Multimedia Tools and Applications (2023) 82:9761–9783
https://doi.org/10.1007/s11042-022-12538-3

1222: INTELLIGENT MULTIMEDIA DATA ANALYTICS AND COMPUTING

The importance of Term Weighting in semantic understanding of text: A review of techniques

R. N. Rathi[1] · **A. Mustafi**[1]

Received: 8 January 2021 / Revised: 31 December 2021 / Accepted: 30 January 2022 /
Published online: 13 April 2022

Abstract
In this paper we review a wide spectrum of techniques which have been proposed in literature to enable acceptable recognition of language and text by machines. We discuss many techniques which have been proposed by researchers in the field of term weighting and explore the mathematical foundations of these methods. Term weighting schemes have broadly been classified as supervised and statistical methods and we present numerous examples from both categories to highlight the difference in approaches between the two broad categories. We pay particular attention to the Vector Space Model and its variants which form the basis of many of the other methods which have been discussed in the paper.

Keywords Term weighting · Word embedding · Term weighting techniques

1 Introduction

The ability to communicate and the use of languages has long been considered the greatest boon that has been bestowed on the humans race in the process of evolution. Our ability to use languages to convey our feelings and the fact that this can be interpreted in an unambiguous manner in the vast majority of instances has set the humans race apart from all other forms of life.

However, this seemingly trivial power has long been considered one of the greatest challenges in the field of computational intelligence. The ability of a computational device to sense and interpret language is still an open problem in the field of computational research. While it is true that giant strides have been made in our endeavour to allow computers to independently understand human languages it is equally true that we are a long way off from the capabilities demonstrated by the human brain in this aspect. One of the issues that hinders the effective use of computers in the field of language and text understanding is the

✉ R. N. Rathi
raunakrathi.rathi@gmail.com

A. Mustafi
abhijit@bitmesra.ac.in

[1] Birla Institute of Technology, Mesra, India

fact that computers thrive in the manipulation of numbers while the representation of languages in a computer is usually done in the form of *strings*. While notionally strings are also stored as numbers these numbers are just symbolic representations and usually do not capture the semantic flavour of the language or the text.

In the last few decades computer scientists have developed several schemes to represent text using numeric systems that capture the effective semantic behind the symbols being used. Our new found ability to represent text in more numerically precise ways has subsequently paved the way for the investigation of a vast array of computational problems in the domain of text processing. These involve problems like topic modeling, keyword extraction, key-phrase extraction, document summarization, document clustering and many more.

The key challenge that needs to be resolved by any text processing algorithm is the ability to look at the tokens in groups rather than in isolation. It is common knowledge that a single word or token in isolation does not carry much information for the recipient and the human brain specializes in processing tokens with reference to the context set by its preceding (and in many cases the subsequent) tokens. Thus, our text processing systems need to be endowed with similar capabilities if they are to mimic the brain's text processing and interpretation capabilities.

Salton et al. were one of the earliest to recognize the power offered by computing devices which could be used effectively to represent text for semantic understanding [64]. Their model of representing a collection of documents as a set of vectors paved the way for many such models, each of which had the capability to effectively compare multiple documents for similarity or dissimilarity [63]. However, these models suffered from the fact that the vectors were usually sparse and often difficult to leverage for meaningful computations when the documents were lengthy. Nonetheless, the Vector Space Model was the first major step towards a mathematically tenable way of representing text documents and fuelled the interest in this field.

1.1 Motivation

Our motivation for the present work is due to the huge increase in demand for acceptable text processing methods and algorithms which are also efficient. A comprehensive review like this allows the research community to understand, analyze and make an informed choice as to the method that is best suited for a particular problem that needs to be solved. By providing a review of the advantages and disadvantages of the various methods used in the field of text processing, we hope to provide the reader a deeper insight to the techniques and tools that are currently available for efficient text processing.

1.2 Key contributions

The main contribution of the paper is to provide the interested readers with a rigorous understanding of the working of the major algorithms used for weighting tokens in text processing. We also demonstrate the result of the application of some of these papers for the purpose of comparison.

1.3 Organization of the paper

The paper is organized as follows: In Section 2 we discussed about vector space model. The Sections 3, 4, and 5 contributes about term weighting scheme: binary weighting, term

frequency and term inverse document frequency (tf-idf) respectively. In Section 6 we mentioned some modified tf-idf models and then in Section 7 represent about the vector based term weighting. All performance analysis is discussed in Section 8 followed by discussion, recent advancement and conclusion in Sections 9, 10, and 11 respectively.

In this paper we first started journey with vector space model to have a general idea about vector space and how most of the algorithm behaves. Then we have represented various term weighting and variation of term–frequency in statistical approach of term weighting. Then we shifted paper focus on supervised following unsupervised approaches for text classification. Then the paper also present vector based term weighting following the discussion of how different approaches are important for various problems.

2 Vector space model

The Vector Space Model (VSM) proposed by Salton et al. in the year 1975 was a breakthrough in the field of term weighting in documents. It proposed a mathematically stringent algorithm which accounted not only for the occurrences of a token in a document but also considered the frequency of the token in a corpus, to assign weights to it. The inherent simplicity of the algorithm coupled with the use of distance measures e.g., the cosine similarity metric quickly established the VSM as the algorithm of choice in the context of term weighting.

