert [132] developed a method for introducing normalization in each class by Equation (96) and then uses the centroid classifier [22] in NBC for unbalanced classes. The centroid c_c for class c is given in Equation (97).

$$\alpha \times \frac{n_{wd}}{\sum_{w'} \sum_{d \in D_c} n_{w'd}} \tag{96}$$

$$c_c = \left\{ \frac{\sum_{d \in D_c} n_{w_1 d}}{\sqrt{\sum_w \left(\sum_{d \in D_c} n_{wd} \right)^2}}, \ldots, \frac{\sum_{d \in D_c} n_{w_i d}}{\sqrt{\sum_w \left(\sum_{d \in D_c} n_{wd} \right)^2}}, \ldots, \frac{\sum_{d \in D_c} n_{w_k d}}{\sqrt{\sum_w \left(\sum_{d \in D_c} n_{wd} \right)^2}} \right\} \tag{97}$$

The scoring function is defined as:

$$x_d.c_1 - x_d.c_2 \tag{98}$$

So log of multinomial Naïve Bayes classifier can be calculated as:

$$\left[\log P(c_1) + \sum_{i=1}^{k} n_{w_i d} \log(P(w_i \mid c_1)) \right] - \left[\log P(c_2) + \sum_{i=1}^{k} n_{w_i d} \log(P(w_i \mid c_2)) \right] \tag{99}$$

Using Equations (95) and (96), and if $\alpha = 1$ we can rewrite:

$$P(w \mid c) = \frac{1 + \frac{n_{wd}}{\sum_{w'} \sum_{d \in D_c} n_{w'd}}}{K + 1} \tag{100}$$

with respect to:

$$\frac{\sum_{d \in D_c} n_{wd}}{\sum_{w'} \sum_{d \in D_c} n_{w'd}} << 1 \tag{101}$$

For text data sets and $log(x+1) \approx x$ and $x << 1$ [132]. In this technique of NBC, the experimental results is very similar to the centroid classifier [22].

4.4.4. Limitation of Naïve Bayes Algorithm

Naïve Bayes algorithm also has several limitations. NBC makes a strong assumption about the shape of the data distribution [134,135]. NBC is also limited by data scarcity for which any possible value in feature space, a likelihood value must be estimated by a frequentist [136].

4.5. K-Nearest Neighbor

The k-nearest Neighbors algorithm (KNN) is a non-parametric technique used for classification. This method is used for text classification applications in many research domains [137] in past decades.

4.5.1. Basic Concept of KNN

Given a test document x, the KNN algorithm finds the k nearest neighbors of x among all the documents in the training set, and scores the category candidates based the class of k neighbors. The similarity of x and each neighbor's document could be the score of the category of the neighbor documents. Multiple KNN documents may belong to the same category; in this case, the summation of these scores would be the similarity score of class k with respect to the test document x. After sorting the score values, the algorithm assigns the candidate to the class with the highest score from the test document x [137]. Figure 11 illustrates KNN architecture, but for simplicity, this figure is designed by a 2D data set (similar and with higher dimensional space like the text data set). The decision rule of KNN is:

$$\begin{aligned} f(x) &= arg \max_j S(x, C_j) \\ &= \sum_{d_i \in KNN} sim(x, d_i) y(d_i, C_j) \end{aligned} \tag{102}$$

where S refers to score value with respect to $S(x, C_j)$, the score value of candidate i to class of j, and output of $f(x)$ is a label to the test set document.

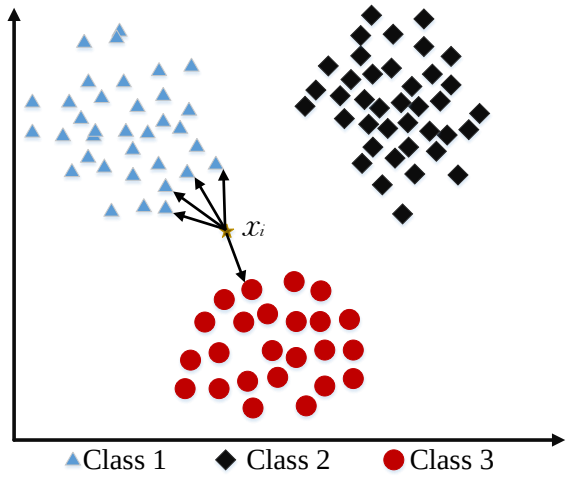

Figure 11. A architecture of k-nearest Neighbor (KNN) model for the 2D data set and three classes.

4.5.2. Weight Adjusted K-Nearest Neighbor Classification

The weight adjusted k-nearest neighbor classification (WAKNN) is a version of KNN which tries to learn the weight vectors for classification [138]. The weighted cosine measure [139] is calculated as follows:

$$\cos(x, y, w) \frac{\sum\limits_{t \in T} (x_t \times w_t) \times (y_t \times w_t)}{\sqrt{\sum\limits_{t \in T} (x_t \times w_t)^2} \times \sqrt{\sum\limits_{t \in T} (y_t \times w_t)^2}} \tag{103}$$

where T refers to the set of words, and x_t and y_t are TF, as discussed in Section 2. For the training model ($d \in D$), let $N_d = \{n_1, n_2, \ldots, n_k\}$ be the set of k-nearest Neighbors of d. Given N_d, the similarity sum of d neighbors that belong to class c is defined as follows:

$$S_c = \sum_{n_i \in N; C(n_i) = c} \cos(d, n_i, w) \tag{104}$$

Total similarity is calculated as follows:

$$T = \sum_{c \in C} S_c \tag{105}$$

The contribution of d is defined in terms of S_c of classes c and T as follows:

$$cont(d) = \begin{cases} 1 & \text{if } \forall c \in C, c \neq class(d), \\ & S_{class(d)} > S_s \text{ and } \frac{S_{class(d)}}{T} \leq p \\ 0 & \text{otherwise} \end{cases} \tag{106}$$

where $cont(d)$ stands for $contribution(d)$

4.5.3. Limitation of K-Nearest Neighbor

KNN is a classification method that is easy to implement and adapts to any kind of feature space. This model also naturally handles multi-class cases [140,141]. However, KNN is limited by data storage constraints for large search problems to find nearest neighbors. Additionally, the performance of KNN is dependent on finding a meaningful distance function, thus making this technique a very data dependent algorithm [142,143].

4.6. Support Vector Machine (SVM)

The original version of SVM was developed by Vapnik and Chervonenkis [144] in 1963. B.E. Boser et al. [145] adapted this version into a nonlinear formulation in the early 1990s. SVM was originally designed for binary classification tasks. However, many researchers work on multi-class problems using this dominate technique [146]. The Figure 12 indicates the linear and non-linear classifier which is used for $2 - dimension$ datasets.

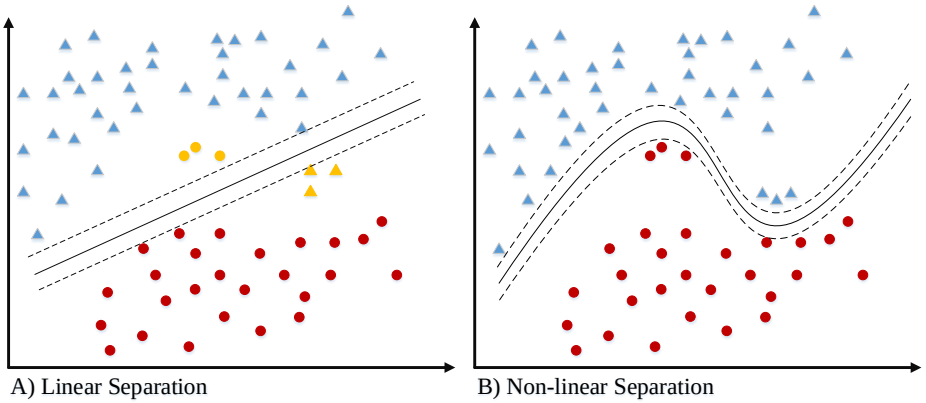

Figure 12. This figure shows the linear and non-linear Support Vector Machine (SVM) for a 2D data set (for text data we have thousands of dimensions). The red is class 1, the blue color is class 2 and yellow color is miss-classified data points.

4.6.1. Binary-Class SVM

In the context of text classification, let $x_1, x_2, ..., x_l$ be training examples belonging to one class X, where X is a compact subset of R^N [21]. Then we can formulate a binary classifier as follows:

$$\min \frac{1}{2}||w||^2 + \frac{1}{vl}\sum_{i=1}^{l}\xi_i - p \tag{107}$$

subject to:

$$(w.\Phi(x_i)) \geq p - \xi_i \;\; i = 1, 2, \ldots, l \;\; \xi \geq 0 \tag{108}$$

If w and p solve this problem, then the decision function is given by:

$$f(x) = sign((w.\Phi(x)) - p) \tag{109}$$

4.6.2. Multi-Class SVM

Since SVMs are traditionally used for the binary classification, we need to generate a Multiple-SVM (MSVM) [147] for multi-class problems. One-vs-One is a technique for multi-class SVM that builds $N(N-1)$ classifiers as follows:

$$f(x) = arg\max_i \left(\sum_j f_{ij}(x)\right) \tag{110}$$

The natural way to solve the k-class problem is to construct a decision function of all k classes at once [148,149]. In general, multi-class SVM is an optimization problem of the following form:

$$\min_{w_1, w_2, .., w_k, \zeta} \frac{1}{2}\sum_k w_k^T w_k + C \sum_{(x_i, y_i) \in D} \zeta_i \tag{111}$$

$$st. \;\; w_{y_i}^T x - w_k^T x \leq i - \zeta_i, \quad \forall (x_i, y_i) \in D, k \in \{1, 2, ..., K\}, k \neq y_i \tag{112}$$

where (x_i, y_i) represent the training data points such that $(x_i, y_i) \in D$, C is the penalty parameter, ζ is a slack parameter, and k stands for the class.

Another technique of multi-class classification using SVM is All-vs-One. Feature extraction via SVM generally uses one of two methods: Word sequences feature extracting [150] and TF-IDF. But for

an unstructured sequence such as RNA and DNA sequences, string kernel is used. However, string kernel can be used for a document categorization [151].

4.6.3. String Kernel

Text classification has also been studied using string kernel [151]. The basic idea of string kernel (SK) is using $\Phi(.)$ to map the string in the feature space.

Spectrum kernel as part of SK has been applied to many different applications, including text, DNA, and protein classification [152,153]. The basic idea of Spectrum kernel is counting the number of times a word appears in string x_i as a feature map where defining feature maps from $x \rightarrow R^{l^k}$.

$$\Phi_k(x) = \Phi_j(x)_{j \in \Sigma^k} \tag{113}$$

where

$$\Phi_j(x) = number\ of\ j\ \ feature\ appears\ in\ x \tag{114}$$

The feature map $\Phi_i(x)$ is generated by the sequence x_i and kernel defines as follows:

$$F = \Sigma^k \tag{115}$$

$$K_i(x, x') = < \Phi_i(x), \Phi_i(x') > \tag{116}$$

The main limitation of SVM when applied to string sequence classification is time complexity [154]. The features are generated using dictionary size Σ and F is the number of features and bounded by Equation (115). The kernel calculation is similar with SP and uses Equation (116), and finally normalizes the kernel using Equation (117).

$$K^{Norm}(x, y) \leftarrow \frac{K(x, y)}{\sqrt{K(x, x)}\sqrt{K(y, y)}} \tag{117}$$

$$< f^x, f^y > = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} h(u_i^{s_1}, u_j^{s_2}) \tag{118}$$

where two sequences, $u_i^{s_1}$ and $u_j^{s_2}$, are lengths of s_1 and s_2 respectively.

4.6.4. Stacking Support Vector Machine (SVM)

Stacking SVM is a hierarchical classification method used for category tree structure based on a top-down level-based approach [155]. This technique provides a hierarchical model of individual SVM classifiers, and thus generally produces more accurate results than single-SVM models [156]. As shown in the Figure 13, the stacking model employs hierarchical classifier which contains several layers (in this Figure we have two level like mane domain, and sub-domains).

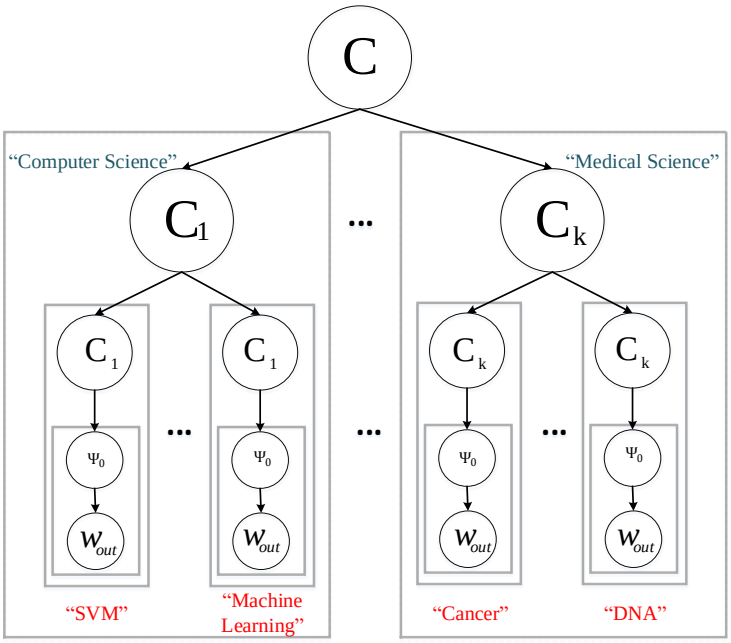

Figure 13. Hierarchical classification method.

4.6.5. Multiple Instance Learning (MIL)

Multiple instance learning (MIL) is a supervised learning method [157] and is typically formulated as one of two SVM-based methods (mi-SVM and MI-SVM) [158]. MIL takes in a set of labeled bags as input instead of instances. A bag is labeled positive if there is at least one instance in it with a positive label, and labeled negative if all instances of it are negative. Then, the learner tries to infer a concept that label individual instances correctly [157]. In statistical pattern recognition, it is assumed that a training set of labeled patterns is available where each pair $(x_i, y_i) \in R^d \times Y$ has been generated from an unknown distribution independently. The goal is to find a classifier from patterns to labels i.e., $f : R^d \to Y$. In MIL, the algorithm assumes the input is available as a set of input patterns $x_1, ..., x_n$ grouped into bags $B_1, ..., B_m$ where $B_I = \{x_i : i \in I\}$ for the given index sets $I \subseteq \{1, ..., n\}$. Each bag B_I is associated with label Y_I where $Y_I = -1$ if $y_i = -1$ for all $i \in I$ and $Y_I = 1$ if there is at least one instance $x_i \in B_I$ with positive label [158]. The relationship between instance labels y_i and bag labels Y_I can be expressed as $Y_I = \max_{i \in I} y_i$ or a set of linear constraints:

$$\begin{aligned} &\sum_{i \in I} \frac{y_i + 1}{2} \geq 1, \\ &\quad \forall I \; s.t. \; Y_I = 1, \\ &\quad y_i = -1, \forall I \; s.t. \; Y_I = -1. \end{aligned} \tag{119}$$

The discriminant function $f : X \to R$ is called MI-separating with respect to a multiple-instance data set if $sgn \max_{i \in I} \; f(x_i) = Y_I$ for all bags B_I holds.

4.6.6. Limitation of Support Vector Machine (SVM)

SVM has been one of the most efficient machine learning algorithms since its introduction in the 1990s [159]. However, the SVM algorithms for text classification are limited by the lack of transparency in results caused by a high number of dimensions. Due to this, it cannot show the company score as a parametric function based on financial ratios nor any other functional form [159]. A further limitation is a variable financial ratios rate [160].

4.7. Decision Tree

One earlier classification algorithm for text and data mining is decision tree [161]. Decision tree classifiers (DTCs) are used successfully in many diverse areas for classification [162]. The structure of this technique is a hierarchical decomposition of the data space [7,161]. Decision tree as classification task was introduced by D. Morgan [163] and developed by J.R. Quinlan [164]. The main idea is creating a tree based on the attribute for categorized data points, but the main challenge of a decision tree is which attribute or feature could be in parents' level and which one should be in child level. To solve this problem, De Mántaras [165] introduced statistical modeling for feature selection in tree. For a training set containing p positive and n negative:

$$\begin{aligned} H\left(\frac{p}{n+p}, \frac{n}{n+p}\right) = &- \frac{p}{n+p} \log_2 \frac{p}{n+p} \\ &- \frac{n}{n+p} \log_2 \frac{n}{n+p} \end{aligned} \tag{120}$$

Choose attribute A with k distinct value, divides the training set E into subsets of $\{E_1, E_2, \ldots, E_k\}$. The expect entropy (EH) remain after trying attribute A (with branches $i = 1, 2, \ldots, k$):

$$EH(A) = \sum_{i=1}^{K} \frac{p_i + n_i}{p+n} H\left(\frac{p_i}{n_i + p_i}, \frac{n_i}{n_i + p_i}\right) \tag{121}$$

Information gain (I) or reduction in entropy for this attribute is :

$$A(I) = H\left(\frac{p}{n+p}, \frac{n}{n+p}\right) - EH(A) \tag{122}$$

Choose the attribute with largest information gain as parent's node.

Limitation of Decision Tree Algorithm

The decision tree is a very fast algorithm for both learning and prediction; but it is also extremely sensitive to small perturbations in the data [166], and can be easily overfit [167]. These effects can be negated by validation methods and pruning, but this is a grey area [166]. This model also has problems with out-of-sample prediction [168].

4.8. Random Forest

Random forests or random decision forests technique is an ensemble learning method for text classification. This method, which used t tree as parallel, was introduced by T. Kam Ho [169] in 1995. As shown in Figure 14, the main idea of RF is generating random decision trees. This technique was further developed in 1999 by L. Breiman [170], who found convergence for RF as margin measures ($mg(X, Y)$) as follows:

$$\begin{aligned} mg(X, Y) = &av_k I(h_k(X) = Y) - \\ &\max_{j \neq Y} av_k I(h_k(X) = j) \end{aligned} \tag{123}$$

where $I(.)$ refers to indicator function.

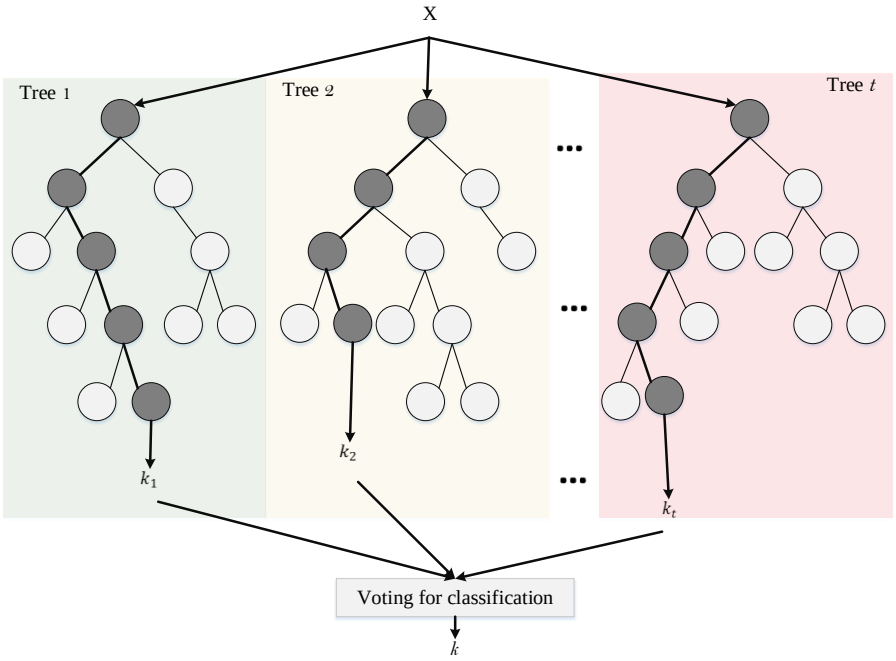

Figure 14. Random forest.

4.8.1. Voting

After training all trees as forest, predictions are assigned based on voting [171] as follows:

$$\delta_V = arg\max_i \sum_{j:j\neq j} I_{\{r_{ij}>r_{ji}\}} \tag{124}$$

such that

$$r_{ij} + r_{ji} = 1 \tag{125}$$

4.8.2. Limitation of Random Forests

Random forests (i.e., ensembles of decision trees) are very fast to train for text data sets in comparison to other techniques such as deep learning, but quite slow to create predictions once trained [172]. Thus, in order to achieve a faster structure, the number of trees in forest must be reduced, as more trees in forest increases time complexity in the prediction step.

4.9. Conditional Random Field (CRF)

CRF is an undirected graphical model, as shown in Figure 15. CRFs are essentially a way of combining the advantages of classification and graphical modeling that combine the ability to compactly model multivariate data, and the ability to leverage a high dimensional features space for prediction [173] (this model is very powerful for text data due to high feature space). CRFs state the conditional probability of a label sequence Y given a sequence of observation X i.e., $P(Y|X)$. CRFs can incorporate complex features into an observation sequence without violating the independence assumption by modeling the conditional probability of the label sequence rather than the joint probability $P(X, Y)$ [174,175]. Clique (i.e., fully connected subgraph) potential is used for computing $P(X|Y)$. With respect to the potential function for each clique in the graph, the probability of a variable configuration corresponds to the product of a series of a non-negative potential functions.

The value computed by each potential function is equivalent to the probability of the variables in the corresponding clique for a particular configuration [174]. That is:

$$P(V) = \frac{1}{Z} \prod_{c \in cliques(V)} \psi(c) \tag{126}$$

where Z is the normalization term. The conditional probability $P(X|Y)$ can be formulated as:

$$P(Y|X) = \frac{1}{Z} \prod_{t=1}^{T} \psi(t, y_{t-1}, y_t, X) \tag{127}$$

Given the potential function ($\psi(t, y_{t-1}, y_t, X) = \exp(w.f(t, y_{t-1}, y_t, X))$), the conditional probability can be rewritten as:

$$P(Y|X) = \prod_{t=1}^{T} \exp(w.f(t, y_{t-1}, y_t, X)) \tag{128}$$

where w is the weight vector associated with a feature vector computed by f.

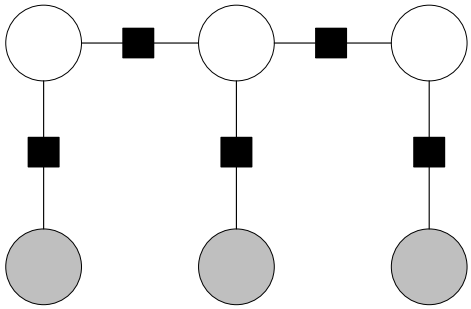

Figure 15. Linear-chain conditional random field (CRF). The black boxes are transition clique

Limitation of Conditional Random Field (CRF)

With regards to CRF, the most evident disadvantage of CRF is the high computational complexity of the training step [176], especially for text data sets due to high feature space. Furthermore, this algorithm does not perform with unseen words (i.e., with words that were not present in the training data sample) [177].

4.10. Deep Learning

Deep learning models have achieved state-of-the-art results across many domains, including a wide variety of NLP applications. Deep learning for text and document classification includes three basic architectures of deep learning in parallel. We describe each individual model in detail below.

4.10.1. Deep Neural Networks

Deep neural networks (DNN) are designed to learn by multi-connection of layers that every single layer only receives the connection from previous and provides connections only to the next layer in a hidden part [2]. Figure 16 depicts the structure of a standard DNN. The input consists of the connection of the input feature space (as discussed in Section 2) with the first hidden layer of the DNN. The input layer may be constructed via TF-IDF, word embedding, or some other feature extraction method. The output layer is equal to the number of classes for multi-class classification or only one for binary classification. In multi-class DNNs, each learning model is generated (number of nodes in each layer and the number of layers are completely randomly assigned). The implementation

of DNN is a discriminative trained model that uses a standard back-propagation algorithm using sigmoid (Equation (129)), ReLU [178] (Equation (130)) as an activation function. The output layer for multi-class classification should be a $Softmax$ function (as shown in Equation (131)).

$$f(x) = \frac{1}{1+e^{-x}} \in (0,1) \tag{129}$$

$$f(x) = \max(0,x) \tag{130}$$

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \forall\, j \in \{1,\ldots,K\} \tag{131}$$

Given a set of example pairs (x,y),$x \in X, y \in Y$, the goal is to learn the relationship between these input and target spaces using hidden layers. In text classification applications, the input is a string which is generated via vectorization of the raw text data.

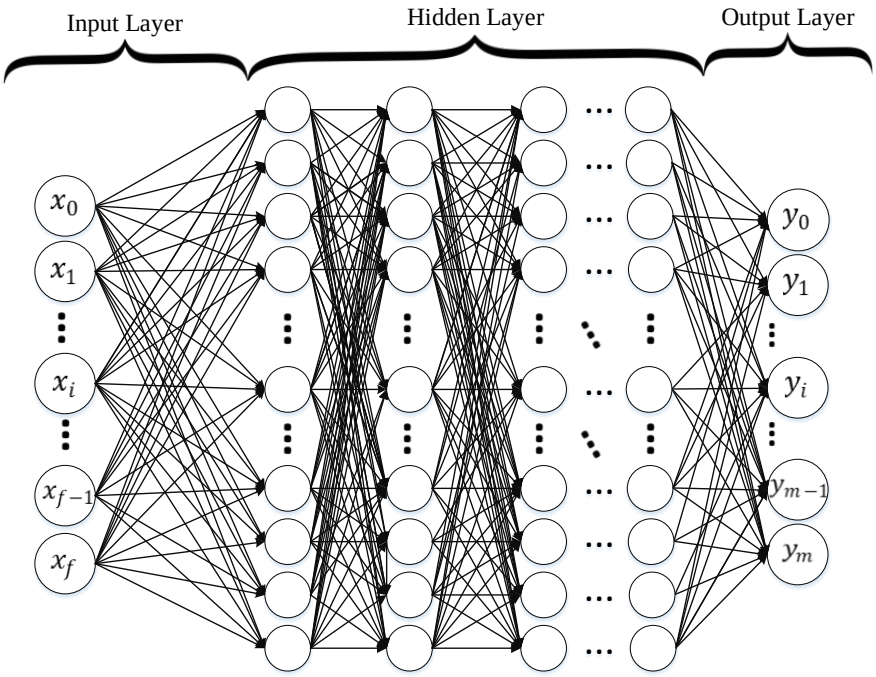

Figure 16. Standard, fully connected deep neural network (DNN).

4.10.2. Recurrent Neural Network (RNN)

Another neural network architecture that researchers have used for text mining and classification is recurrent neural network (RNN) [179,180]. RNN assigns more weights to the previous data points of a sequence. Therefore, this technique is a powerful method for text, string, and sequential data classification. A RNN considers the information of previous nodes in a very sophisticated method which allows for better semantic analysis of a data set's structure. RNN mostly works by using LSTM or GRU for text classification, as shown in Figure 17 which contains input layer (word embedding), hidden layers, and finally output layer. This method can be formulated as:

$$x_t = F(x_{t-1}, \boldsymbol{u_t}, \theta) \tag{132}$$

where x_t is the state at time t and $\boldsymbol{u_t}$ refers to the input at step t. More specifically, we can use weights to formulate Equation (132), parameterized by:

$$x_t = \mathbf{W_{rec}}\sigma(x_{t-1}) + \mathbf{W_{in}}\mathbf{u_t} + \mathbf{b} \tag{133}$$

where $\mathbf{W_{rec}}$ refers to recurrent matrix weight, $\mathbf{W_{in}}$ refers to input weights, $\mathbf{b}$ is the bias, and σ denotes an element-wise function.

Figure 17 illustrates an extended RNN architecture. Despite the benefits described above, RNN is vulnerable to the problems of vanishing gradient and exploding gradient when the error of the gradient descent algorithm is back propagated through the network [181].

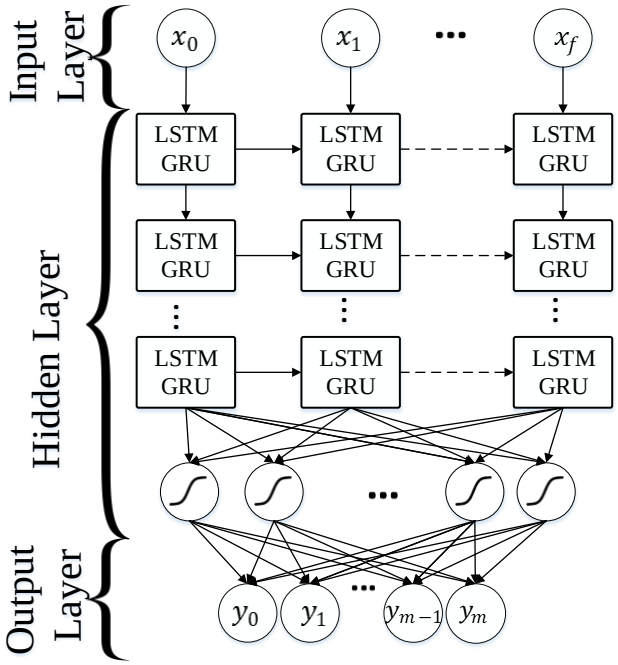

Figure 17. Standard long short-term memory (LSTM)/GRU recurrent neural networks.

Long Short-Term Memory (LSTM)

LSTM was introduced by S. Hochreiter and J. Schmidhuber [182], and has since been augmented by many research scientists [183].

LSTM is a special type of RNN that addresses these problems by preserving long term dependency in a more effective way in comparison to the basic RNN. LSTM is particularly useful with respect to overcoming the vanishing gradient problem [184]. Although LSTM has a chain-like structure similar to RNN, LSTM uses multiple gates to carefully regulate the amount of information that is allowed into each node state. Figure 18 shows the basic cell of an LSTM model. A step-by-step explanation of a LSTM cell is as follows:

$$i_t = \sigma(W_i[x_t, h_{t-1}] + b_i), \tag{134}$$
$$\tilde{C}_t = \tanh(W_c[x_t, h_{t-1}] + b_c), \tag{135}$$
$$f_t = \sigma(W_f[x_t, h_{t-1}] + b_f), \tag{136}$$
$$C_t = i_t * \tilde{C}_t + f_t C_{t-1}, \tag{137}$$
$$o_t = \sigma(W_o[x_t, h_{t-1}] + b_o), \tag{138}$$
$$h_t = o_t \tanh(C_t), \tag{139}$$

where Equation (134) represents the input gate, Equation (135) represents the candid memory cell value, Equation (136) defines forget-gate activation, Equation (137) calculates the new memory cell value, and Equations (138) and (139) define the final output gate value. In the above description, each b

represents a bias vector, each W represent a weight matrix, and x_t represents input to the memory cell at time t. Furthermore, i,c,f,o indices refer to input, cell memory, forget and output gates respectively. Figure 18 shows a graphical representation of the structure of these gates.

A RNN can be biased when later words are more influential than earlier ones. Convolutional neural network (CNN) models (discussed in Section 4.10.3) were introduced to overcome this bias by deploying a max-pooling layer to determine discriminative phrases in text data [6].

Gated Recurrent Unit (GRU)

GRUs are a gating mechanism for RNN formulated by J. Chung et al. [185] and K. Cho et al. [101]. GRUs are a simplified variant of the LSTM architecture. However, a GRU differs from LSTM because it contains two gates and a GRU does not possess internal memory (i.e., the C_{t-1} in Figure 18). Furthermore, a second non-linearity is not applied (i.e., tanh in Figure 18). A step-by-step explanation of a GRU cell is as follows:

$$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z), \tag{140}$$

where z_t refers to the update gate vector of t, x_t stands for input vector, W, U, and b represent parameter matrices/vectors. The activation function (σ_g) is either a sigmoid or ReLU and can be formulated as follow:

$$\tilde{r_t} = \sigma_g(W_r x_t + U_r h_{t-1} + b_r), \tag{141}$$

where r_t stands for reset gate vector of t, z_t is update gate vector of t.

$$\begin{aligned} h_t =& z_t \circ h_{t-1} + (1 - z_t) \circ \\ & \sigma_h(W_h x_t + U_h(r_t \circ h_{t-1}) + b_h) \end{aligned} \tag{142}$$

where h_t is output vector of t, and σ_h indicates the hyperbolic tangent function.

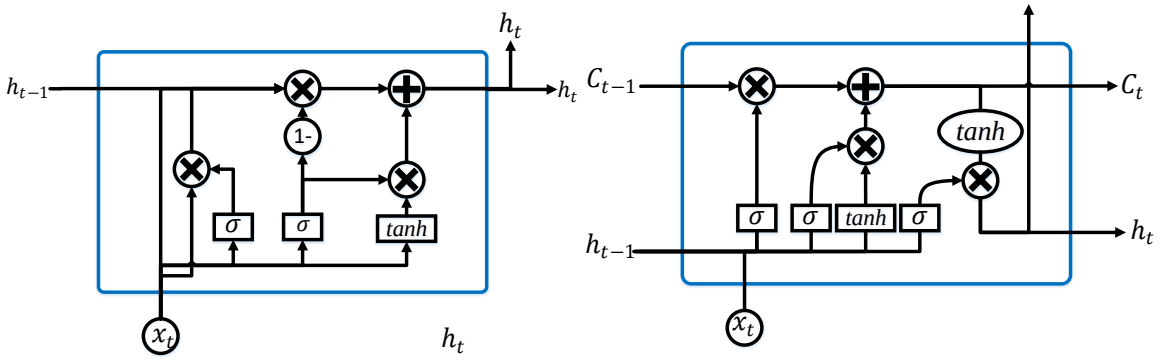

Figure 18. The left figure is a GRU cell while the right figure is a LSTM cell.

4.10.3. Convolutional Neural Networks (CNN)

A convolutional neural network (CNN) is a deep learning architecture that is commonly used for hierarchical document classification [6,186]. Although originally built for image processing, CNNs have also been effectively used for text classification [27,187]. In a basic CNN for image processing, an image tensor is convolved with a set of kernels of size $d \times d$. These convolution layers are called feature maps and can be stacked to provide multiple filters on the input. To reduce the computational complexity, CNNs use pooling to reduce the size of the output from one layer to the next in the network. Different pooling techniques are used to reduce outputs while preserving important features [188].

The most common pooling method is max pooling where the maximum element in the pooling window is selected. In order to feed the pooled output from stacked featured maps to the next layer, the maps are flattened into one column. The final layers in a CNN are typically fully connected.

In general, during the back-propagation step of a convolutional neural network, both the weights and the feature detector filters are adjusted. A potential problem that arises when using CNN for text classification is the number of 'channels', Σ (size of the feature space). While image classification application generally have few channels (e.g., only 3 channels of RGB), Σ may be very large (e.g., 50 K) for text classification applications [189], thus resulting in very high dimensionality. Figure 19 illustrate the CNN architecture for text classification which contains word embedding as input layer 1D convolutional layers, 1D pooling layer, fully connected layers, and finally output layer.