Consider a corpus containing "D" documents and a vocabulary of "t" terms. The vocabulary is considered as the set of all relevant tokens collected from all the documents in the corpus. It is to be remembered that the relevant tokens are usually a subset of all the raw tokens present in the corpus obtained by removing stop–words, performing case transformations, stemming and other data cleaning activities. Each document can then be represented in a "t" dimensional vector space having order D x t. Each coefficient in this matrix is an indicator of the relevance of the token t_j in document D_i, based on some notion of numerosity and relevance of the token not only in the document but in the entire corpus. The VSM is a great tool to quantify the semantic similarity between two different documents, because of its amenability to distance metrics computations particularly the cosine similarity metrics.

The "cosine similarity" between two documents can be represented as

$$cos\theta = \frac{\mathbf{a}.\mathbf{b}}{||\mathbf{a}||||\mathbf{b}||} \tag{1}$$

The formula could be used both for finding similarity between the document and finding the similarity between the query and the document. To find similarity between documents a and b are considered to be d_1 and d_2, two different documents from the set D. $||\mathbf{a}||$ and $||\mathbf{b}||$ is calculated using $\sqrt{\sum_{i=0}^{n} w_i^2}$. The calculation is specified as the sum of every weighted term *"w"* in a vocabulary of all documents where *"n"* is the maximum number of terms. Every term's weight is non-zero if it is present in the document.

Similarly for information retrieval tasks *"a"* and *"b"* is considered as the document and query vectors. For e.g. consider two different documents *d1: "WHO declared COVID–19 a pandemic." d2: "Corona affects, most part of the world and maximum countries are under lock–down"*. A query such as *q1: "Why 2020 was under lock–down?"* is considered to be best answered by the document whose vector direction is most favourably inclined to the direction of the query vector in the vector space. The general notion of similarity between the documents and the query are easily captured by the cosine similarity metrics.

However, the distance metrics is heavily influenced by the choice of the term weighting scheme chosen to populate the coefficients of the Vector Space,

Polettini in his works suggested that there are three major components for term weighting [53]

1. The global weight g_i weight of i^{th} term.
2. The local weight of the term represented as $t_{i,j}$ where i is the term in j^{th} document.
3. The Normalization factor represented as d_j for the j^{th} document.

$$a_{i,j} = g_i * t_{i,j} * d_j \tag{2}$$

Based on these observations several term weightings schemes have been utilized in the VSM. The most notable of which is the *bag-of-word* scheme [2, 18, 32]. The BOW scheme represents each term in a document vector using its numerical count of occurrence in the document. While easy to compute the scheme fails to capture the correlation between different terms present in a single document. This greatly depreciates the scheme's ability to capture the semantic sense of the document. However as demonstrated by [48, 77] the scheme is still a good alternative in text processing tasks like phrase extraction. To overcome the shortcomings of the BOW scheme a graph based term weighting scheme was proposed by [36, 40, 75] which used the notion of edge weights to capture the relationship between different terms in a document. By representing the BOW in a graphical form, edge weight helps to maintain the coherent relationship between words in a document. These edge weights are then converted into the coefficients to be used in the VSM matrix.

While the original VSM continues to be relevant and still merits great theoretical dissection, over time the model has shown severe drawbacks in terms of applicability. Most notably the vector representation used in the model tends to create sparse vectors which have fallen out of favour with modern learning algorithms like deep neural networks. Attempts to improve the performance of the model using techniques like one hot encoding and vector weighting schemes have only met with limited success in extremely large datasets.

A few other notable term weighting schemes are discussed in the following sections:

3 Binary weighting

The simplest of all term weighting scheme is binary weighting [65, 73]. In this scheme each term is represented by a 1 if it is present in a document and by a 0 if it is absent in the document. While the binary weighting scheme is difficult to beat in terms of simplicity, it loses all indication of the numerosity of a term in a document. Consequently, it is not recommended for tasks where the frequency of occurrence a term in a document is important. The method also does not capture the semantic correlation between different terms in a document but merely registers their presence or absence. However, it is interesting to note that the binary weighting scheme does lay the foundation for the much more useful one hot encoding method widely used in different machine learning applications. [1, 76]. Mathematically, the scheme is represented as

$$f(t) = \begin{cases} 1, & \text{if } t > 0 \\ 0, & \text{if } t = 0 \end{cases}$$

where t is the count of a term in a document.

4 Term frequency

The most popular single document weighting scheme is term frequency [16, 22]. The scheme calculates the ratio between the number of times a term appears in a document to the number of times the most frequent term appears in the document. Thus, the term frequency ranges in value from 0 to 1. The term frequency is an easy metrics to calculate and provides an accurate representation of the document in terms of keywords. However, it still falls short of capturing the semantic correlation between the different terms in the document. The term frequency *tf* of a term i in a document is mathematically defined as:

$$tf_{i,d} = \frac{f_d(i)}{\max\limits_{\{w \epsilon d\}} f_d(w)} \tag{3}$$

where f_d represent frequency in d.