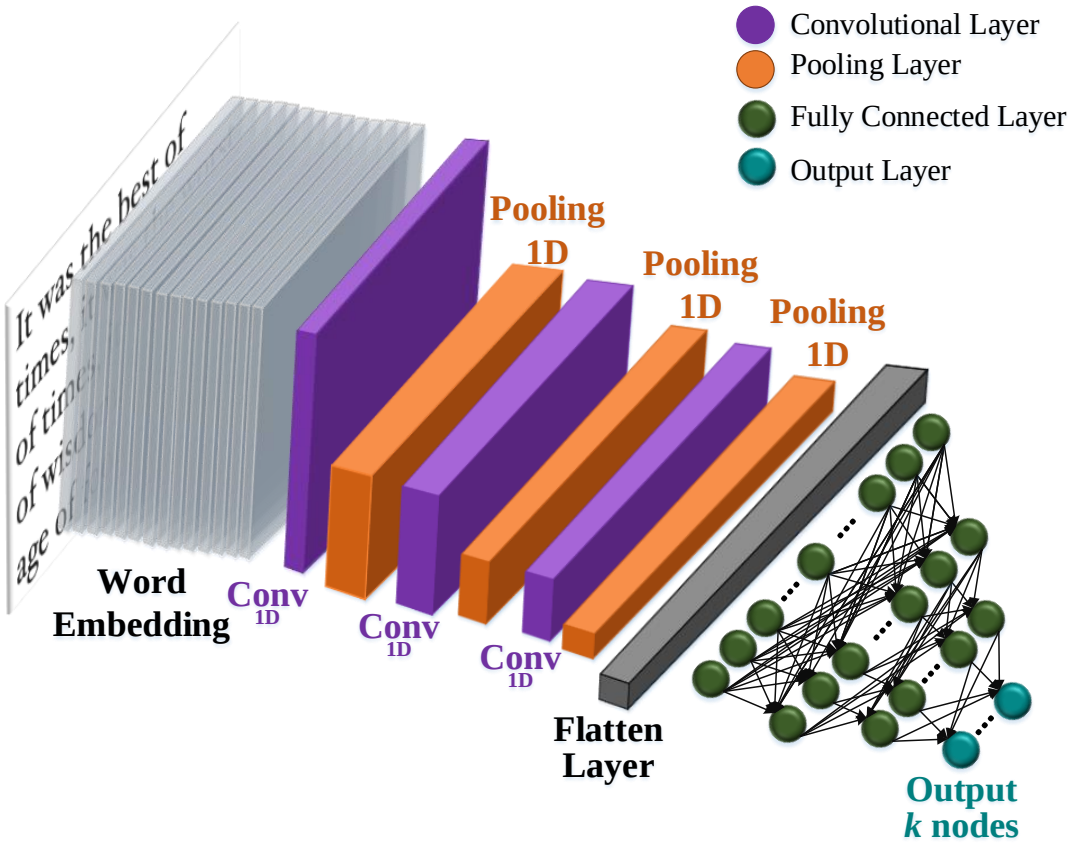

Figure 19. Convolutional neural network (CNN) architecture for text classification.

4.10.4. Deep Belief Network (DBN)

A deep belief network (DBN) is a deep learning structure that is superposed by restricted Boltzmann machines (RBMs) [1]. A RBM is a generative artificial neural network which could learn probability distribution over samples. Contrastive divergence (CD) [190] is a training technique used for RBMs [191,192].

The energy function is as follows:

$$E(v,h) = -\sum_i a_i v_i - \sum_j b_j h_j - \sum_i \sum_j v_i w_{i,j} h_j \tag{143}$$

where a_i is visible units and b_i refers to hidden units in matrix notation. This expression can be simplified as:

$$E(v,h) = -a^{\mathrm{T}} v - b^{\mathrm{T}} h - v^{\mathrm{T}} W h \tag{144}$$

Given a configuration of the hidden units h is defined as follows:

$$P(v|h) = \prod_{i=1}^{m} P(v_i|h) \tag{145}$$

For Bernoulli, the logistic function for visible units is replaced as follows:

$$P(v_i^k = 1|h) = \frac{\exp(a_i^k + \Sigma_j W_{ij}^k h_j)}{\Sigma_{k'=1}^{K} \exp(a_i^{k'} + \Sigma_j W_{ij}^{k'} h_j)} \tag{146}$$

The update function with gradient descent is as follows:

$$w_{ij}(t+1) = w_{ij}(t) + \eta \frac{\partial \log(p(v))}{\partial w_{ij}} \tag{147}$$

4.10.5. Hierarchical Attention Networks (HAN)

One of the successful deep architecture for text and document classification is hierarchical attention networks (HAN). This technique was introduced by Z. Yang et al. [193] and S.P. Hongsuck et al. [194]. The structure of a HAN focuses on the document-level classification which a document has L sentences and each sentence contains T_i words, where w_{it} with $t \int [1, T]$ represents the words in the ith sentence. HAN architecture is illustrated in Figure 20, where the lower level contains word encoding and word attention and the upper level contains sentence encoding and sentence attention.

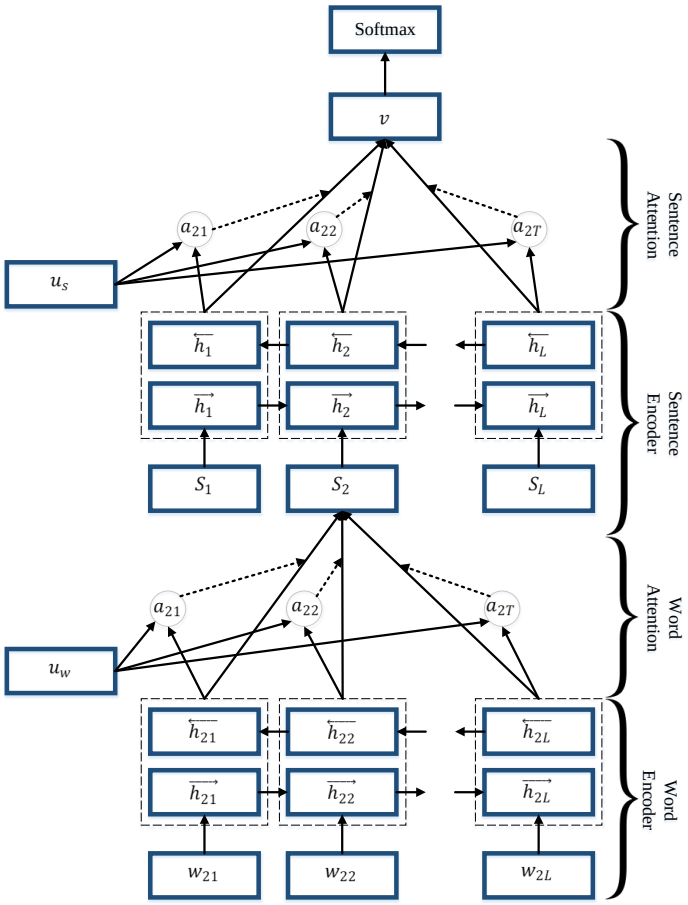

Figure 20. Hierarchical attention networks for document classification.

4.10.6. Combination Techniques

Many researchers combine or concatenate standard deep learning architectures in order to develop novel techniques with more robust and accurate architectures for classification tasks. In this sub-section, we describe recent and popular deep learning architectures and structure.

Random Multimodel Deep Learning (RMDL)

Random multimodel deep learning (RMDL) was introduced by K. Kowsari et al. [4,5] as a novel deep learning technique for classification. RMDL can be used in any kind of data set for classification. An overview of this technique is shown in Figure 21 which illustrates the architecture using multi-DNN, deep CNN, and deep RNN. The number of layers and nodes for all of these deep learning multi-models are generated randomly (e.g., 9 random models in RMDL constructed from 3 CNNs, 3 RNNs, and 3 DNNs, all of which are unique due to random creation).

$$M(y_{i1}, y_{i2}, ..., y_{in}) = \left\lfloor \frac{1}{2} + \frac{(\sum_{j=1}^{n} y_{ij}) - \frac{1}{2}}{n} \right\rfloor \tag{148}$$

where n is the number of random models, and y_{ij} is the output prediction of model for data point i in model j (Equation (148) is used for binary classification, $k \in \{0 \text{ or } 1\}$). The output space uses majority vote to calculate the final value of $\hat{y}_i$. Therefore, $\hat{y}_i$ is given as follows:

$$\hat{y}_i = \left[\hat{y}_{i1} \ \dots \ \hat{y}_{ij} \ \dots \ \hat{y}_{in} \right]^T \tag{149}$$

where n is the number of the random model, and $\hat{y}_{ij}$ shows the prediction of the label of data point (e.g., document) of $D_i \in \{x_i, y_i\}$ for model j and $\hat{y}_{i,j}$ is defined as follows:

$$\hat{y}_{i,j} = arg\max_{k}[softmax(y^*_{i,j})] \tag{150}$$

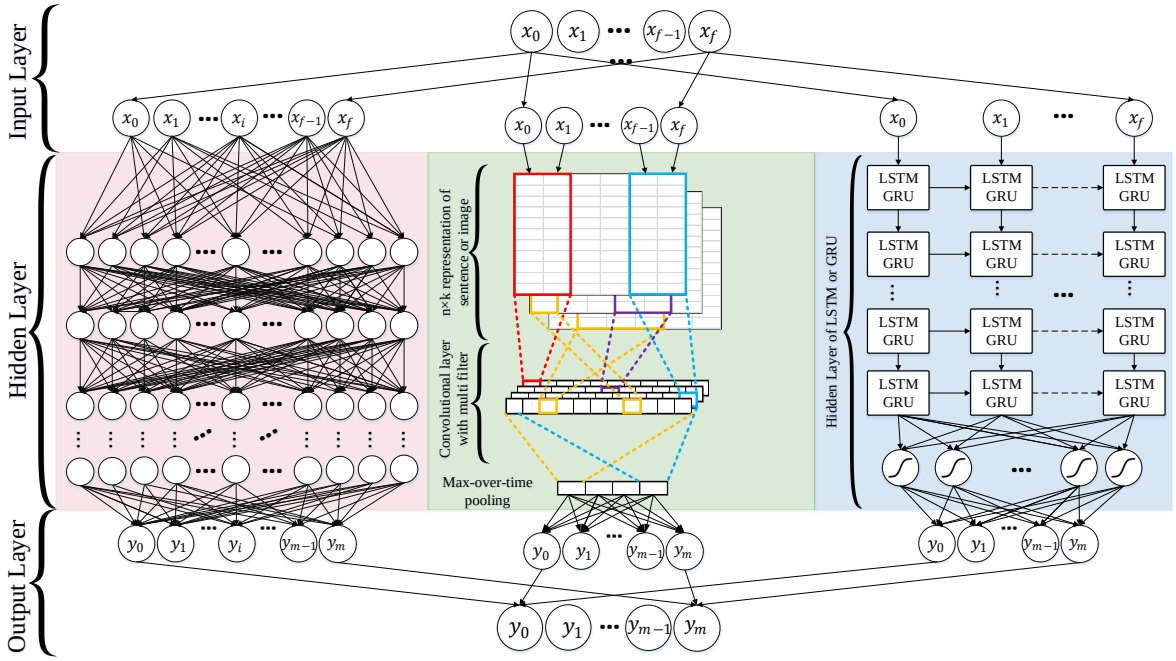

Figure 21. Random multimodel eeep learning (RDML) architecture for classification. RMDL includes 3 random models: A deep neural network (DNN) classifier (**left**), a deep CNN classifier (**middle**), and a deep recurrent neural network (RNN) classifier (**right**). Each unit could be a LSTM or GRU).

After all the RDL models (RMDL) are trained, the final prediction is calculated using majority vote on the output of these models. The main idea of using multi-model with different optimizers is that if one optimizer does not provide a good fit for a specific data set, the RMDL model with n random models (where some of them might use different optimizers) could ignore k models which are not efficient if and only if $n > k$. Using multi-techniques of optimizers (e.g., SGD, Adam, RMSProp, Adagrad, Adamax) helps the RMDL model be more suitable for any type of data sets. While we only used 2 optimizers (Adam and RMSProp) for evaluating the model in this research, the RMDL model can use any kind of optimizer. In this part, we describe common optimization techniques used in deep learning architectures.

Stochastic Gradient Descent (SGD) Optimizer:

The basic equation for stochastic gradient descent (SGD) [195] is shown in Equation (151). SGD uses a momentum on re-scaled gradient which is shown in Equation (152) for updating parameters.

$$\theta \leftarrow \theta - \alpha \nabla_\theta J(\theta, x_i, y_i) \tag{151}$$

$$\theta \leftarrow \theta - \left(\gamma\theta + \alpha \nabla_\theta J(\theta, x_i, y_i)\right) \tag{152}$$

RMSprop:

T. Tieleman and G. Hinton [196] introduced RMSprop as a novel optimizer which divides the learning rate for a weight by a running average of the magnitudes of recent gradients for that weight. The equation of the momentum method for RMSprop is as follows:

$$v(t) = \alpha\, v(t-1) - \epsilon \frac{\partial E}{\partial w}(t) \tag{153}$$

$$\begin{aligned} \Delta w(t) &= v(t) \\ &= \alpha\, v(t-1) - \epsilon \frac{\partial E}{\partial w}(t) \\ &= \alpha\, \Delta v(t-1) - \epsilon \frac{\partial E}{\partial w}(t) \end{aligned} \tag{154}$$

RMSProp does not do bias correction, which causes significant problems when dealing with a sparse gradient.

Adam Optimizer

Adam is another stochastic gradient optimizer which uses only the first two moments of gradient (v and m, shown in Equations (155)–(158)) and calculate the average over them. It can handle non-stationary of the objective function as in RMSProp while overcoming the sparse gradient issue limitation of RMSProp [197].

$$\theta \leftarrow \theta - \frac{\alpha}{\sqrt{\hat{v}} + \epsilon} \hat{m} \tag{155}$$

$$g_{i,t} = \nabla_\theta J(\theta_i, x_i, y_i) \tag{156}$$

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_{i,t} \tag{157}$$

$$m_t = \beta_2 v_{t-1} + (1 - \beta_2) g_{i,t}^2 \tag{158}$$

where m_t is the first moment and v_t indicates second moment that both are estimated. $\hat{m}_t = \frac{m_t}{1-\beta_1^t}$ and $\hat{v}_t = \frac{v_t}{1-\beta_2^t}$.

Adagrad:

Adagrad is addressed in [198] as a novel family of sub-gradient methods which dynamically absorb knowledge of the geometry of the data to perform more informative gradient-based learning.

AdaGrad is an extension of SGD. In iteration k, the gradient is defined as:

$$G^{(k)} = diag\left[\sum_{i=1}^{k} g^{(i)}(g^{(i)})^T\right]^{1/2} \tag{159}$$

diagonal matrix:

$$G_{jj}^{(k)} = \sqrt{\sum_{i=1}^{k}(g_i^{(i)})^2} \tag{160}$$

update rule:

$$\begin{aligned} x^{(k+1)} &= arg\min_{x\in X}\{\langle \nabla f(x^{(k)}), x\rangle + \\ &\qquad \frac{1}{2\alpha_k}||x - x^{(k)}||^2_{G^{(k)}}\} \\ &= x^{(k)} - \alpha B^{-1}\nabla f(x^{(k)}) \qquad (\text{if } X = R^n) \end{aligned} \tag{161}$$

Adadelta:

AdaDelta, introduced by M.D. Zeiler [199], uses the exponentially decaying average of g_t as 2_{nd} moment of gradient. This method is an updated version of Adagrad which relies on only first order information. The update rule for Adadelta is:

$$g_{t+1} = \gamma g_t + (1-\gamma)\nabla\mathcal{L}(\theta)^2 \tag{162}$$

$$x_{t+1} = \gamma x_t + (1-\gamma)v_{t+1}^2 \tag{163}$$

$$v_{t+1} = -\frac{\sqrt{x_t + \epsilon}\delta L(\theta_t)}{\sqrt{g_{t+1} + \epsilon}} \tag{164}$$

Hierarchical Deep Learning for Text (HDLTex)

The primary contribution of the hierarchical deep learning for text (HDLTex) architecture is hierarchical classification of documents [2]. A traditional multi-class classification technique can work well for a limited number of classes, but performance drops with an increasing number of classes, as is present in hierarchically organized documents. In this hierarchical deep learning model, this problem was solved by creating architectures that specialize deep learning approaches for their level of the document hierarchy (e.g., see Figure 22). The structure of the HDLTex architecture for each deep learning model is as follows:

DNN: 8 hidden layers with 1024 cells in each hidden layer.
RNN: GRU and LSTM are used in this implementation, 100 cells with GRU with two hidden layers.
CNN: Filter sizes of $\{3, 4, 5, 6, 7\}$ and max-pool of 5, layer sizes of $\{128, 128, 128\}$ with max pooling of $\{5, 5, 35\}$, the CNN contains 8 hidden layers.

All models were constructed using the following parameters: *Batch Size = 128*, *learning parameters* = 0.001, β_1 = *0.9*, β_2 = *0.999*, $\epsilon = 1e^{08}$, *decay* = 0.0, *Dropout = 0.5 (DNN), and Dropout = 0.25 (CNN and RNN).*

HDLTex uses the following cost function for the deep learning models evaluation:

$$Acc(X) = \sum_{\varrho} \left[\frac{Acc(X_{\Psi_\varrho})}{k_\varrho - 1} \sum_{\Psi \in \{\Psi_1, .. \Psi_k\}} Acc(X_{\Psi_i}).n_{\Psi_k} \right] \tag{165}$$

where ϱ is the number of levels, k indicates number of classes for each level, and Ψ refers to the number of classes in the child's level of the hierarchical model.

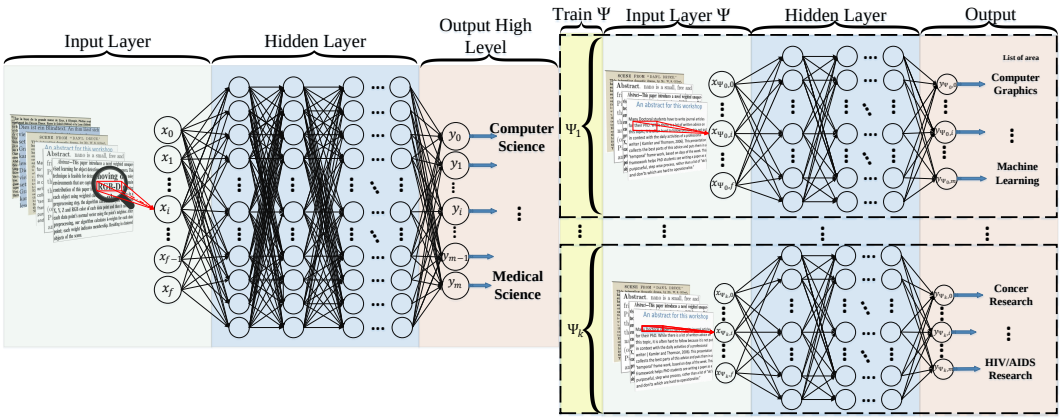

Figure 22. HDLTex: Hierarchical deep learning for text classification. DNN approach for the text classification. The top figure depicts the parent-level of our model, and the bottom figure depicts child-level models (Ψ_i) as input documents in the parent level.

Other Techniques

In this section, we discuss other techniques of text classification that come from combining deep learning architectures. Recurrent convolutional neural networks (RCNN) is used for text classification [6,200]. RCNNs can capture contextual information with the recurrent structure and construct the representation of text using a CNN [6]. This architecture is a combination of RNN and CNN that leverages the advantages of both techniques in a model.

C-LSTM is another technique of text and document classification that was introduced by C. Zhou et al. [201]. C-LSTM combines CNN with LSTM in order to learn phrase-level features using convolutional layers. This architecture feeds sequences of higher level representations into the LSTM to learn long-term dependent.

4.10.7. Limitation of Deep Learning

Model interpretability of deep learning (DL), especially DNN, has always been a limiting factor for use cases requiring explanations of the features involved in modelling and such is the case for many healthcare problems. This problem is due to scientists preferring to use traditional techniques such as linear models, Bayesian Models, SVM, decision trees, etc. for their works. The weights in a neural network are a measure of how strong each connection is between each neuron to find the important feature space. As shown in Figure 23, the more accurate model, the interpretability is lower which means the complex algorithms such as deep learning is hard to understand.

Deep learning (DL) is one of the most powerful techniques in artificial intelligence (AI), and many researchers and scientists focus on deep learning architectures to improve the robustness and computational power of this tool. However, deep learning architectures also have some disadvantages and limitations when applied to classification tasks. One of the main problems of this model is that DL does not facilitate a comprehensive theoretical understanding of learning [202]. A well-known

disadvantage of DL methods is their "black box" nature [203,204]. That is, the method by which DL methods come up with the convolved output is not readily understandable. Another limitation of DL is that it usually requires much more data than traditional machine learning algorithms, which means that this technique cannot be applied to classification tasks over small data sets [205,206]. Additionally, the massive amount of data needed for DL classification algorithms further exacerbates the computational complexity during the training step [207].

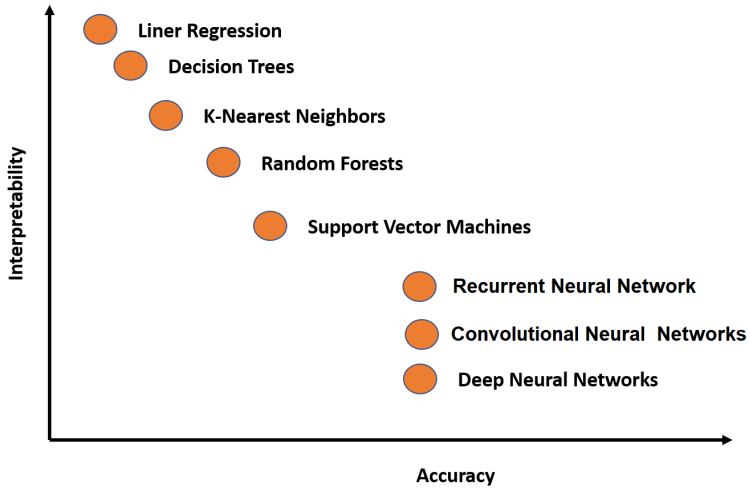

Figure 23. The model interpretability comparison between traditional and deep learning techniques.

4.11. Semi-Supervised Learning for Text Classification

Many researchers have developed many efficient classifiers for both labeled and unlabeled documents. Semi-supervised learning is a type of supervised learning problem that uses unlabeled data to train a model. Usually, researchers and scientists prefer to use semi-supervised techniques when a small part of the data set contains labeled data points and a large amount of data set does not include labels [208]. Most of the semi-supervised learning algorithms for classification tasks use a clustering technique (generally used for unsupervised learning [209]) as follows: Initially a clustering technique is applied on D^T with $K = K$ (the number of classes), since D^T has labeled samples of all K classes [208]. If a partition P_i has labeled samples, then, all data points on that cluster belongs to that label.

The research goal for clustering techniques is to determine if we have more than one class labeled on one cluster, and what happens if we have no labeled data point in one cluster [210]. In this part, we briefly describe the most popular technique of semi-supervised text and document classification. O. Chapelle and A. Zien [211] worked on semi-supervised classification via low density separation, which combines graph distance computation with transductive support vector machine (TSVM) training. K. Nigam et al. [212] developed a technique for text classification using expectation maximization (EM) and generative models for semi-supervised learning with labeled and unlabeled data in the field of text classifications. L. Shi et al. [213] introduced a method for transferring classification knowledge across languages via translated features. This technique uses an EM algorithm that naturally takes into account the ambiguity associated with the translation of a word. J. Su et al. [213] introduced "Semi-supervised Frequency Estimate (SFE)", a MNBC method for large scale text classification. S. Zhou et al. [214] invented a novel deep learning method that uses fuzzy DBN for semi-supervised sentiment classification. This method employs a fuzzy membership function for each category of reviews based on the learned architecture.

5. Evaluation

In the research community, having shared and comparable performance measures to evaluate algorithms is preferable. However, in reality such measures may only exist for a handful of methods. The major problem when evaluating text classification methods is the absence of standard data collection protocols. Even if a common collection method existed (e.g., Reuters news corpus), simply choosing different training and test sets can introduce inconsistencies in model performance [215]. Another challenge with respect to method evaluation is being able to compare different performance measures used in separate experiments. Performance measures generally evaluate specific aspects of classification task performance, and thus do not always present identical information. In this section, we discuss evaluation metrics and performance measures and highlight ways in which the performance of classifiers can be compared. Since the underlying mechanics of different evaluation metrics may vary, understanding what exactly each of these metrics represents and what kind of information they are trying to convey is crucial for comparability. Some examples of these metrics include recall, precision, accuracy, F-measure, micro-average, and macro average. These metrics are based on a "confusion matrix" (shown in Figure 24) that comprises true positives (TP), false positives (FP), false negatives (FN), and true negatives (TN) [216]. The significance of these four elements may vary based on the classification application. The fraction of correct predictions over all predictions is called accuracy (Equation (166)). The fraction of known positives that are correctly predicted is called sensitivity i.e., true positive rate or recall (Equation (167)). The ratio of correctly predicted negatives is called specificity (Equation (168)). The proportion of correctly predicted positives to all positives is called precision, i.e., positive predictive value (Equation (169)).

$$accuracy = \frac{(TP + TN)}{(TP + FP + FN + TN)} \tag{166}$$

$$sensitivity = \frac{TP}{(TP + FN)} \tag{167}$$

$$specificity = \frac{TN}{(TN + FP)} \tag{168}$$

$$precision = \frac{\sum_{l=1}^{L} TP_l}{\sum_{l=1}^{L} TP_l + FP_l} \tag{169}$$

$$recall = \frac{\sum_{l=1}^{L} TP_l}{\sum_{l=1}^{L} TP_l + FN_l} \tag{170}$$

$$F1 - Score = \frac{\sum_{l=1}^{L} 2TP_l}{\sum_{l=1}^{L} 2TP_l + FP_l + FN_l} \tag{171}$$

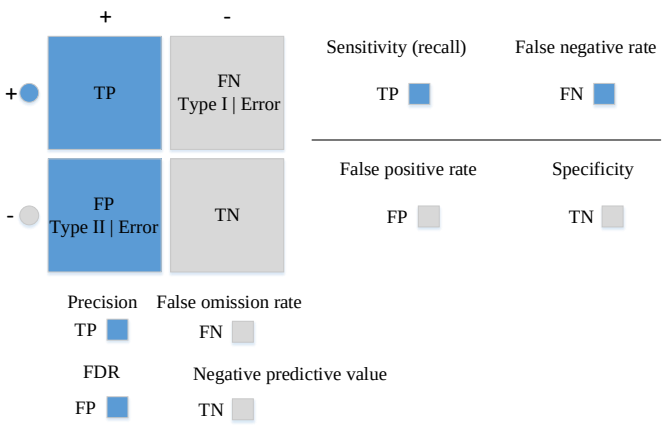

Figure 24. Confusion matrix.

5.1. Macro-Averaging and Micro-Averaging

A single aggregate measure is required when several two-class classifiers are being used to process a collection. Macro-averaging gives a simple average over classes while micro-averaging combines per-document decisions across classes and then outputs an effective measure on the pooled contingency table [217]. Macro-averaged results can be computed as follows:

$$B_{macro} = \frac{1}{q}\sum_{\lambda=1}^{q} B(TP_\lambda + FP_\lambda + TN_\lambda + FN_\lambda) \tag{172}$$

where B is a binary evaluation measure calculated based on true positives (TP), false positives (FP), false negatives (FN), and true negatives (TN), and $L = \{\lambda_j : j = 1...q\}$ is the set of all labels.

Micro-averaged results [156,218] can be computed as follows:

$$B_{macro} = B\left(\sum_{\lambda=1}^{q} TP_\lambda, \sum_{\lambda=1}^{q} FP_\lambda, \sum_{\lambda=1}^{q} TN_\lambda, \sum_{\lambda=1}^{q} FN_\lambda\right) \tag{173}$$

Micro-average score assigns equal weights to every document as a consequence, and it is considered to be a per-document average. On the other hand, macro-average score assigns equal weights to each category without accounting for frequency and therefore, it is a per-category average.

5.2. F_β Score

F_β is one of the most popular aggregated evaluation metrics for classifier evaluation [216]. The parameter β is used to balance recall and precision and is defined as follows:

$$F_\beta = \frac{(1+\beta^2)(precision \times recall)}{\beta^2 \times precision + recall} \tag{174}$$

For commonly used $\beta = 1$ i.e., F_1, recall and precision are given equal weights and Equation (174) can be simplified to:

$$F_1 = \frac{2TP}{2TP + FP + FN} \tag{175}$$

Since F_β is based on recall and precision, it does not represent the confusion matrix fully.

5.3. Matthews Correlation Coefficient (MCC)

The Matthews correlation coefficient (MCC) [30] captures all the data in a confusion matrix and measures the quality of binary classification methods. MCC can be used for problems with uneven class sizes and is still considered a balanced measure. MCC ranges from -1 to 0 (i.e., the classification is always wrong and always true, respectively). MCC can be calculated as follows:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{\begin{array}{c}(TP + FP) \times (TP + FN) \times \\ (TN + FP) \times (TN + FN)\end{array}}} \tag{176}$$

While comparing two classifiers, one may have a higher score using MCC and the other one has a higher score using F_1 and as a result one specific metric cannot captures all the strengths and weaknesses of a classifier [216].

5.4. Receiver Operating Characteristics (ROC)

Receiver operating characteristics (ROC) [219] curves are valuable graphical tools for evaluating classifiers. However, class imbalances (i.e., differences in prior class probabilities [220]) can cause ROC

curves to poorly represent the classifier performance. ROC curve plots true positive rate (TPR) and false positive rate (FPR):

$$TPR = \frac{TP}{TP + FN} \tag{177}$$

$$FPR = \frac{FP}{FP + TN} \tag{178}$$

5.5. Area Under ROC Curve (AUC)

The area under ROC curve (AUC) [31,32] measures the entire area underneath the ROC curve. AUC leverages helpful properties such as increased sensitivity in the analysis of variance (ANOVA) tests, independence from decision threshold, invariance to a priori class probabilities, and indication of how well negative and positive classes are in regarding the decision index [221].

For binary classification tasks, AUC can be formulated as:

$$\begin{aligned} AUC &= \int_{-\infty}^{\infty} TPR(T)FPR'(T)dT \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(T' > T)f_1(T')f_0(T)dTdT' \\ &= P(X_1 > X_0) \end{aligned} \tag{179}$$

For multi-class AUC, an average AUC can be defined [222] as follows:

$$AUC = \frac{2}{|C|(|C|-1)}\sum_{i=1}^{|C|} AUC_i \tag{180}$$

where C is the number of the classes.

Yang [215] evaluated statistical approaches for text categorization and reported the following important factors that should be considered when comparing classifier algorithms:

- Comparative evaluation across methods and experiments which gives insight about factors underlying performance variations and will lead to better evaluation methodology in the future;
- Impact of collection variability such as including unlabeled documents in training or test set and treat them as negative instances can be a serious problem;
- Category ranking evaluation and binary classification evaluation show the usefulness of classifier in interactive applications and emphasize their use in a batch mode respectively. Having both types of performance measurements to rank classifiers is helpful in detecting the effects of thresholding strategies;
- Evaluation of the scalability of classifiers in large category spaces is a rarely investigated area.

6. Discussion

In this article, we aimed to present a brief overview of text classification techniques, alongside a discussion of corresponding pre-processing steps and evaluation methods. In this section, we compare and contrast each of these techniques and algorithms. Moreover, we discuss the limitations of existing classification techniques and evaluation methods. The main challenge to choosing an efficient classification system is understanding similarity and differences of available techniques in different pipeline steps.

6.1. Text and Document Feature Extraction

We outlined the following two main feature extraction approaches: Weighted words (bag-of-words) and word embedding. Word embedding techniques learn from sequences of words by

taking into consideration their occurrence and co-occurrence information. Also, these methods are unsupervised models for generating word vectors. In contrast, weighted words features are based on counting words in documents and can be used as a simple scoring scheme of word representation. Each technique presents unique limitations.

Weighted words computes document similarity directly from the word-count space which increases the computational time for large vocabularies [223]. While counts of unique words provide independent evidence of similarity, they do not account for semantic similarities between words (e.g., "Hello" and "Hi"). Word embedding methods address this issue but are limited by the necessitation of a huge corpus of text data sets for training [224]. As a result, scientists prefer to use pre-trained word embedding vectors [224]. However, this approach cannot work for words missing from these text data corpora.

For example, in some short message service (SMS) data sets, people use words with multiple meaning such as slang or abbreviation which do not have semantic similarities. Furthermore, abbreviations are not included in the pre-trained word embedding vectors. To solve these problems, many researchers work on text cleaning as we discussed in Section 2. The word embedding techniques such as GloVe, FastText, and Word2Vec were trained based on the word and nearest neighbor of that word, and this contains a very critical limitation (the meaning of a word could be different in two different sentences). To solve this problem, scientist have come up with the novel methods called contextualized word representations, which train based on the context of the word in a document.

As shown in Table 1, we compare and evaluate each technique including weighted words, TF-IDF, Word2Vec, Glove, FastText, and contextualized word representations.

Table 1. Feature extraction comparison.

Model	Advantages	Limitation
Weighted Words	• Easy to compute • Easy to compute the similarity between 2 documents using it • Basic metric to extract the most descriptive terms in a document • Works with unknown word (e.g., New words in languages)	• It does not capture position in text (syntactic) • It does not capture meaning in text (semantics) • Common words effect on the results (e.g., "am", "is", etc.)
TF-IDF	• Easy to compute • Easy to compute the similarity between 2 documents using it • Basic metric to extract the most descriptive terms in a document • Common words do not effect the results due to IDF (e.g., "am", "is", etc.)	• It does not capture position in text (syntactic) • It does not capture meaning in text (semantics)
Word2Vec	• It captures position of the words in the text (syntactic) • It captures meaning in the words (semantics)	• It cannot capture the meaning of the word from the text (fails to capture polysemy) • It cannot capture out-of-vocabulary words from corpus
GloVe (Pre-Trained)	• It captures position of the words in the text (syntactic) • It captures meaning in the words (semantics) • Trained on huge corpus	• It cannot capture the meaning of the word from the text (fails to capture polysemy) • Memory consumption for storage • It cannot capture out-of-vocabulary words from corpus

Table 1. *Cont.*

Model	Advantages	Limitation
GloVe (Trained)	• It is very straightforward, e.g., to enforce the word vectors to capture sub-linear relationships in the vector space (performs better than Word2vec) • Lower weight for highly frequent word pairs such as stop words like "am", "is", etc. Will not dominate training progress	• Memory consumption for storage • Needs huge corpus to learn • It cannot capture out-of-vocabulary words from corpus • It cannot capture the meaning of the word from the text (fails to capture polysemy)
FastText	• Works for rare words (rare in their character n-grams which are still shared with other words • Solves out of vocabulary words with n-gram in character level	• It cannot capture the meaning of the word from the text (fails to capture polysemy) • Memory consumption for storage • Computationally is more expensive in comparing with GloVe and Word2Vec
Contextualized Word Representations	• It captures the meaning of the word from the text (incorporates context, handling polysemy)	• Memory consumption for storage • Improves performance notably on downstream tasks. Computationally is more expensive in comparison to others • Needs another word embedding for all LSTM and feed forward layers • It cannot capture out-of-vocabulary words from corpus • Works only sentence and document level (it cannot work for individual word level)

6.2. Dimensionality Reduction

In Section 3, we outlined many dimensionality reduction techniques. In this section, we discuss how the efficacy of this step with respect to a text classification system's computational time and robustness. Dimensionality reduction is mostly used for improving computational time and reducing memory complexity.