To analyse the number of occurrences of a term in a document the Zipf's law's [52, 84] contribution is worth noting. The law state that the frequency of a term is inversely proportional to its occurrence or rank in frequency table. According to the law the most frequent word occurs twice as frequently as the second most occurring word and the same is true for all other ranks. This paradigm helps us to understand the structure of a text document and the increased emphasis on the keywords. It is also noteworthy that the Zipf's Law is applicable to almost all known languages in the world.

Many researchers have argued about the validity of the assertion that a more frequent term conveys greater knowledge about the semantic content of a document. It has often been stated in literature that a term that appears too frequently in a document often communicates non–specific information which does not operate as a discerning factor in the understanding of the document [5, 33, 57, 62]. To counter the excessive importance attached to the numerosity of a term in the term frequency scheme a new scheme called the *term frequency–inverse document frequency* was proposed.

5 Term frequency–inverse document frequency

As discussed in the previous section the use of the term frequency as a weighting scheme can unnecessarily over emphasize the semantic contribution of frequently occurring terms. This can compromise the ability of an algorithm to infer the semantic meaning of the document to a great extent. To overcome this limitation, the *term frequency inverse document frequency (Tf-Idf)* was proposed. The scheme has found widespread acceptance and is still the method of choice in many applications even after almost fifty years after it has been first proposed [6, 11, 23, 34, 55, 82].

The Tf-Idf scheme offsets the contribution of a frequent term by taking into account the propensity of the term in the entire corpus. Thus, a term that occurs in almost all the documents is considered to have lesser discerning ability then a term which occurs in only a few documents. Mathematically the term weight associated with each term in a document is given as,

$$W_{i,j} = tf_{i,j} \times log\left(\frac{N}{df_i}\right) \tag{4}$$

where,

$tf_{i,j}$ - number of occurrence of term i in document j.

df_i - number of documents containing i.

The log function used in the equation is introduced to boost the impact of the document frequency terms, which tend to be much smaller compared to the term frequency values. Thus, it is just a mathematical convenience and only acts as a scaling factor.

6 Modified variants of TF-IDF

In the following sections we discuss various modification and improvements proposed to improve the performance of the tf-idf scheme [61]. These modifications can be broadly categorized as Supervised and Unsupervised methods.

6.1 Supervised approach

Supervised learning is an approach in machine learning (ML) to map input-output pairs available for training. The data is labelled and most of the algorithms are used to solve classification problems. While supervised approaches can provide excellent results there is also a possibility that these algorithms can be biased if trained on a representative dataset.

6.1.1 Supervised term weighting

Debole and Sebastiani [19] first proposed Supervised Term Weighting (STW). The model performs the term weight calculation by extracting certain information from the training corpus. It has three variations, i.e., TF-CHI (term frequency chi-square), TF-IG (term frequency information gain), and TF-GR (term frequency gain ratio) [54]. The basic idea underlying all these variations is to replace the IDF factor with an alternate global factor.

To reduce the length of the global factor, which depends on the size of the corpus, a reducing factor is also used. The model applies two policies. First, a local policy in which every term's weight(T') is calculated independently for each document category. Second, a global policy is applied, in which the term is calculated in a single set using a single score derived from the individual score. As the size of vocabulary in the training set is too large, represent as T and $|T'| \ll |T|$. As size of every term, is mostly based on size of the document. So, to control the size of the vector reduction factor(ξ) is also introduced [19]:

$$\xi = \frac{|T| - |T'|}{|T|} \tag{5}$$

Some other popular variants of tf-idf such as probability based approach, Mutual Information, Balanced relative frequency and odds ratio proposed by Liu et al. [37, 37], Sebastiani et al. [69], Tsai et al. [72] and Mladeni et al. [45] are explained and analyzed by Altınçay et al. in [3].

6.1.2 Confidence weight

STW was further used to find another weight variant known as confidence weight for the task of classifying documents [71]. The weighting model is a n base classifier. In this model ($\widetilde{p}$) Willson proportion [79] estimate is used for deriving the proportion of a term in a document. Willson equation is represented as:

$$\widetilde{p} = \frac{x_i + 1.96}{n + 3.84} \tag{6}$$

Where x_i is the number of documents including the term and n as number of documents. The strength of every term is calculated (maxstr(t)) and confidence weight of term t in document d is represented as [71]:

$$ConfWeight_{t,d} = log(tf_{t,d} + 1) * maxstr(t) \tag{7}$$

During training the strength of every term maxstr(t) is mainly categorized as positive and negative category. The strength of term t is calculated for positive category:

$$str_t = \begin{cases} log_2(2MinPosRelFreq), & if MinPos > MaxPos \\ 0, & otherwise \end{cases} \tag{8}$$

The weight of MinPosRelFreq is calculated from the lower interval *"MinPos"* to higher interval *"MaxPos"*. While the MinPosRelFreq is calculated defined as:

$$MinPosRelFreq = \frac{MinPos}{MinPos + MaxPos} \tag{9}$$

6.1.3 Relevance frequency

Over the years, many incremental changes have been proposed in the STW by various researchers. One of the major observations with regards to the STW was the fact that the STW was not consistent and tf-idf almost always outperformed it as shown in [20] and [30].