PCA attempts to find orthogonal projections of the data set which contains the highest variance possible in order to extract linear correlations between variables of the data set. The main limitation of PCA is the computational complexity of this technique for dimensionality reduction [225]. To solve this problem, scientists introduced the random projection technique (Section 3).

LDA is a supervised technique for dimension reduction that can improve the predictive performance of the extracted features. However, LDA requires researchers to manually input the number of components, requires labeled data, and produces features that are not easily interpretable [226].

Random projection is much faster computationally than PCA. However, this method does not perform well for small data sets [227].

Autoencoders requires more data to train than other DR methods, and thus cannot be used as a general-purpose dimensionality reduction algorithm without sufficient data.

T-SNE is mostly used for data visualization in text and document data sets.

6.3. Existing Classification Techniques

In this section, we discuss the limitations and advantages of existing text and document classification algorithms. Then we compare state-of-the-art techniques in two tables.

6.3.1. Limitations and Advantages

As shown in Tables 2 and 3, the Rocchio algorithm is limited by the fact that the user can only retrieve a few relevant documents using this model [108]. Furthermore, the algorithms' results illustrate several limitations in text classification, which could be addressed by taking semantics into consideration [109]. Boosting and bagging methods also have many limitations and disadvantages, such as the computational complexity and loss of interpretability [117]. LR works well for predicting categorical outcomes. However, this prediction requires that each data point be independent [124] which is attempting to predict outcomes based on a set of independent variables [125]. Naïve Bayes algorithm also has several limitations. NBC makes a strong assumption about the shape of the data distribution [134,135]. NBC is also limited by data scarcity for which any possible value in feature space, a likelihood value must be estimated by a frequentist [136]. KNN is a classification method that is easy to implement and adapts to any kind of feature space. This model also naturally handles multi-class cases [140,141]. However, KNN is limited by data storage constraint for large search problems to find the nearest neighbors. Additionally, the performance of KNN is dependent on finding a meaningful distance function, thus making this technique a very data-dependent algorithm [142,143]. SVM has been one of the most efficient machine learning algorithms since its introduction in the 1990s [159]. However, they are limited by the lack of transparency in results caused by a high number of dimensions. Due to this, it cannot show the company score as a parametric function based on financial ratios nor any other functional form [159]. A further limitation is a variable financial ratios rate [160]. The decision tree is a very fast algorithm for both learning and prediction, but it is also extremely sensitive to small perturbations in the data [166], and can be easily overfit [167]. These effects can be negated by validation methods and pruning, but this is a grey area [166]. This model also has problems with out-of-sample prediction [168]. Random forests (i.e., ensembles of decision trees) are very fast to train in comparison to other techniques, but quite slow to create predictions once trained [172]. Thus, in order to achieve a faster structure, the number of trees in forest must be reduced, as more trees in forest increases time complexity in the prediction step. With regards to CRF, the most evident disadvantage of CRF is the high computational complexity of the training step [176], and this algorithm does not perform with unknown words (i.e., with words that were not present in training data sample) [177]. Deep learning (DL) is one of the most powerful techniques in artificial intelligence (AI), and many researchers and scientists focus on deep learning architectures to improve the robustness and computational power of this tool. However, deep learning architectures also have some disadvantages and limitations when applied to classification tasks. One of the main problems of this model is that DL does not facilitate comprehensive theoretical understanding of learning [202]. A well-known disadvantage of DL methods is their "black box" nature [203,204]. That is, the method by which DL methods come up with the convolved output is not readily understandable. Another limitation of DL is that it usually requires much more data than traditional machine learning algorithms, which means that this technique cannot be applied to classification tasks over small data sets [205,206]. Additionally, the massive amount of data needed for DL classification algorithms further exacerbates the computational complexity during the training step [207].

Table 2. Text classification comparison (Rocchio algorithm, boosting, bagging, logistic regressio, Naïve Bayes classifier, k-nearest Neighbor, and Support Vector Machine).

Model	Advantages	Pitfall
Rocchio Algorithm	• Easy to implement • Computationally is very cheap • Relevance feedback mechanism (benefits to ranking documents as not relevant)	• The user can only retrieve a few relevant documents • Rocchio often misclassifies the type for multimodal class • This techniques is not very robust • Linear combination in this algorithm is not good for multi-class data sets
Boosting and Bagging	• Improves the stability and accuracy (takes advantage of ensemble learning where in multiple weak learner outperform a single strong learner) • Reducing variance which helps to avoid overfitting problems	• Computational complexity • Loss of interpretability (if number of model is high, understanding the model is very difficult) • Requires careful tuning of different hyper-parameters
Logistic Regressio	• Easy to implement • Does not require too many computational resources • It does not require input features to be scaled (pre-processing) • It does not require any tuning	• It cannot solve non-linear problems • Prediction requires that each data point be independent • Attempting to predict outcomes based on a set of independent variables
Naïve Bayes Classifier	• It works very well with text data • Easy to implement • Fast in comparison to other algorithms	• A strong assumption about the shape of the data distribution • Limited by data scarcity for which any possible value in feature space, a likelihood value must be estimated by a frequentist
K-Nearest Neighbor	• Effective for text data sets • Non-parametric • More local characteristics of text or document are considered • Naturally handles multi-class data sets	• Computational of this model is very expensive • Difficult to find optimal value of k • Constraint for large search problems to find nearest neighbors • Finding a meaningful distance function is difficult for text data sets
Support Vector Machine (SVM)	• SVM can model non-linear decision boundaries • Performs similarly to logistic regression when linear separation • Robust against overfitting problems (especially for text data set due to high-dimensional space)	• Lack of transparency in results caused by a high number of dimensions (especially for text data). • Choosing an efficient kernel function is difficult (susceptible to overfitting/training issues depending on kernel) • Memory complexity

Table 3. Text classification comparison (decision tree, conditional random field(CRF), random forest, and deep learning).

Model	Advantages	Pitfall
Decision Tree	• Can easily handle qualitative (categorical) features • Works well with decision boundaries parellel to the feature axis • Decision tree is a very fast algorithm for both learning and prediction	• Issues with diagonal decision boundaries • Can be easily overfit • Extremely sensitive to small perturbations in the data • Problems with out-of-sample prediction
Conditional Random Field (CRF)	• Its feature design is flexible • Since CRF computes the conditional probability of global optimal output nodes, it overcomes the drawbacks of label bias • Combining the advantages of classification and graphical modeling which combine the ability to compactly model multivariate data	• High computational complexity of the training step • Rhis algorithm does not perform with unknown words • Problem about online learning (It makes it very difficult to re-train the model when newer data becomes available)
Random Forest	• Ensembles of decision trees are very fast to train in comparison to other techniques • Reduced variance (relative to regular trees) • Does not require preparation and pre-processing of the input data	• Quite slow to create predictions once trained • More trees in forest increases time complexity in the prediction step • Not as easy to visually interpret • Overfitting can easily occur • Need to choose the number of trees at forest
Deep Learning	• Flexible with features design (reduces the need for feature engineering, one of the most time-consuming parts of the machine learning practice) • Architecture that can be adapted to new problems • Can deal with complex input-output mappings • Can easily handle online learning (It makes it very easy to re-train the model when newer data becomes available) • Parallel processing capability (It can perform more than one job at the same time)	• Requires a large amount of data (if you only have small sample text data, deep learning is unlikely to outperform other approaches. • Is extremely computationally expensive to train. • Model interpretability is the most important problem of deep learning (deep learning most of the time is a black-box) • Finding an efficient architecture and structure is still the main challenge of this technique

6.3.2. State-of-the-Art Techniques' Comparison

Regarding Tables 4 and 5, text classification techniques are compared with the criteria: Architecture, author(s), model, novelty, feature extraction, details, corpus, validation measure, and limitation of each technique. Each text classification technique (system) contains a model which is the classifier algorithm, and also needs a feature extraction technique which means converting texts or documents data set into numerical data (as discussed in Section 2). Another important part in our comparison is the validation measure which is used to evaluate the system.

Table 4. Comparison of text classification techniques.

Model	Author(s)	Architecture	Novelty	Feature Extraction	Details	Corpus	Validation Measure	Limitation
Rocchio Algorithm	B.J. Sowmya et al. [106]	Hierarchical Rocchio	Classificationon hierarchical data	TF-IDF	Use CUDA on GPU to compute and compare the distances.	Wikipedia	F1-Macro	Works only on hierarchical data sets and retrieves a few relevant documents
Boosting	S. Bloehdorn et al. [114]		AdaBoost for with semantic features	BOW	Ensemble learning algorithm	Reuters-21578	F1-Macro and F1-Micro	Computational complexity and loss of interpretability
Logistic Regression	A. Genkin et al. [120]	Bayesian Logistic Regression	Logistic regression analysis of high-dimensional data	TF-IDF	It is based on Gaussian Priors and Ridge Logistic Regression	RCV1-v2	F1-Macro	Prediction outcomes is based on a set of independent variables
Naïve Bayes	Kim, S.B et al. [131]	Weight Enhancing Method	Multivariate poisson model for text Classification	Weights words	Per-document term frequency normalization to estimate the Poisson parameter	Reuters-21578	F1-Macro	This method makes a strong assumption about the shape of the data distribution
SVM and KNN	K. Chen et al. [148]	Inverse Gravity Moment	Introduced TFIGM (term frequency & inverse gravity moment)	TF-IDF and TFIGM	Incorporates a statistical model to precisely measure the class distinguishing power of a term	20 Newsgroups and Reuters-21578	F1-Macro	Fails to capture polysemy and also still semantic and sentatics is not solved
Support Vector Machines	H. Lodhi et al. [151]	String Subsequence Kernel	Use of a special kernel	Similarity using TF-IDF	The kernel is an inner product in the feature space generated by all subsequences of length k	Reuters-21578	F1-Macro	The lack of transparency in the results
Conditional Random Field (CRF)	T. Chen et al. [175]	BiLSTM-CRF	Apply a neural network based sequence model to classify opinionated sentences into three types according to the number of targets appearing in a sentence	Word embedding	Improve sentence-level sentiment analysis via sentence type classification	Customer reviews	Accuracy	High computational complexity and this algorithm does not perform with unseen words

Table 5. Comparison of the text classification techniques (continue).

Model	Author(s)	Architecture	Novelty	Feature Extraction	Details	Corpus	Validation Measure	Limitation
Deep Learning	Z. Yang et al. [193]	Hierarchical Attention Networks	It has a hierarchical structure	Word embedding	Two levels of attention mechanisms applied at the word and sentence-level	Yelp, IMDB review, and Amazon review	Accuracy	Works only for document-level
Deep Learning	J. Chen et al. [228]	Deep Neural Networks	Convolutional neural networks (CNN) using 2-dimensional TF-IDF features	2D TF-IDF	A new solution to the verbal aggression detection task	Twitter comments	F1-Macro and F1-Micro	Data dependent for designed a model architecture
Deep Learning	M. Jiang et al. [1]	Deep Belief Network	Hybrid text classification model based on deep belief network and softmax regression.	DBN	DBN completes the feature learning to solve the high dimension and sparse matrix problem and softmax regression is employed to classify the texts	Reuters-21578 and 20-Newsgroup	Error-rate	Computationally is expensive and model interpretability is still a problem of this model
Deep Learning	X. Zhang et al. [229]	CNN	Character-level convolutional networks (ConvNets) for text classification	Encoded Characters	Character-level ConvNet contains 6 convolutional layers and 3 fully-connected layers	Yelp, Amazon review and Yahoo! Answers data set	Relative errors	This model is only designed to discover position-invariant features of their inputs
Deep Learning	K. Kowsari [4]	Ensemble deep learning algorithm (CNN, DNN and RNN)	Solves the problem of finding the best deep learning structure and architecture	TF-IDF and GloVe	Random Multimodel Deep Learning (RDML)	IMDB review, Reuters-21578, 20NewsGroup, and WOS	Accuracy	Computationally is expensive
Deep Learning	K. Kowsari [2]	Hierarchical structure	Employs stacks of deep learning architectures to provide specialized understanding at each level of the document hierarchy	TF-IDF and GloVe	Hierarchical Deep Learning for Text Classification (HDLTex)	Web of science data set	Accuracy	Works only for hierarchical data sets

6.4. Evaluation

The experimental evaluation of text classifiers measures effectiveness (i.e., capacity to make the right classification or categorization decision). Precision and recall are widely used to measure the effectiveness of text classifiers. Accuracy and error ($\frac{FP+FN}{TP+TN+FP+FN} = 1 -$ accuracy), on the other hand, are not widely used for text classification applications because they are insensitive to variations in the number of correct decisions due to the large value of the denominator ($TP + TN$) [215]. The pitfalls of each of the above-mentioned metrics are listed in Table 6.

Table 6. Metrics pitfalls.

	Limitation
Accuracy	Gives us no information on False Negative (FN) and False Positive (FP)
Sensitivity	Does not evaluate True Negative (TN) and FP and any classifier that predicts data points as positives considered to have high sensitivity
Specificity	Similar to sensitivity and does not account for FN and TP
Precision	Does not evaluate TN and FN and considered to be very conservative and goes for a case which is most certain to be positive

7. Text Classification Usage

In the earliest history of ML and AI, text classification techniques have mostly been used for information retrieval systems. However, as technological advances have emerged over time, text classification and document categorization have been globally used in many domains such as medicine, social sciences, healthcare, psychology, law, engineering, etc. In this section, we highlight several domains which make use of text classification techniques.

7.1. Text Classification Applications

7.1.1. Information Retrieval

Information retrieval is finding documents of an unstructured data that meet an information need from within large collections of documents [230]. With the rapid growth of online information, particularly in text format, text classification has become a significant technique for managing this type of data [231]. Some of the important methods used in this area are Naïve Bayes, SVM, decision tree, J48, KNN, and IBK [232]. One of the most challenging applications for document and text data set processing is applying document categorization methods for information retrieval [34,233].

7.1.2. Information Filtering

Information filtering refers to the selection of relevant information or rejection of irrelevant information from a stream of incoming data. Information filtering systems are typically used to measure and forecast users' long-term interests [234]. Probabilistic models, such as the Bayesian inference network, are commonly used in information filtering systems. Bayesian inference networks employ recursive inference to propagate values through the inference network and return documents with the highest ranking [34]. Buckley, C. [235] used vector space model with iterative refinement for filtering task.

7.1.3. Sentiment Analysis

Sentiment analysis is a computational approach toward identifying opinion, sentiment, and subjectivity in text [236]. Sentiment classification methods classify a document associated with an opinion to be positive or negative. The assumption is that document d is expressing an opinion on a single entity e and opinions are formed via a single opinion holder h [237]. Naive Bayesian

classification and SVM are some of the most popular supervised learning methods that have been used for sentiment classification [238]. Features such as terms and their respective frequency, part of speech, opinion words and phrases, negations, and syntactic dependency have been used in sentiment classification techniques.

7.1.4. Recommender Systems

Content-based recommender systems suggest items to users based on the description of an item and a profile of the user's interests [239].

A user's profile can be learned from user feedback (history of the search queries or self reports) on items as well as self-explained features (filter or conditions on the queries) in one's profile. In this way, input to such recommender systems can be semi-structured such that some attributes are extracted from free-text field while others are directly specified [240]. Many different types of text classification methods, such as decision trees, nearest neighbor methods, Rocchio's algorithm, linear classifiers, probabilistic methods, and Naive Bayes, have been used to model user's preference.

7.1.5. Knowledge Management

Textual databases are significant sources of information and knowledge. A large percentage of corporate information (nearly 80%) exists in textual data formats (unstructured). In knowledge distillation, patterns, or knowledge are inferred from immediate forms that can be semi-structured (e.g., conceptual graph representation) or structured/relational (e.g., data representation). A given intermediate form can be document-based such that each entity represents an object or concept of interest in a particular domain. Document categorization is one of the most common methods for mining document-based intermediate forms [241]. In other work, text classification has been used to find the relationship between railroad accidents' causes and their correspondent descriptions in reports [242].

7.1.6. Document Summarization

Text classification used for document summarizing in which the summary of a document may employ words or phrases which do not appear in the original document [243]. Multi-document summarization also is necessitated due to the rapid increase in online information [244]. Thus, many researchers focus on this task using text classification to extract important features out of a document.

7.2. Text Classification Support

7.2.1. Health

Most textual information in the medical domain is presented in an unstructured or narrative form with ambiguous terms and typographical errors. Such information needs to be available instantly throughout the patient-physicians encounters in different stages of diagnosis and treatment [245]. Medical coding, which consists of assigning medical diagnoses to specific class values obtained from a large set of categories, is an area of healthcare applications where text classification techniques can be highly valuable. In other research, J. Zhang et al. introduced Patient2Vec to learn an interpretable deep representation of longitudinal electronic health record (EHR) data which is personalized for each patient [246]. Patient2Vec is a novel technique of text data set feature embedding that can learn a personalized interpretable deep representation of EHR data based on recurrent neural networks and the attention mechanism. Text classification has also been applied in the development of Medical Subject Headings (MeSH) and Gene Ontology (GO) [247].

7.2.2. Social Sciences

Text classification and document categorization has increasingly been applied to understanding human behavior in past decades [38,248]. Recent data-driven efforts in human behavior research have focused on mining language contained in informal notes and text data sets, including short message service (SMS), clinical notes, social media, etc. [38]. These studies have mostly focused on using approaches based on frequencies of word occurrence (i.e., how often a word appears in a document) or features based on linguistic inquiry word count (LIWC) [249], a well-validated lexicon of categories of words with psychological relevance [250].

7.2.3. Business and Marketing

Profitable companies and organizations are progressively using social media for marketing purposes [251]. Opening mining from social media such as Facebook, Twitter, and so on is main target of companies to rapidly increase their profits [252]. Text and documents classification is a powerful tool for companies to find their customers more easily.

7.2.4. Law

Huge volumes of legal text information and documents have been generated by government institutions. Retrieving this information and automatically classifying it can not only help lawyers but also their clients [253]. In the United States, the law is derived from five sources: Constitutional law, statutory law, treaties, administrative regulations, and the common law [254]. Many new legal documents are created each year. Categorization of these documents is the main challenge for the lawyer community.

8. Conclusions

The classification task is one of the most indispensable problems in machine learning. As text and document data sets proliferate, the development and documentation of supervised machine learning algorithms becomes an imperative issue, especially for text classification. Having a better document categorization system for this information requires discerning these algorithms. However, the existing text classification algorithms work more efficiently if we have a better understanding of feature extraction methods and how to evaluate them correctly. Currently, text classification algorithms can be chiefly classified in the following manner: (I) Feature extraction methods, such as Term Frequency-Inverse document frequency (TF-IDF), term frequency (TF), word-embedding (e.g., Word2Vec, contextualized word representations, Global Vectors for Word Representation (GloVe), and FastText), are widely used in both academic and commercial applications. In this paper, we had addressed these techniques. However, text and document cleaning could help the accuracy and robustness of an application. We described the basic methods of text pre-processing step. (II) Dimensionality reduction methods, such as principal component analysis (PCA), linear discriminant analysis (LDA), non-negative matrix factorization (NMF), random projection, Autoencoder, and t-distributed Stochastic Neighbor Embedding (t-SNE), can be useful in reducing the time complexity and memory complexity of existing text classification algorithms. In a separate section, the most common methods of dimensionality reduction were presented. (III) Existing classification algorithms, such as the Rocchio algorithm, bagging and boosting, logistic regression (LR), Naïve Bayes Classifier (NBC), k-nearest Neighbor (KNN), Support Vector Machine (SVM), decision tree classifier (DTC), random forest, conditional random field (CRF), and deep learning, are the primary focus of this paper. (IV) Evaluation methods, such as accuracy, F_β, Matthew correlation coefficient (MCC), receiver operating characteristics (ROC), and area under curve (AUC), are explained. With these metrics, the text classifcation algorithm can be evaluated. (V) Critical limitations of each component of the text classification pipeline (i.e., feature extraction, dimensionality reduction, existing classification algorithms, and evaluation) were addressed in order to each technique. And finally we

compare the most common text classification algorithm in this section. (V) Finally, the usage of text classification as an application and/or support other majors such as lay, medicine, etc. are covered in a separate section.

In this survey, Recent techniques and trending of text classification algorithm have discussed.

Author Contributions: K.K. and K.J.M. worked on the idea and designed the platform, and also they worked on GitHub sample code for all of these models. M.H. and S.M. organized and proofread the paper. This work is under the supervision of L.B. and D.B.

Funding: This work was supported by The United States Army Research Laboratory under Grant W911NF-17-2-0110.

Acknowledgments: The authors would like to thank Matthew S. Gerber for his feedback and comments.

Conflicts of Interest: The authors declare no conflict of interest. The funding sponsors had no role in the design of the study; in the collection, analyses or interpretation of data; in the writing of the manuscript; nor in the decision to publish the results.

References

1. Jiang, M.; Liang, Y.; Feng, X.; Fan, X.; Pei, Z.; Xue, Y.; Guan, R. Text classification based on deep belief network and softmax regression. *Neural Comput. Appl.* **2018**, *29*, 61–70. [CrossRef]
2. Kowsari, K.; Brown, D.E.; Heidarysafa, M.; Jafari Meimandi, K.; Gerber, M.S.; Barnes, L.E. HDLTex: Hierarchical Deep Learning for Text Classification. Machine Learning and Applications (ICMLA). In Proceedings of the 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA), Cancun, Mexico, 18–21 December 2017.
3. McCallum, A.; Nigam, K. A comparison of event models for naive bayes text classification. In Proceedings of the AAAI-98 Workshop on Learning for Text Categorization, Madison, WI, USA, 26–27 July 1998; Volume 752, pp. 41–48.
4. Kowsari, K.; Heidarysafa, M.; Brown, D.E.; Jafari Meimandi, K.; Barnes, L.E. RMDL: Random Multimodel Deep Learning for Classification. In Proceedings of the 2018 International Conference on Information System and Data Mining, Lakeland, FL, USA, 9–11 April 2018; doi:10.1145/3206098.3206111.
5. Heidarysafa, M.; Kowsari, K.; Brown, D.E.; Jafari Meimandi, K.; Barnes, L.E. An Improvement of Data Classification Using Random Multimodel Deep Learning (RMDL). *IJMLC* **2018**, *8*, 298–310. [CrossRef]
6. Lai, S.; Xu, L.; Liu, K.; Zhao, J. Recurrent Convolutional Neural Networks for Text Classification. In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, Austin, TX, USA, 25–30 January 2015; Volume 333, pp. 2267–2273.
7. Aggarwal, C.C.; Zhai, C. A survey of text classification algorithms. In *Mining Text Data*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 163–222.
8. Aggarwal, C.C.; Zhai, C.X. *Mining Text Data*; Springer: Berlin/Heidelberg, Germany, 2012.
9. Salton, G.; Buckley, C. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manag.* **1988**, *24*, 513–523. [CrossRef]
10. Goldberg, Y.; Levy, O. Word2vec explained: Deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv* **2014**, arXiv:1402.3722.
11. Pennington, J.; Socher, R.; Manning, C.D. Glove: Global Vectors for Word Representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014; Volume 14, pp. 1532–1543.
12. Mamitsuka, N.A.H. Query learning strategies using boosting and bagging. In *Machine Learning: Proceedings of the Fifteenth International Conference (ICML'98)*; Morgan Kaufmann Pub.: Burlington, MA, USA, 1998; Volume 1.
13. Kim, Y.H.; Hahn, S.Y.; Zhang, B.T. Text filtering by boosting naive Bayes classifiers. In Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Athens, Greece, 24–28 July 2000; pp. 168–175.
14. Schapire, R.E.; Singer, Y. BoosTexter: A boosting-based system for text categorization. *Mach. Learn.* **2000**, *39*, 135–168. [CrossRef]
15. Harrell, F.E. Ordinal logistic regression. In *Regression Modeling Strategies*; Springer: Berlin/Heidelberg, Germany, 2001; pp. 331–343.

16. Hosmer, D.W., Jr.; Lemeshow, S.; Sturdivant, R.X. *Applied Logistic Regression*; John Wiley & Sons: Hoboken, NJ, USA, 2013; Volume 398.
17. Dou, J.; Yamagishi, H.; Zhu, Z.; Yunus, A.P.; Chen, C.W. TXT-tool 1.081-6.1 A Comparative Study of the Binary Logistic Regression (BLR) and Artificial Neural Network (ANN) Models for GIS-Based Spatial Predicting Landslides at a Regional Scale. In *Landslide Dynamics: ISDR-ICL Landslide Interactive Teaching Tools*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 139–151.
18. Chen, W.; Xie, X.; Wang, J.; Pradhan, B.; Hong, H.; Bui, D.T.; Duan, Z.; Ma, J. A comparative study of logistic model tree, random forest, and classification and regression tree models for spatial prediction of landslide susceptibility. *Catena* **2017**, *151*, 147–160. [CrossRef]
19. Larson, R.R. Introduction to information retrieval. *J. Am. Soc. Inf. Sci. Technol.* **2010**, *61*, 852–853. [CrossRef]
20. Li, L.; Weinberg, C.R.; Darden, T.A.; Pedersen, L.G. Gene selection for sample classification based on gene expression data: Study of sensitivity to choice of parameters of the GA/KNN method. *Bioinformatics* **2001**, *17*, 1131–1142. [CrossRef]
21. Manevitz, L.M.; Yousef, M. One-class SVMs for document classification. *J. Mach. Learn. Res.* **2001**, *2*, 139–154.
22. Han, E.H.S.; Karypis, G. Centroid-based document classification: Analysis and experimental results. In *European Conference on Principles of Data Mining and Knowledge Discovery*; Springer: Berlin/Heidelberg, Germany, 2000; pp. 424–431.
23. Xu, B.; Guo, X.; Ye, Y.; Cheng, J. An Improved Random Forest Classifier for Text Categorization. *JCP* **2012**, *7*, 2913–2920. [CrossRef]
24. Lafferty, J.; McCallum, A.; Pereira, F.C. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In Proceedings of the 18th International Conference on Machine Learning 2001 (ICML 2001), Williamstown, MA, USA, 28 June–1 July 2001.
25. Shen, D.; Sun, J.T.; Li, H.; Yang, Q.; Chen, Z. Document Summarization Using Conditional Random Fields. *IJCAI* **2007**, *7*, 2862–2867.
26. Zhang, C. Automatic keyword extraction from documents using conditional random fields. *J. Comput. Inf. Syst.* **2008**, *4*, 1169–1180.
27. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef] [PubMed]
28. Huang, J.; Ling, C.X. Using AUC and accuracy in evaluating learning algorithms. *IEEE Trans. Knowl. Data Eng.* **2005**, *17*, 299–310. [CrossRef]
29. Lock, G. Acute mesenteric ischemia: Classification, evaluation and therapy. *Acta Gastro-Enterol. Belg.* **2002**, *65*, 220–225.
30. Matthews, B.W. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochim. Biophys. Acta (BBA)-Protein Struct.* **1975**, *405*, 442–451. [CrossRef]
31. Hanley, J.A.; McNeil, B.J. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology* **1982**, *143*, 29–36. [CrossRef]
32. Pencina, M.J.; D'Agostino, R.B.; Vasan, R.S. Evaluating the added predictive ability of a new marker: From area under the ROC curve to reclassification and beyond. *Stat. Med.* **2008**, *27*, 157–172. [CrossRef]
33. Jacobs, P.S. *Text-Based Intelligent Systems: Current Research and Practice in Information Extraction and Retrieval*; Psychology Press: Hove, UK, 2014.
34. Croft, W.B.; Metzler, D.; Strohman, T. *Search Engines: Information Retrieval in Practice*; Addison-Wesley Reading: Boston, MA, USA, 2010; Volume 283.
35. Yammahi, M.; Kowsari, K.; Shen, C.; Berkovich, S. An efficient technique for searching very large files with fuzzy criteria using the pigeonhole principle. In Proceedings of the 2014 Fifth International Conference on Computing for Geospatial Research and Application, Washington, DC, USA, 4–6 August 2014; pp. 82–86.
36. Chu, Z.; Gianvecchio, S.; Wang, H.; Jajodia, S. Who is tweeting on Twitter: Human, bot, or cyborg? In Proceedings of the 26th Annual Computer Security Applications Conference, Austin, TX, USA, 6–10 December 2010; pp. 21–30.
37. Gordon, R.S., Jr. An operational classification of disease prevention. *Public Health Rep.* **1983**, *98*, 107. [PubMed]
38. Nobles, A.L.; Glenn, J.J.; Kowsari, K.; Teachman, B.A.; Barnes, L.E. Identification of Imminent Suicide Risk Among Young Adults using Text Messages. In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, Montreal, QC, Canada, 21–26 April 2018; p. 413.

39. Gupta, G.; Malhotra, S. Text Document Tokenization for Word Frequency Count using Rapid Miner (Taking Resume as an Example). *Int. J. Comput. Appl.* **2015**, *975*, 8887.
40. Verma, T.; Renu, R.; Gaur, D. Tokenization and filtering process in RapidMiner. *Int. J. Appl. Inf. Syst.* **2014**, *7*, 16–18. [CrossRef]
41. Aggarwal, C.C. *Machine Learning for Text*; Springer: Berlin/Heidelberg, Germany, 2018.
42. Saif, H.; Fernández, M.; He, Y.; Alani, H. On stopwords, filtering and data sparsity for sentiment analysis of twitter. In Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC 2014), Reykjavik, Iceland, 26–31 May 2014.
43. Gupta, V.; Lehal, G.S. A survey of text mining techniques and applications. *J. Emerg. Technol. Web Intell.* **2009**, *1*, 60–76. [CrossRef]
44. Dalal, M.K.; Zaveri, M.A. Automatic text classification: A technical review. *Int. J. Comput. Appl.* **2011**, *28*, 37–40. [CrossRef]
45. Whitney, D.L.; Evans, B.W. Abbreviations for names of rock-forming minerals. *Am. Mineral.* **2010**, *95*, 185–187. [CrossRef]
46. Helm, A. Recovery and reclamation: A pilgrimage in understanding who and what we are. In *Psychiatric and Mental Health Nursing: The Craft of Caring*; Routledge: London, UK, 2003; pp. 50–55.
47. Dhuliawala, S.; Kanojia, D.; Bhattacharyya, P. SlangNet: A WordNet like resource for English Slang. In Proceedings of the LREC, Portorož, Slovenia, 23–28 May 2016.
48. Pahwa, B.; Taruna, S.; Kasliwal, N. Sentiment Analysis-Strategy for Text Pre-Processing. *Int. J. Comput. Appl.* **2018**, *180*, 15–18. [CrossRef]
49. Mawardi, V.C.; Susanto, N.; Naga, D.S. Spelling Correction for Text Documents in Bahasa Indonesia Using Finite State Automata and Levinshtein Distance Method. *EDP Sci.* **2018**, *164*. [CrossRef]
50. Dziadek, J.; Henriksson, A.; Duneld, M. Improving Terminology Mapping in Clinical Text with Context-Sensitive Spelling Correction. In *Informatics for Health: Connected Citizen-Led Wellness and Population Health*; IOS Press: Amsterdam, The Netherlands, 2017; Volume 235, pp. 241–245.
51. Mawardi, V.C.; Rudy, R.; Naga, D.S. Fast and Accurate Spelling Correction Using Trie and Bigram. *TELKOMNIKA (Telecommun. Comput. Electron. Control)* **2018**, *16*, 827–833. [CrossRef]
52. Spirovski, K.; Stevanoska, E.; Kulakov, A.; Popeska, Z.; Velinov, G. Comparison of different model's performances in task of document classification. In Proceedings of the 8th International Conference on Web Intelligence, Mining and Semantics, Novi Sad, Serbia, 25–27 June 2018; p. 10.
53. Singh, J.; Gupta, V. Text stemming: Approaches, applications, and challenges. *ACM Compu. Surv. (CSUR)* **2016**, *49*, 45. [CrossRef]
54. Sampson, G. *The'Language Instinct'Debate: Revised Edition*; A&C Black: London, UK, 2005.
55. Plisson, J.; Lavrac, N.; Mladenić, D. A rule based approach to word lemmatization. In Proceedings of the 7th International MultiConference Information Society IS 2004, Ljubljana, Slovenia, 13–14 October 2004.
56. Korenius, T.; Laurikkala, J.; Järvelin, K.; Juhola, M. Stemming and lemmatization in the clustering of finnish text documents. In Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management, Washington, DC, USA, 8–13 November 2004; pp. 625–633.
57. Caropreso, M.F.; Matwin, S. Beyond the bag of words: A text representation for sentence selection. In *Conference of the Canadian Society for Computational Studies of Intelligence*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 324–335.
58. Sidorov, G.; Velasquez, F.; Stamatatos, E.; Gelbukh, A.; Chanona-Hernández, L. Syntactic dependency-based n-grams as classification features. In *Mexican International Conference on Artificial Intelligence*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 1–11.
59. Sparck Jones, K. A statistical interpretation of term specificity and its application in retrieval. *J. Doc.* **1972**, *28*, 11–21. [CrossRef]
60. Tokunaga, T.; Makoto, I. Text categorization based on weighted inverse document frequency. *Inf. Process. Soc. Jpn. SIGNL* **1994**, *94*, 33–40.
61. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient estimation of word representations in vector space. *arXiv* **2013**, arXiv:1301.3781.
62. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.S.; Dean, J. Distributed representations of words and phrases and their compositionality. *Adv. Neural Inf. Process. Syst.* **2013**, *26*, 3111–3119.
63. Rong, X. word2vec parameter learning explained. *arXiv* **2014**, arXiv:1411.2738.