The most consistent approach at that time was proposed by Lan and et al. [30] using the concept of relevance frequency. The term weighting scheme was developed for text classification as an extension of the STW. Considering multi-label documents in a corpus, the scheme mainly labels documents as positive or negative. Documents in a corpus are subdivided into 4 categories a,b,c and d, as listed below :

- a is number of documents that contains the term in positive category.
- b is number of documents that do not contain the term in positive category
- c is number of documents that contains the term in negative category
- d is number of documents that do not contain the term in negative category

The paper also mentions that a,b,c$\ll d$ which is usually the case for most real datasets. The authors also modified many equations such as idf, idf_prob, etc to suit their formulations. Various cases were also discussed to show the shortcomings of idf and idf_prob in term classification. Consequently, relevance frequency (rf) was proposed as [30]:

$$rf = log\left(2 + \frac{a}{max(1, c)}\right) \tag{10}$$

Term weighting is finally calculated using tf*rf and the results were demonstrated on various datasets in comparison to tf-idf.

6.1.4 Inverse gravity moment

Inverse gravity moment (IGM) was introduced by Chen et al. [15]. In their paper they presented the journey from tf-idf to tf-igm and introduced a new method for computation of IGM. The igm model is finally combined with tf. The model is based on frequency (f) of a term (k) in a document based on various class (r) specified as gravity (F_{kr}). In this model frequency is arranged in descending order, that often leads to a biased inter-class distribution. However, in case the frequency in uniformly distributed then the inter-class distribution will be concentrated on a central point known as the gravity center. The rank r is calculated

as a distance from the origin (0) of specific gravity (f_{kr}) of a term. The product of gravity and rank is known as "gravity moment" and the terms are presented in descending order of gravity moment. f_{k1} is referred as highest–class specific frequency. The method highlighted the fact that the term frequency is directly proportional to inter-class distribution of a term in a corpus. The IGM is mathematically presented as [15]:

$$igm(t_k) = \frac{1}{\sum_{r=1}^{m} \frac{f_{kr}}{\max_{1 \le i \le m} (f_{ki})} * r} \tag{11}$$

6.2 Unsupervised term weighting

Unsupervised learning is an approach in ML to learn the pattern in data from an unlabeled corpus or data. The approach is mostly used for embedded weighting, language conversion or clustering. To train such data most of the time weighted input is provided, and some error correction mechanism is used to improve the efficiency of the algorithm. Unsupervised method can also suffer from bias in case a non–representative dataset is provided as input.

6.2.1 Relevance weighting

In the early days of information retrieval term weighting was an extremely challenging task [39]. The main problem was to find a suitable numeric representation for tokens which were essentially strings in a manner that the numerical representations were able to capture the relevance of the term in the document. Additionally, the problem was further complicated by the fact that if many terms ended up having the same relevance factor then it hampered the working of query processing systems. To overcome these challenges Sparck Johnes formulated a term weight using the equation [56]:

$$w = -log\left(\frac{n}{N}\right) \tag{12}$$

Where N is the count of documents in the set and n as count of documents containing term t. Considering this as a base line function the model is further decomposed into four functions. The first function (f^1) is the ratio of relevant documents in which term t occurs to the ratio of all documents in which t occurs. This is specified as [56]:

$$f^1 = log\frac{\left(\frac{r}{R}\right)}{\left(\frac{n}{N}\right)} \tag{13}$$

Where R is the as count of documents for request q and r is count of relevant documents containing term t.

The second function (f^2) is the ratio of relevant documents to the ratio of non-relevant documents specified as [56]:

$$f^2 = log\frac{\left(\frac{r}{R}\right)}{\left(\frac{n-r}{N-R}\right)} \tag{14}$$

The third equation (f^3) is the ratio of the count of relevant documents in which the term t does not occur and "collection of odds" for term t. The function is specified as [56]:

$$f^3 = log\frac{\left(\frac{r}{R-r}\right)}{\left(\frac{n}{N-n}\right)} \tag{15}$$

The fourth function (f^4) is the ratio of "relevance odds" and "non-relevance odds" specified as [56]:

$$f^4 = log \frac{\left(\frac{r}{R-r}\right)}{\left(\frac{n-r}{N-n-R+r}\right)} \tag{16}$$

The authors have also explained the reason for their choice of these four functions with examples from real datasets. The paper also proposes two "Independent Assumptions" and two "order principles" to further improve the term weightings. Thus, the proposed model was considered to be quite robust and remained a much favoured model for many years.

6.2.2 OKAPI BM25

One of the most dominating term weighting technique based on relevance weighting [56] in the field of information retrieval is Okapi Best Matching 25 (Okapi BM25). Okapi refers to the name of the first system on which the method was executed. The term weighting is query based where Q is set of keywords ($q_1, q_2, \ldots, q_n$). The method is based on the basic principle of bag-of-words and assumes that the presence of each term is independent from the presence of other terms. Unfortunately, this assumption takes away from the methods capability to understand the semantic meaning of a document. Calculating the term frequency of each term t for a Document D ($tf_{t,D}$) and using the IDF function explained in Section 5 the BM25 function is defined as [58]:

$$S_{Q,D} = \sum_{i=1}^{n} \frac{tf_{q_i,D} * (k_1 + 1)}{tf_{q_i,D} + k_1 * \left(1 - b + b * \frac{|D|}{avgdl}\right)} \tag{17}$$

Where avgdl is the average length of document and k_1 and b are free parameters. The specified range of k_1 is $k_1 \in [1.2, 2.0]$ and b is 0.75. The IDF function has certain drawbacks. most notably the possibility of negative values for term weights which makes computations very difficult. To further improve the performance certain measures were proposed [27].