64. Maaten, L.V.D.; Hinton, G. Visualizing data using t-SNE. *J. Mach. Learn. Res.* **2008**, *9*, 2579–2605.
65. Bojanowski, P.; Grave, E.; Joulin, A.; Mikolov, T. Enriching word vectors with subword information. *arXiv* **2016**, arXiv:1607.04606.
66. Melamud, O.; Goldberger, J.; Dagan, I. context2vec: Learning generic context embedding with bidirectional lstm. In Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, Berlin, Germany, 11–12 August 2016; pp. 51–61.
67. Peters, M.E.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; Zettlemoyer, L. Deep contextualized word representations. *arXiv* **2018**, arXiv:1802.05365.
68. Abdi, H.; Williams, L.J. Principal component analysis. *Wiley Interdiscip. Rev. Comput. Stat.* **2010**, *2*, 433–459. [CrossRef]
69. Jolliffe, I.T.; Cadima, J. Principal component analysis: A review and recent developments. *Philos. Trans. R. Soc. A* **2016**, *374*, 20150202. [CrossRef]
70. Ng, A. Principal components analysis. Generative Algorithms, Regularization and Model Selection. *CS* **2015**, *229*, 71.
71. Cao, L.; Chua, K.S.; Chong, W.; Lee, H.; Gu, Q. A comparison of PCA, KPCA and ICA for dimensionality reduction in support vector machine. *Neurocomputing* **2003**, *55*, 321–336. [CrossRef]
72. Hérault, J. Réseaux de neurones à synapses modifiables: Décodage de messages sensoriels composites par une apprentissage non supervisé et permanent. *CR Acad. Sci. Paris* **1984**, *299* 525–528.
73. Jutten, C.; Herault, J. Blind separation of sources, part I: An adaptive algorithm based on neuromimetic architecture. *Signal Process.* **1991**, *24*, 1–10. [CrossRef]
74. Hyvärinen, A.; Hoyer, P.O.; Inki, M. Topographic independent component analysis. *Neural Comput.* **2001**, *13*, 1527–1558. [CrossRef] [PubMed]
75. Hyvärinen, A.; Oja, E. Independent component analysis: algorithms and applications. *Neural Netw.* **2000**, *13*, 411–430. [CrossRef]
76. Sugiyama, M. Dimensionality reduction of multimodal labeled data by local fisher discriminant analysis. *J. Mach. Learn. Res.* **2007**, *8*, 1027–1061.
77. Balakrishnama, S.; Ganapathiraju, A. Linear discriminant analysis-a brief tutorial. *Inst. Signal Inf. Process.* **1998**, *18*, 1–8.
78. Sugiyama, M. Local fisher discriminant analysis for supervised dimensionality reduction. In Proceedings of the 23rd International Conference on Machine Learning, Pittsburgh, PA, USA, 25–29 June 2006, pp. 905–912.
79. Pauca, V.P.; Shahnaz, F.; Berry, M.W.; Plemmons, R.J. Text mining using non-negative matrix factorizations. In Proceedings of the 2004 SIAM International Conference on Data Mining, Lake Buena Vista, FL, USA, 22–24 April 2004; pp. 452–456.
80. Tsuge, S.; Shishibori, M.; Kuroiwa, S.; Kita, K. Dimensionality reduction using non-negative matrix factorization for information retrieval. In Proceedings of the 2001 IEEE International Conference on Systems, Man, and Cybernetics, Tucson, AZ, USA, 7–10 October 2001; Volume 2, pp. 960–965.
81. Kullback, S.; Leibler, R.A. On information and sufficiency. *Ann. Math. Stat.* **1951**, *22*, 79–86. [CrossRef]
82. Johnson, D.; Sinanovic, S. Symmetrizing the Kullback-Leibler Distance. *IEEE Trans. Inf. Theory* **2001**. Available online: https://scholarship.rice.edu/bitstream/handle/1911/19969/Joh2001Mar1Symmetrizi.PDF?sequence=1 (accessed on 23 April 2019).
83. Bingham, E.; Mannila, H. Random projection in dimensionality reduction: Applications to image and text data. In Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 26–29 August 2001; pp. 245–250.
84. Chakrabarti, S.; Roy, S.; Soundalgekar, M.V. Fast and accurate text classification via multiple linear discriminant projections. *VLDB J.* **2003**, *12*, 170–185. [CrossRef]
85. Rahimi, A.; Recht, B. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. *Adv. Neural Inf. Process. Syst.* **2009**, *21*, 1313–1320.
86. Morokoff, W.J.; Caflisch, R.E. Quasi-monte carlo integration. *J. Comput. Phys.* **1995**, *122*, 218–230. [CrossRef]
87. Johnson, W.B.; Lindenstrauss, J.; Schechtman, G. Extensions of lipschitz maps into Banach spaces. *Isr. J. Math.* **1986**, *54*, 129–138. [CrossRef]
88. Dasgupta, S.; Gupta, A. An elementary proof of a theorem of Johnson and Lindenstrauss. *Random Struct. Algorithms* **2003**, *22*, 60–65. [CrossRef]
89. Vempala, S.S. *The Random Projection Method*; American Mathematical Society: Providence, RI, USA, 2005.

90. Mao, X.; Yuan, C. *Stochastic Differential Equations with Markovian Switching*; World Scientific: Singapore, 2016.
91. Goodfellow, I.; Bengio, Y.; Courville, A.; Bengio, Y. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016; Volume 1.
92. Wang, W.; Huang, Y.; Wang, Y.; Wang, L. Generalized autoencoder: A neural network framework for dimensionality reduction. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, Columbus, OH, USA, 23–28 June 2014; pp. 490–497.
93. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. *Learning Internal Representations by Error Propagation*; Technical Report; California University San Diego, Institute for Cognitive Science: La Jolla, CA, USA, 1985.
94. Liang, H.; Sun, X.; Sun, Y.; Gao, Y. Text feature extraction based on deep learning: A review. *EURASIP J. Wirel. Commun. Netw.* **2017**, *2017*, 211. [CrossRef]
95. Baldi, P. Autoencoders, unsupervised learning, and deep architectures. In Proceedings of the ICML Workshop on Unsupervised and Transfer Learning, Bellevue, WA, USA, 2 July 2011; pp. 37–49.
96. AP, S.C.; Lauly, S.; Larochelle, H.; Khapra, M.; Ravindran, B.; Raykar, V.C.; Saha, A. An autoencoder approach to learning bilingual word representations. *Adv. Neural Inf. Process. Syst.* **2014**, *27*, 1853–1861.
97. Masci, J.; Meier, U.; Cireşan, D.; Schmidhuber, J. Stacked convolutional auto-encoders for hierarchical feature extraction. In *International Conference on Artificial Neural Networks*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 52–59.
98. Chen, K.; Seuret, M.; Liwicki, M.; Hennebert, J.; Ingold, R. Page segmentation of historical document images with convolutional autoencoders. In Proceedings of the 2015 13th International Conference on Document Analysis and Recognition (ICDAR), Tunis, Tunisia, 23–26 August 2015; pp. 1011–1015.
99. Geng, J.; Fan, J.; Wang, H.; Ma, X.; Li, B.; Chen, F. High-resolution SAR image classification via deep convolutional autoencoders. *IEEE Geosci. Remote Sens. Lett.* **2015**, *12*, 2351–2355. [CrossRef]
100. Sutskever, I.; Vinyals, O.; Le, Q.V. Sequence to sequence learning with neural networks. *Adv. Neural Inf. Process. Syst.* **2014**, *27*, 3104–3112.
101. Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv* **2014**, arXiv:1406.1078.
102. Hinton, G.E.; Roweis, S.T. Stochastic neighbor embedding. *Adv. Neural Inf. Process. Syst.* **2002**, *15*, 857–864.
103. Joyce, J.M. Kullback-leibler divergence. In *International Encyclopedia of Statistical Science*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 720–722.
104. Rocchio, J.J. Relevance feedback in information retrieval. In *The SMART Retrieval System: Experiments in Automatic Document Processing*; Englewood Cliffs: Prentice-Hall, NJ, USA, 1971; pp. 313–323.
105. Partalas, I.; Kosmopoulos, A.; Baskiotis, N.; Artieres, T.; Paliouras, G.; Gaussier, E.; Androutsopoulos, I.; Amini, M.R.; Galinari, P. LSHTC: A benchmark for large-scale text classification. *arXiv* **2015**, arXiv:1503.08581.
106. Sowmya, B.; Srinivasa, K. Large scale multi-label text classification of a hierarchical data set using Rocchio algorithm. In Proceedings of the 2016 International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS), Bangalore, India, 6–8 October 2016; pp. 291–296.
107. Korde, V.; Mahender, C.N. Text classification and classifiers: A survey. *Int. J. Artif. Intell. Appl.* **2012**, *3*, 85.
108. Selvi, S.T.; Karthikeyan, P.; Vincent, A.; Abinaya, V.; Neeraja, G.; Deepika, R. Text categorization using Rocchio algorithm and random forest algorithm. In Proceedings of the 2016 Eighth International Conference on Advanced Computing (ICoAC), Chennai, India, 19–21 January 2017; pp. 7–12.
109. Albitar, S.; Espinasse, B.; Fournier, S. Towards a Supervised Rocchio-based Semantic Classification of Web Pages. In Proceedings of the KES, San Sebastian, Spain , 10–12 September 2012; pp. 460–469.
110. Farzi, R.; Bolandi, V. Estimation of organic facies using ensemble methods in comparison with conventional intelligent approaches: A case study of the South Pars Gas Field, Persian Gulf, Iran. *Model. Earth Syst. Environ.* **2016**, *2*, 105. [CrossRef]
111. Bauer, E.; Kohavi, R. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Mach. Learn.* **1999**, *36*, 105–139. [CrossRef]
112. Schapire, R.E. The strength of weak learnability. *Mach. Learn.* **1990**, *5*, 197–227. [CrossRef]
113. Freund, Y. An improved boosting algorithm and its implications on learning complexity. In Proceedings of the Fifth Annual Workshop on Computational Learning Theory, Pittsburgh, PA, USA, 27–29 July 1992; pp. 391–398.

114. Bloehdorn, S.; Hotho, A. Boosting for text classification with semantic features. In *International Workshop on Knowledge Discovery on the Web*; Springer: Berlin/Heidelberg, Germany, 2004; pp. 149–166.
115. Freund, Y.; Kearns, M.; Mansour, Y.; Ron, D.; Rubinfeld, R.; Schapire, R.E. Efficient algorithms for learning to play repeated games against computationally bounded adversaries. In Proceedings of the 36th Annual Symposium on Foundations of Computer Science, Milwaukee, WI, USA, 23–25 October 1995; pp. 332–341.
116. Breiman, L. Bagging predictors. *Mach. Learn.* **1996**, *24*, 123–140. [CrossRef]
117. Geurts, P. Some enhancements of decision tree bagging. In *European Conference on Principles of Data Mining and Knowledge Discovery*; Springer: Berlin/Heidelberg, Germany, 2000; pp. 136–147.
118. Cox, D.R. *Analysis of Binary Data*; Routledge: London, UK, 2018.
119. Fan, R.E.; Chang, K.W.; Hsieh, C.J.; Wang, X.R.; Lin, C.J. LIBLINEAR: A library for large linear classification. *J. Mach. Learn. Res.* **2008**, *9*, 1871–1874.
120. Genkin, A.; Lewis, D.D.; Madigan, D. Large-scale Bayesian logistic regression for text categorization. *Technometrics* **2007**, *49*, 291–304. [CrossRef]
121. Juan, A.; Vidal, E. On the use of Bernoulli mixture models for text classification. *Pattern Recogn.* **2002**, *35*, 2705–2710. [CrossRef]
122. Cheng, W.; Hüllermeier, E. Combining instance-based learning and logistic regression for multilabel classification. *Mach. Learn.* **2009**, *76*, 211–225. [CrossRef]
123. Krishnapuram, B.; Carin, L.; Figueiredo, M.A.; Hartemink, A.J. Sparse multinomial logistic regression: Fast algorithms and generalization bounds. *IEEE Trans. Pattern Anal. Mach. Intell.* **2005**, *27*, 957–968. [CrossRef]
124. Huang, K. Unconstrained Smartphone Sensing and Empirical Study for Sleep Monitoring and Self-Management. Ph.D. Thesis, University of Massachusetts Lowell, Lowell, MA, USA, 2015.
125. Guerin, A. *Using Demographic Variables and In-College Attributes to Predict Course-Level Retention for Community College Spanish Students*; Northcentral University: Scottsdale, AZ, USA, 2016.
126. Kaufmann, S. CUBA: Artificial Conviviality and User-Behaviour Analysis in Web-Feeds. PhD Thesis, Universität Hamburg, Hamburg, Germany 1969.
127. Porter, M.F. An algorithm for suffix stripping. *Program* **1980**, *14*, 130–137. [CrossRef]
128. Pearson, E.S. Bayes' theorem, examined in the light of experimental sampling. *Biometrika* **1925**, *17*, 388–442. [CrossRef]
129. Hill, B.M. Posterior distribution of percentiles: Bayes' theorem for sampling from a population. *J. Am. Stat. Assoc.* **1968**, *63*, 677–691.
130. Qu, Z.; Song, X.; Zheng, S.; Wang, X.; Song, X.; Li, Z. Improved Bayes Method Based on TF-IDF Feature and Grade Factor Feature for Chinese Information Classification. In Proceedings of the 2018 IEEE International Conference on Big Data and Smart Computing (BigComp), Shanghai, China, 15–17 January 2018; pp. 677–680.
131. Kim, S.B.; Han, K.S.; Rim, H.C.; Myaeng, S.H. Some effective techniques for naive bayes text classification. *IEEE Trans. Knowl. Data Eng.* **2006**, *18*, 1457–1466.
132. Frank, E.; Bouckaert, R.R. Naive bayes for text classification with unbalanced classes. In *European Conference on Principles of Data Mining and Knowledge Discovery*; Springer: Berlin/Heidelberg, Germany, 2006, pp. 503–510.
133. Liu, Y.; Loh, H.T.; Sun, A. Imbalanced text classification: A term weighting approach. *Expert Syst. Appl.* **2009**, *36*, 690–701. [CrossRef]
134. Soheily-Khah, S.; Marteau, P.F.; Béchet, N. Intrusion detection in network systems through hybrid supervised and unsupervised mining process-a detailed case study on the ISCX benchmark data set. *HAL* **2017**. [CrossRef]
135. Wang, Y.; Khardon, R.; Protopapas, P. Nonparametric bayesian estimation of periodic light curves. *Astrophys. J.* **2012**, *756*, 67. [CrossRef]
136. Ranjan, M.N.M.; Ghorpade, Y.R.; Kanthale, G.R.; Ghorpade, A.R.; Dubey, A.S. Document Classification Using LSTM Neural Network. *J. Data Min. Manag.* **2017**, *2*. Available online: http://matjournals.in/index.php/JoDMM/article/view/1534 (accessed on 20 April 2019).
137. Jiang, S.; Pang, G.; Wu, M.; Kuang, L. An improved K-nearest-neighbor algorithm for text categorization. *Expert Syst. Appl.* **2012**, *39*, 1503–1509. [CrossRef]
138. Han, E.H.S.; Karypis, G.; Kumar, V. Text categorization using weight adjusted k-nearest neighbor classification. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*; Springer: Berlin/Heidelberg, Germany, 2001; pp. 53–65.

139. Salton, G. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of*; Addison-Wesley: Reading, UK, 1989.
140. Sahgal, D.; Ramesh, A. On Road Vehicle Detection Using Gabor Wavelet Features with Various Classification Techniques. In Proceedings of the 14th International Conference on Digital Signal Processing Proceedings. DSP 2002 (Cat. No.02TH8628), Santorini, Greece, 1–3 July 2002, doi:10.1109/ICDSP.2002.1028263.
141. Patel, D.; Srivastava, T. Ant Colony Optimization Model for Discrete Tomography Problems. In *Proceedings of the Third International Conference on Soft Computing for Problem Solving*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 785–792.
142. Sahgal, D.; Parida, M. Object Recognition Using Gabor Wavelet Features with Various Classification Techniques. In *Proceedings of the Third International Conference on Soft Computing for Problem Solving*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 793–804.
143. Sanjay, G.P.; Nagori, V.; Sanjay, G.P.; Nagori, V. Comparing Existing Methods for Predicting the Detection of Possibilities of Blood Cancer by Analyzing Health Data. *Int. J. Innov. Res. Sci. Technol.* **2018**, *4*, 10–14.
144. Vapnik, V.; Chervonenkis, A.Y. A class of algorithms for pattern recognition learning. *Avtomat. Telemekh* **1964**, *25*, 937–945.
145. Boser, B.E.; Guyon, I.M.; Vapnik, V.N. A training algorithm for optimal margin classifiers. In Proceedings of the Fifth Annual Workshop on Computational Learning Theory, Pittsburgh, PA, USA, 27–29 July 1992; pp. 144–152.
146. Bo, G.; Xianwu, H. SVM Multi-Class Classification. *J. Data Acquis. Process.* **2006**, *3*, 017.
147. Mohri, M.; Rostamizadeh, A.; Talwalkar, A. *Foundations of Machine Learning*; MIT Press: Cambridge, MA, USA, 2012.
148. Chen, K.; Zhang, Z.; Long, J.; Zhang, H. Turning from TF-IDF to TF-IGM for term weighting in text classification. *Expert Syst. Appl.* **2016**, *66*, 245–260. [CrossRef]
149. Weston, J.; Watkins, C. *Multi-Class Support Vector Machines*; Technical Report CSD-TR-98-04; Department of Computer Science, Royal Holloway, University of London: Egham, UK, 1998.
150. Zhang, W.; Yoshida, T.; Tang, X. Text classification based on multi-word with support vector machine. *Knowl.-Based Syst.* **2008**, *21*, 879–886. [CrossRef]
151. Lodhi, H.; Saunders, C.; Shawe-Taylor, J.; Cristianini, N.; Watkins, C. Text classification using string kernels. *J. Mach. Learn. Res.* **2002**, *2*, 419–444.
152. Leslie, C.S.; Eskin, E.; Noble, W.S. The spectrum kernel: A string kernel for SVM protein classification. *Biocomputing* **2002**, *7*, 566–575.
153. Eskin, E.; Weston, J.; Noble, W.S.; Leslie, C.S. Mismatch string kernels for SVM protein classification. *Adv. Neural Inf. Process. Syst.* **2002**, *15*, 1417–1424.
154. Singh, R.; Kowsari, K.; Lanchantin, J.; Wang, B.; Qi, Y. GaKCo: A Fast and Scalable Algorithm for Calculating Gapped k-mer string Kernel using Counting. *bioRxiv* **2017**. [CrossRef]
155. Sun, A.; Lim, E.P. Hierarchical text classification and evaluation. In Proceedings of the IEEE International Conference on Data Mining (ICDM 2001), San Jose, CA, USA, 29 November–2 December 2001; pp. 521–528.
156. Sebastiani, F. Machine learning in automated text categorization. *ACM Comput. Surv. (CSUR)* **2002**, *34*, 1–47. [CrossRef]
157. Maron, O.; Lozano-Pérez, T. A framework for multiple-instance learning. *Adv. Neural Inf. Process. Syst.* **1998**, *10*, 570–576.
158. Andrews, S.; Tsochantaridis, I.; Hofmann, T. Support vector machines for multiple-instance learning. *Adv. Neural Inf. Process. Syst.* **2002**, *15*, 577–584.
159. Karamizadeh, S.; Abdullah, S.M.; Halimi, M.; Shayan, J.; Javad Rajabi, M. Advantage and drawback of support vector machine functionality. In Proceedings of the 2014 International Conference on Computer, Communications, and Control Technology (I4CT), Langkawi, Malaysia, 2–4 September 2014; pp. 63–65.
160. Guo, G. Soft biometrics from face images using support vector machines. In *Support Vector Machines Applications*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 269–302.
161. Morgan, J.N.; Sonquist, J.A. Problems in the analysis of survey data, and a proposal. *J. Am. Stat. Assoc.* **1963**, *58*, 415–434. [CrossRef]
162. Safavian, S.R.; Landgrebe, D. A survey of decision tree classifier methodology. *IEEE Trans. Syst. Man Cybern.* **1991**, *21*, 660–674. [CrossRef]