- The results were computed using the floor function to avoid the possibility of negative weights, also similar terms were dropped while presenting the relevances.
- Replacing IDF function with a function having non-negative value
- the relevance of frequently occuring or common terms was normalized.

Inspite of the modifications, Okapi BM25 does not really provide better results, particularly for long documents. Consequently, certian modifications like BM25L [38] were also proposed to eradicate the issues in the BM25 method. [59, 78].

6.2.3 Feature based term weighting

One of the current state-of-art algorithms in the field of phrase extraction is YAKE! [13]. The model employs the concept of feature–based term weighting to measure the relevance of terms. The model also focuses on location of stop words in the document to extract the phrases. As an example, consider the phrase "Game of Thrones". The model considers the importance of the stop-words ("of") in the phrase, while evaluating the importance of the terms "Game", "Throne" and the phrase "Game of Thrones". The relevant paper provides a collection of documents and phrases for testing the validity of the model and is available at [https://github.com/LIAAD/yake].

At its core, the model is a combination of five features. The first feature in YAKE! is based on the case of a term (T_{CASE}). The second feature is based on the position ($T_{position}$) of a term. The third feature is based on normalized term frequency (tf_{norm}) of a term in a document. Interestingly, the model is single document based so any other parameter with term frequency is not added. The fourth feature (T_{rel}) is relatedness of term to the context of a document. The fifth and final feature ($T_{sentence}$) is based on the relevance of the sentences in which the term appears. Finally the term score w is calculated as:

$$w = \frac{T_{rel} + T_{position}}{T_{CASE} + \frac{tf_{norm}}{T_{rel}} + \frac{T_{sentence}}{T_{rel}}} \tag{18}$$

The researchers also proposed an n-gram keyword scoring algorithm. The model has been compared using the F-score value with other term weighting models and has been found to perform excellently. This is mainly attributed to the fact that the model considers the position of stopwords as a parameter in the term weighing scheme which significantly boosts its performance. The developers of the model have made the source code of the model freely available at [https://boudinfl.github.io/pke/build/html/index.html].

6.3 Coherent approach

Another famous approach based on the concept of coherent term weighting is presented in [33]. The concept addresses the challenge of doing term weighting in the context of a single document. The concept of entropy weighting (CEW) is simple to understand and easy to implement.

The entire model is made up of of three different parts a. log weighting (π_{log}) [80] b. BDC weighting [81] (π_{BDC}) and c. Entropy weighting (π_{EW}). While the CEW weight (π_{CEW} of word w is represented as [33] :

$$\pi_{CEW}(w) = \pi_{log}(w) \times \pi_{BDC}(w) \times \pi_{EW}(w) \tag{19}$$

The coherent approach is majorly focused on coherent nature of words rather than analysing the whole document as whole. This approach helps to understand or move forward with the concept of neighbour words as used by many NN based model such as word2vec. The approach could be consider better in prediction as the neighbour analysis of different words could provide a better result.

7 Vector based term weighting

There have been numerous other term weighting schemes suggested based on the concept of vectors. Many of these models have used the concept of n-grams to make the algorithms more robust in terms of semantic understanding of a document [12, 25]. Unfortunately, many of these models also tend to suffer from the *curse of dimensionality*, and their performances fell away as the size of the vocabulary increased. The size of vocabulary will increase the size of matrix and it leads to higher computation power and will gradually increase the time complexity. The reduce size of dimension will be light weighted and can be use easily without spending more computation power. Every time considering high dimension will not be beneficial as it may produce many ambiguous results.

To overcome these drawbacks various supervised or learning model were proposed.

7.1 Neural probabilistic model

The neural probabilistic model [8] is one of the first model to deal with semantic similarity in term weighting. While this model is considered as a base for supervised learning it is also inspired from the statistical learning model and probability. It is easy to understand that the conditional probability of every word depends on the words that precede it in a document, however the computation of this conditional probability is an arduous task for each term in a document.

The neural probabilistic model deals with the concept of n number of words in a vocabulary of size V in the context of a document. The model proposes two methods for training a supervised system and optimizes the training using parallel computing.

The basic idea of the model is to factorize the function f$(w_t,\ldots,w_{t-n+1}) = \hat{P}(w_t|w_1^{t-1})$ were w_t represents the t^{th} word from training corpus,$w_1^{t-1} = (w_1 \ldots w_1^{t-1})$ and P is its probability [8]:

- The major concept in this model is the mapping of distributed feature vectors connected with each word in a vocabulary represented as C. It is represented as the $|V| \times$ m matrix of free parameters
- Another important part is the mapping of the probability function C with a function "g" which maps an input sequence. The sequence is a feature vector of every word in the context represented as $(C(w_{t-n+1}, \ldots, C(w_{t-1})))$. It is the conditional probability of word w_t in V in reference to all words that follow it. The function is represented as [8]:

$$f(i, w_{t-1}, \ldots, w_{t-n+1}) = g(i, C(w_{t-1}. \ldots, C(w_{t-n+1}))) \tag{20}$$

The model is very simple to understand with only two hidden layers. The first is a hyperbolic tangent layer that shares the term features in layer C. This is followed by a softmax output layer. The model has been further modified by many other researchers [7, 26, 46, 47, 49].

7.2 Continuous space representations

Focusing more on semantic relation between words Mikolov et. al. designed a Recurrent Neural Network (RNN) [28] using the concept of continuous space based language model [42, 67, 68]. The model focuses on a relationship of words and models the relationship using an equation of the form $y = a - b + c$ where a:b and c:d are analogous questions. The results of y are represented in the continuous space very close to d as is expected. Representing words in the form of distributed vectors allows us to build various connections for a single word with many other words. The model also focuses on different singular/plural relation of similar words. For e.g., $apple - apples \approx car - cars$. The RNN model based architecture consists of an input layer w(t) where t is encoded using 1 of N coding. The next layer is a hidden layer s(t) with recurrent connections and last layer is an output layer y(t). The output layer is computed as [42].

$$s(t) = f(Uw(t) + Ws(t)) \tag{21}$$

$$y(t) = g(Vs(t)) \tag{22}$$

where, U is a column of word representations in this framework, $f(z) = \frac{1}{1+e^{-z}}$ and $g(z_m) = \frac{e^{zm}}{\sum_k e^{zk}}$. The model has been trained with 320M words [41] and correctly answers almost 40% of all questions asked.

7.3 Word2vec

The most famous word representation scheme proposed by Mikolov et al. is word2vec [43, 44]. The model is trained with billions of words and at the time it was proposed was considered revolutionary in concept. Other than training with huge corpus the model supports many languages other than English and specializes in generation of vectors capable of capturing semantic similarity of different words [24, 60]. The model proposed a dense vector space model to represent the tokens which was a departure from the sparse vector space models that were in vogue at that time. The use of the dense vectors provided a tremendous computational advantage to the word2vec [31, 35, 70, 83]. The implementation of the model is proposed using two different approaches. The first model uses skip-grams while the second model uses a technique called the continuous bag of words [43, 44]. To provide a generalise idea about the word2vec model, we plotted a graph of some countries in Asia represented in Fig. 1. The graph allow us to understand the vector relation between countries.

7.3.1 Skip-gram

In the skip-gram model, the input is given as a single word which is known as the target word *"t"* and the desired output is the set of context words *"o"*. The context words form a prefix and postfix window for the target word of window size m. Most researchers consider

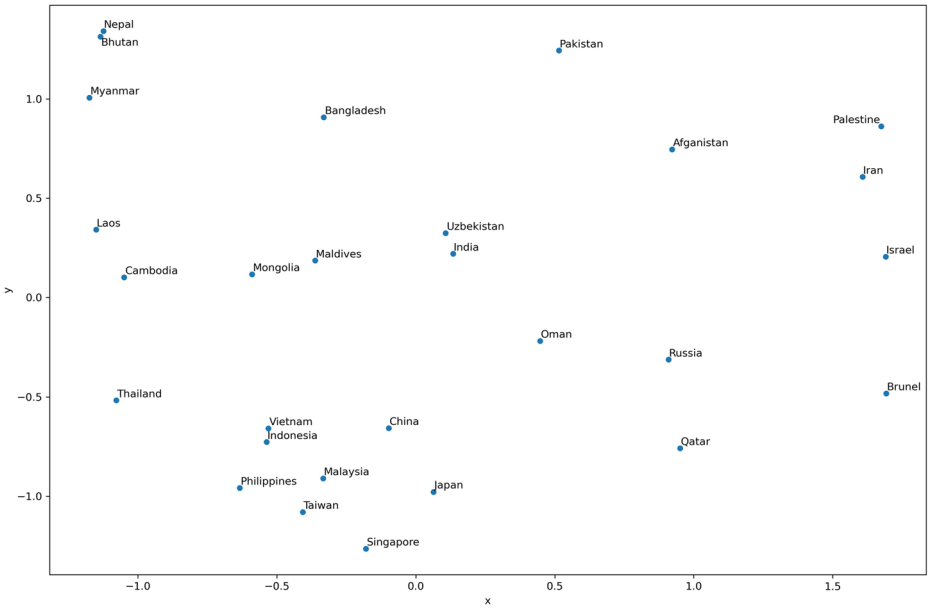

Fig. 1 The graph allow us to understand the vector relation between countries

five as an ideal window size for use in the skip gram technique. The model is designed to predict the set of most probable words with respect to the target word. To maximize the probability of each target word $t = 1, 2, 3 \ldots T$ in a window size m for each word the following function is used :

$$J'(\theta) = \prod_{it1}^{T} \prod_{-m \leq j \leq m; j \neq 0} p(w_{t+j}|w_t) \tag{23}$$

Unfortunately, (23) is very computationally intensive and thus Mikolov proposed the use of the negative log likelihood for finding the probability of context words. The negative log likelihood equation for vector "θ" is given as

$$J(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{-m \leq j \leq m; j \neq 0} log(p(w_{t+j}|w_t)) \tag{24}$$

To calculate the probability of a context word given a target word the concept of conditional probability has been used i.e. $p(w_{t+j}|w_t)$. In literature, the context word is also referred as the output word "u_o" and the center word is known as "v_c" respectively. The model uses the softmax function defined as :

$$p(o|c) = \frac{exp\left(u_o^T v_c\right)}{\sum_{w=1}^{v} exp\left(u_w^T v_c\right)} \tag{25}$$

to train the network.