163. Magerman, D.M. Statistical decision-tree models for parsing. In Proceedings of the 33rd Annual Meeting on Association for Computational Linguistics, Cambridge, MA, USA, 26–30 June 1995; Association for Computational Linguistics: Stroudsburg, PA, USA, 1995; pp. 276–283.
164. Quinlan, J.R. Induction of decision trees. *Mach. Learn.* **1986**, *1*, 81–106. [CrossRef]
165. De Mántaras, R.L. A distance-based attribute selection measure for decision tree induction. *Mach. Learn.* **1991**, *6*, 81–92. [CrossRef]
166. Giovanelli, C.; Liu, X.; Sierla, S.; Vyatkin, V.; Ichise, R. Towards an aggregator that exploits big data to bid on frequency containment reserve market. In Proceedings of the 43rd Annual Conference of the IEEE Industrial Electronics Society (IECON 2017), Beijing, China, 29 October–1 November 2017; pp. 7514–7519.
167. Quinlan, J.R. Simplifying decision trees. *Int. J. Man-Mach. Stud.* **1987**, *27*, 221–234. [CrossRef]
168. Jasim, D.S. Data Mining Approach and Its Application to Dresses Sales Recommendation. Available online: https://www.researchgate.net/profile/Dalia_Jasim/publication/293464737_main_steps_for_doing_data_mining_project_using_weka/links/56b8782008ae44bb330d2583/main-steps-for-doing-data-mining-project-using-weka.pdf (accessed on 23 April 2019).
169. Ho, T.K. Random decision forests. In Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, Canada 14–16 August 1995; Volume 1, pp. 278–282. [CrossRef]
170. Breiman, L. *Random Forests*; UC Berkeley TR567; University of California: Berkeley, CA, USA, 1999.
171. Wu, T.F.; Lin, C.J.; Weng, R.C. Probability estimates for multi-class classification by pairwise coupling. *J. Mach. Learn. Res.* **2004**, *5*, 975–1005.
172. Bansal, H.; Shrivastava, G.; Nhu, N.; Stanciu, L. *Social Network Analytics for Contemporary Business Organizations*; IGI Global: Hershey, PA, USA, 2018.
173. Sutton, C.; McCallum, A. An introduction to conditional random fields. *Found. Trends® Mach. Learn.* **2012**, *4*, 267–373. [CrossRef]
174. Vail, D.L.; Veloso, M.M.; Lafferty, J.D. Conditional random fields for activity recognition. In Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems, Honolulu, HI, USA, 14–18 May 2007; p. 235.
175. Chen, T.; Xu, R.; He, Y.; Wang, X. Improving sentiment analysis via sentence type classification using BiLSTM-CRF and CNN. *Expert Syst. Appl.* **2017**, *72*, 221–230. [CrossRef]
176. Sutton, C.; McCallum, A. An introduction to conditional random fields for relational learning. In *Introduction to Statistical Relational Learning*; MIT Press: Cambridge, MA, USA, 2006; Volume 2.
177. Tseng, H.; Chang, P.; Andrew, G.; Jurafsky, D.; Manning, C. A conditional random field word segmenter for sighan bakeoff 2005. In Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing, Jeju Island, Korea, 14–15 October 2005.
178. Nair, V.; Hinton, G.E. Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th International Conference on Machine Learning (ICML-10), Haifa, Israel, 21–24 June 2010; pp. 807–814.
179. Sutskever, I.; Martens, J.; Hinton, G.E. Generating text with recurrent neural networks. In Proceedings of the 28th International Conference on Machine Learning (ICML-11), Bellevue, WA, USA, 28 June–2 July 2011; pp. 1017–1024.
180. Mandic, D.P.; Chambers, J.A. *Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures and Stability*; Wiley Online Library: Hoboken, NJ, USA, 2001.
181. Bengio, Y.; Simard, P.; Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Netw.* **1994**, *5*, 157–166. [CrossRef]
182. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
183. Graves, A.; Schmidhuber, J. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Netw.* **2005**, *18*, 602–610. [CrossRef] [PubMed]
184. Pascanu, R.; Mikolov, T.; Bengio, Y. On the difficulty of training recurrent neural networks. *ICML* **2013**, *28*, 1310–1318.
185. Chung, J.; Gulcehre, C.; Cho, K.; Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv* **2014**, arXiv:1412.3555.
186. Jaderberg, M.; Simonyan, K.; Vedaldi, A.; Zisserman, A. Reading text in the wild with convolutional neural networks. *Int. J. Comput. Vis.* **2016**, *116*, 1–20. [CrossRef]

187. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
188. Scherer, D.; Müller, A.; Behnke, S. Evaluation of pooling operations in convolutional architectures for object recognition. In Proceedings of the Artificial Neural Networks–ICANN 2010, Thessaloniki, Greece, 15–18 September 2010; pp. 92–101.
189. Johnson, R.; Zhang, T. Effective use of word order for text categorization with convolutional neural networks. *arXiv* **2014**, arXiv:1412.1058.
190. Hinton, G.E. Training products of experts by minimizing contrastive divergence. *Neural Comput.* **2002**, *14*, 1771–1800. [CrossRef]
191. Hinton, G.E.; Osindero, S.; Teh, Y.W. A fast learning algorithm for deep belief nets. *Neural Comput.* **2006**, *18*, 1527–1554. [CrossRef]
192. Mohamed, A.R.; Dahl, G.E.; Hinton, G. Acoustic modeling using deep belief networks. *IEEE Trans. Audio Speech Lang. Process.* **2012**, *20*, 14–22. [CrossRef]
193. Yang, Z.; Yang, D.; Dyer, C.; He, X.; Smola, A.J.; Hovy, E.H. Hierarchical Attention Networks for Document Classification. In Proceedings of the HLT-NAACL, San Diego, CA, USA, 12–17 June 2016; pp. 1480–1489.
194. Seo, P.H.; Lin, Z.; Cohen, S.; Shen, X.; Han, B. Hierarchical attention networks. *arXiv* **2016**, arXiv:1606.02393.
195. Bottou, L. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 177–186.
196. Tieleman, T.; Hinton, G. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA Neural Netw. Mach. Learn.* **2012**, *4*, 26–31.
197. Kingma, D.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
198. Duchi, J.; Hazan, E.; Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.* **2011**, *12*, 2121–2159.
199. Zeiler, M.D. ADADELTA: An adaptive learning rate method. *arXiv* **2012**, arXiv:1212.5701.
200. Wang, B.; Xu, J.; Li, J.; Hu, C.; Pan, J.S. Scene text recognition algorithm based on faster RCNN. In Proceedings of the 2017 First International Conference on Electronics Instrumentation & Information Systems (EIIS), Harbin, China, 3–5 June 2017; pp. 1–4.
201. Zhou, C.; Sun, C.; Liu, Z.; Lau, F. A C-LSTM neural network for text classification. *arXiv* **2015**, arXiv:1511.08630.
202. Shwartz-Ziv, R.; Tishby, N. Opening the black box of deep neural networks via information. *arXiv* **2017**, arXiv:1703.00810.
203. Gray, A.; MacDonell, S. Alternatives to Regression Models for Estimating Software Projects. Available online: https://www.researchgate.net/publication/2747623_Alternatives_to_Regression_Models_for_Estimating_Software_Projects (accessed on 23 April 2019).
204. Shrikumar, A.; Greenside, P.; Kundaje, A. Learning important features through propagating activation differences. *arXiv* **2017**, arXiv:1704.02685.
205. Anthes, G. Deep learning comes of age. *Commun. ACM* **2013**, *56*, 13–15. [CrossRef]
206. Lampinen, A.K.; McClelland, J.L. One-shot and few-shot learning of word embeddings. *arXiv* **2017**, arXiv:1710.10280.
207. Severyn, A.; Moschitti, A. Learning to rank short text pairs with convolutional deep neural networks. In Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, 9–13 August 2015; pp. 373–382.
208. Gowda, H.S.; Suhil, M.; Guru, D.; Raju, L.N. Semi-supervised text categorization using recursive K-means clustering. In *International Conference on Recent Trends in Image Processing and Pattern Recognition*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 217–227.
209. Kowsari, K. Investigation of Fuzzyfind Searching with Golay Code Transformations. Ph.D. Thesis, Department of Computer Science, The George Washington University, Washington, DC, USA, 2014.
210. Kowsari, K.; Yammahi, M.; Bari, N.; Vichr, R.; Alsaby, F.; Berkovich, S.Y. Construction of fuzzyfind dictionary using golay coding transformation for searching applications. *arXiv* **2015**, arXiv:1503.06483.
211. Chapelle, O.; Zien, A. Semi-Supervised Classification by Low Density Separation. In Proceedings of the AISTATS, The Savannah Hotel, Barbados, 6–8 January 2005; pp. 57–64.
212. Nigam, K.; McCallum, A.; Mitchell, T. Semi-supervised text classification using EM. In *Semi-Supervised Learning*; MIT Press: Cambridge, MA, USA, 2006; pp. 33–56.

213. Shi, L.; Mihalcea, R.; Tian, M. Cross language text classification by model translation and semi-supervised learning. In Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, Cambridge, MA, USA, 9–11 October 2010; Association for Computational Linguistics: Stroudsburg, PA, USA, 2010; pp. 1057–1067.
214. Zhou, S.; Chen, Q.; Wang, X. Fuzzy deep belief networks for semi-supervised sentiment classification. *Neurocomputing* **2014**, *131*, 312–322. [CrossRef]
215. Yang, Y. An evaluation of statistical approaches to text categorization. *Inf. Retr.* **1999**, *1*, 69–90. [CrossRef]
216. Lever, J.; Krzywinski, M.; Altman, N. Points of significance: Classification evaluation. *Nat. Methods* **2016**, *13*, 603–604 . [CrossRef]
217. Manning, C.D.; Raghavan, P.; Schütze, H. Matrix decompositions and latent semantic indexing. In *Introduction to Information Retrieval*; Cambridge University Press: Cambridge, UK, 2008; pp. 403–417.
218. Tsoumakas, G.; Katakis, I.; Vlahavas, I. Mining multi-label data. In *Data Mining and Knowledge Discovery Handbook*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 667–685.
219. Yonelinas, A.P.; Parks, C.M. Receiver operating characteristics (ROCs) in recognition memory: A review. *Psychol. Bull.* **2007**, *133*, 800. [CrossRef]
220. Japkowicz, N.; Stephen, S. The class imbalance problem: A systematic study. *Intell. Data Anal.* **2002**, *6*, 429–449. [CrossRef]
221. Bradley, A.P. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recogn.* **1997**, *30*, 1145–1159. [CrossRef]
222. Hand, D.J.; Till, R.J. A simple generalisation of the area under the ROC curve for multiple class classification problems. *Mach. Learn.* **2001**, *45*, 171–186. [CrossRef]
223. Wu, H.C.; Luk, R.W.P.; Wong, K.F.; Kwok, K.L. Interpreting tf-idf term weights as making relevance decisions. *ACM Trans. Inf. Syst. (TOIS)* **2008**, *26*, 13. [CrossRef]
224. Rezaeinia, S.M.; Ghodsi, A.; Rahmani, R. Improving the Accuracy of Pre-trained Word Embeddings for Sentiment Analysis. *arXiv* **2017**, arXiv:1711.08609.
225. Sharma, A.; Paliwal, K.K. Fast principal component analysis using fixed-point algorithm. *Pattern Recogn. Lett.* **2007**, *28*, 1151–1155. [CrossRef]
226. Putthividhya, D.P.; Hu, J. Bootstrapped named entity recognition for product attribute extraction. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, Edinburgh, UK, 27–31 July 2011; Association for Computational Linguistics: Stroudsburg, PA, USA, 2011; pp. 1557–1567.
227. Banerjee, M. *A Utility-Aware Privacy Preserving Framework for Distributed Data Mining with Worst Case Privacy Guarantee*; University of Maryland: Baltimore County, MD, USA, 2011.
228. Chen, J.; Yan, S.; Wong, K.C. Verbal aggression detection on Twitter comments: Convolutional neural network for short-text sentiment analysis. *Neural Comput. Appl.* **2018**, 1–10. [CrossRef]
229. Zhang, X.; Zhao, J.; LeCun, Y. Character-level convolutional networks for text classification. *Adv. Neural Inf. Process. Syst.* **2015**, *28*, 649–657.
230. Schütze, H.; Manning, C.D.; Raghavan, P. *Introduction to Information Retrieval*; Cambridge University Press: Cambridge, UK, 2008; Volume 39.
231. Hoogeveen, D.; Wang, L.; Baldwin, T.; Verspoor, K.M. Web forum retrieval and text analytics: A survey. *Found. Trends® Inf. Retr.* **2018**, *12*, 1–163. [CrossRef]
232. Dwivedi, S.K.; Arya, C. Automatic Text Classification in Information retrieval: A Survey. In Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies, Udaipur, India, 4–5 March 2016; p. 131.
233. Jones, K.S. Automatic keyword classification for information retrieval. *Libr. Q.* **1971**, *41*, 338–340. [CrossRef]
234. O'Riordan, C.; Sorensen, H. Information filtering and retrieval: An overview. In Proceedings of the 16th Annual International Conference of the IEEE, Atlanta, GA, USA, 28–31 October 1997; p. 42.
235. Buckley, C. *Implementation of the SMART Information Retrieval System*; Technical Report; Cornell University: Ithaca, NY, USA, 1985.
236. Pang, B.; Lee, L. Opinion mining and sentiment analysis. *Found. Trends® Inf. Retr.* **2008**, *2*, 1–135. [CrossRef]
237. Liu, B.; Zhang, L. A survey of opinion mining and sentiment analysis. In *Mining Text Data*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 415–463.

238. Pang, B.; Lee, L.; Vaithyanathan, S. Thumbs up?: Sentiment classification using machine learning techniques. In *ACL-02 Conference on Empirical Methods in Natural Language Processing*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2002; Volume 10, pp. 79–86.
239. Aggarwal, C.C. Content-based recommender systems. In *Recommender Systems*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 139–166.
240. Pazzani, M.J.; Billsus, D. Content-based recommendation systems. In *The Adaptive Web*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 325–341.
241. Sumathy, K.; Chidambaram, M. Text mining: Concepts, applications, tools and issues—An overview. *Int. J. Comput. Appl.* **2013**, *80*, 29–32.
242. Heidarysafa, M.; Kowsari, K.; Barnes, L.E.; Brown, D.E. Analysis of Railway Accidents' Narratives Using Deep Learning. In Proceedings of the 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), Orlando, FL, USA, 17–20 December 2018.
243. Mani, I. *Advances in Automatic Text Summarization*; MIT Press: Cambridge, MA, USA, 1999.
244. Cao, Z.; Li, W.; Li, S.; Wei, F. Improving Multi-Document Summarization via Text Classification. In Proceedings of the AAAI, San Francisco, CA, USA, 4–9 February 2017; pp. 3053–3059.
245. Lauría, E.J.; March, A.D. Combining Bayesian text classification and shrinkage to automate healthcare coding: A data quality analysis. *J. Data Inf. Qual. (JDIQ)* **2011**, *2*, 13. [CrossRef]
246. Zhang, J.; Kowsari, K.; Harrison, J.H.; Lobo, J.M.; Barnes, L.E. Patient2Vec: A Personalized Interpretable Deep Representation of the Longitudinal Electronic Health Record. *IEEE Access* **2018**, *6*, 65333–65346. [CrossRef]
247. Trieschnigg, D.; Pezik, P.; Lee, V.; De Jong, F.; Kraaij, W.; Rebholz-Schuhmann, D. MeSH Up: Effective MeSH text classification for improved document retrieval. *Bioinformatics* **2009**, *25*, 1412–1418. [CrossRef] [PubMed]
248. Ofoghi, B.; Verspoor, K. Textual Emotion Classification: An Interoperability Study on Cross-Genre data sets. In *Australasian Joint Conference on Artificial Intelligence*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 262–273.
249. Pennebaker, J.; Booth, R.; Boyd, R.; Francis, M. *Linguistic Inquiry and Word Count: LIWC2015*; Pennebaker Conglomerates: Austin, TX, USA, 2015. Available online: www.LIWC.net (accessed on 10 January 2019).
250. Paul, M.J.; Dredze, M. Social Monitoring for Public Health. *Synth. Lect. Inf. Concepts Retr. Serv.* **2017**, *9*, 1–183, doi:10.2200/ S00791ED1V01Y201707ICR060. [CrossRef]
251. Yu, B.; Kwok, L. Classifying business marketing messages on Facebook. In Proceedings of the Association for Computing Machinery Special Interest Group on Information Retrieval, Bejing, China, 24–28 July 2011.
252. Kang, M.; Ahn, J.; Lee, K. Opinion mining using ensemble text hidden Markov models for text classification. *Expert Syst. Appl.* **2018**, *94*, 218–227. [CrossRef]
253. Turtle, H. Text retrieval in the legal world. *Artif. Intell. Law* **1995**, *3*, 5–54. [CrossRef]
254. Bergman, P.; Berman, S.J. *Represent Yourself in Court: How to Prepare & Try a Winning Case*; Nolo: Berkeley, CA, USA, 2016.