As show in Fig. 2 the probability of the contexts word is calculated with respect to the center word using the softmax function. Subsequently, the negative log likelihood function is minimized to help detect the final set of context words. As can be seen skip-gram does not emphasize on the position of a word in a document but rather emphasises on the relationship between words which are frequently found in proximity of each other. This allows the model to weigh each token irrespective of its location in a particular document. While the model employs a simple back propagation technique to train itself one of the challenges is to choose an appropriate learning rate for the model. If the text is small, then the learning rate should be kept at a very small values so that the model does not overfit. As a final step the model use stochastic gradient decent (SGC) to minimize the function represented in (24). As the model is trained with 40 billion tokens it offers very good performance on test data even though the train set was quite noisy.

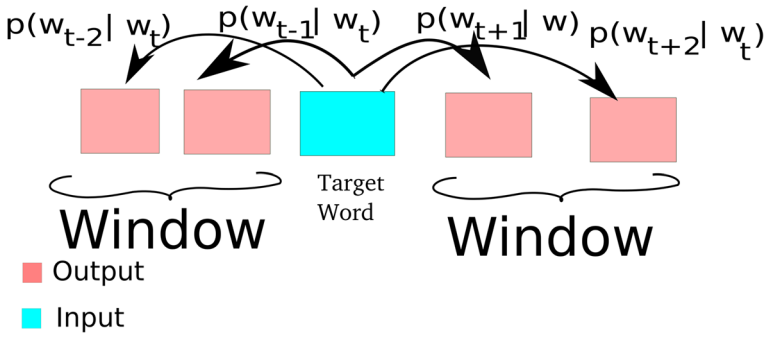

Fig. 2 Softmax probability calculation in skip-gram

7.3.2 Continuous bag of words

Another approach of implementing word2vec is the Continuous Bag of Words (CBOW) technique. In CBOW we use the context words to try to predict the target word. The approach is nearly the same as with the skip-gram approach and the probability calculations are shown in Fig. 3. Only the input and output matrix changes. In CBOW the input is size of context word with hot encoding vector of vocabulary size. Let the size of context be of length 5 and *"v"* be the size of vocabulary. Then the size of input is 5 $1 \times |v|$. While the size of input in skip-gram is $1 \times |v|$ as only one word is pased as input. Similarly, the output dimensions and the activation function is also changed. While CBOW uses the same (24) of negative log likelihood for minimizing the probability.

Both skip-grams and CBOW have different implementations and the desired outputs are also different. But still if we compare both the approaches skip-gram is more efficient on small corpus. While the training time of CBOW is very low for large data as compared to skip-gram. The word2vec model has been further enhanced for sentence vector representations, document vector representations and co-occurrence-based GLOVE model [50].

As discussed earlier the word2vec model has revolutionized the field of word embeddings and is still the most popular word embedding model as on date. The reason for this popularity is the model's ability to capture semantic similarity using a deep learning model which is already pre-trained and could directly be utilized. One of the criticisms of the word2vec model is the fact that while the model does make very good predictions it has been impossible to discern the inner working of the model. Let, us consider an example for the analogy if man:women :: king:?. The answer of such a query is queen, which is exactly what we have expect. However, the weights used in making these predictions have not been fully understood in the context of human interpretation.

Apart from word2vec, another model Glove [50] has been quite successful in capturing the semantic similarity of tokens. Glove considers the co-occurrence probability of each token with each other and uses the same matrix for context or word prediction. However, both Glove and word2vec consider a global context (due to training on vast amount of text and n number of language) for word embedding and it sometimes doesn't apply for random text. Another model known as ELMO [51] (Embeddings from Language Model) considers local context for word embedding. On most occasions, ELMO is used as an add on tool

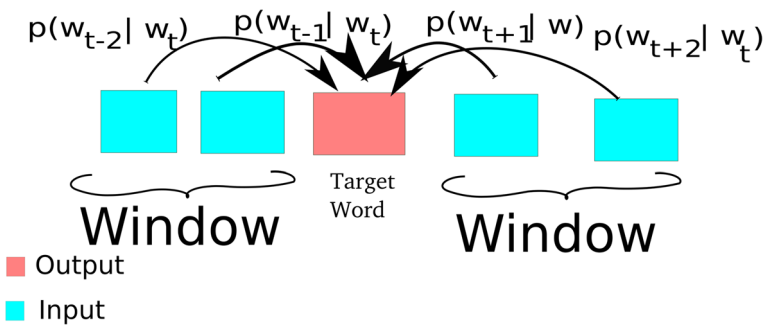

Fig. 3 CBOW probability calculation in skip-gram

for enhancement of the word2vec or Glove model. ELMO deploys a bidirectional LSTM for saving the contextual information [51]. This helps ELMO to solve the homonym word problem. Another model with similar logic is fastext [9]. The training time for both these models is low as compared to word2vec or Glove. fastext creates a binary tree of tokens which is then used to calculate the similarity between tokens using the concept of node path length.

8 Performance analysis

Term weighting finds numerous uses in the domain of NLP and different practitioners use it to accomplish different things. It could be used for keyword extraction, paraphrasing, eliminating irrelevant comments and lots more. Consequently no one measure can be used to truly measure the performance of a term weighting scheme. As every term weighting scheme has its own flavour, we tried to analyse the models using three major factors we found the most important in our research i.e.

– Ranking a term in a corpus or single document
– Term based classification.
– Measuring the semantic similarity between words relevant to the document.

8.1 Datasets

In our analysis we used seven different datasets to truly measure the capabilities of the various term weighting models. We also used datsets in different languages to check the performance of these models in non–English contexts. We present brief details about the datasets used in the following sections.

8.1.1 Cacic

Cacic [4] is a collection of 888 scientific papers published between 2005-2013. The dataset is collected from the Argentine Congress of Computer Science and is in Spanish language. The dataset also contains golden keywords of every document.

8.1.2 Krapvin2009

Krapvin2009 [29] is a English corpus of 2,304 complete research papers collected from ACM in the domain of computer science in the period 2003-2005. The dataset also contain golden standard keywords assigned by the authors. Interestingly many of the documents contain keywords which do not feature in the actual text at all.

8.1.3 Schutz2008

Schutz2008 [66] is also an English language dataset of 1,231 full-length research papers collected from PubMed central from 254 different journals. The collection contains diverse set of data ranging from Abdominal imaging to World journal of Urology. The dataset also contains golden standard keywords assigned by the authors for every document.

8.1.4 Wicc

Wicc [4] is similar to cacic as it is also a Spanish dataset of scientific research papers collected from Argentine Congress of Computer Science. The dataset contains 1640 documents and golden keywords collected from a period of 1999-2012.

8.1.5 WikiNews

WikiNews [10] is a collection of 100 French news articles. The data is collected from french version news published between May to December in the year of 2012. The data also contains golden keywords annotated by the curators.

8.1.6 NewsGroup

One of the most popular datasets for classification is newsgroup. The data has 20 class label with different kind of news. Every class folder is a set of news documents. The corpus in total contains nearly 18000 newsgroup documents.

8.1.7 BBC news

Similar to newsgroup it is also news dataset and popular for classification. The dataset has 5 class label with different news section. The dataset contains nearly 2200 documents. This data does not provide much leverage for training due to the size of dataset.

8.2 Popular measure for analysis

Various measures have been proposed for the performance analysis of term weighting schemes. Most of these models are based on the tenets of supervised learning and usually require the presence of golden keywords in the documents to assist the measurement of the performances. Some of the common performance measures used in our analysis are listed below.

8.2.1 F-score

The F-score is widely used for measuring the performance of a term ranking schemes. If the dataset is labeled it is often prudent to use the F-score to judge the performance of the term weighting scheme because it is easy to interpret. Let TN, FN, TP, FP denote the true negative, false negative, true positive and False positive counts respectively. The precision, recall and f-score (which is the harmonic mean of precision and recall) are defined as ratios as in (26).

$$\begin{aligned} Precision = \frac{TP}{TP + FP}, Recall = \frac{TP}{TP + FN} \\ F1 - score = \frac{2(Precision * Recall)}{Precision + Recall} \end{aligned} \tag{26}$$

As is obvious the F-score ranges from 0 to 1, where 1 is the best and 0 the worst.

8.2.2 Accuracy

Probably the simplest technique to analyse classification methods is to measure the accuracy of the model. Every model tries to predict the desired class "$\hat{x_j}$" of a document j. If it predicts the class correctly then it is awarded a score 1.0 and if not then it is awarded a score of 0. Summation of predictions is normalized by the size of sample "n_{sample}". If *"x"* is considered as actual label and f(x) as performance indicator then accuracy is measured as:

$$accuracy(x, \hat{x}) = \frac{1}{n_{samples}} \sum_{j=0}^{n_{samples}-1} f(\hat{x_j} == x_j) \tag{27}$$

8.3 Quantitative analysis

In this section we present the results in a tabulated format of the algorithms discussed in this paper.

To rank the different models we used only statistical algorithms, as supervised and some other neural network model treat word embeddings as a classification task. Although statistical algorithms are more focused on ranking rather than measuring the semantic similarity of words. The analysis provides a good insight as to their effectiveness as can be seen in Table 1

Some term weighting models are essentially classification algorithms for all practical purposes. To measure the performance of theses we use the metric of accuracy and the results are shown in Table 2. The BBC news dataset shows the highest accuracy because documents in the dataset are more separated in term of context. Consequently, almost all the models found it easy to work with this dataset. The reason to include such dataset in our analysis is to show that measuring accuracy on such a dataset will not give any conclusive understanding of the actual effectiveness of the models.

The last measure used in our analysis was the detection of semantic similarity of documents. In this method we calculated the semantic similarity of document with its own corpus and with another corpus. In this analysis we used four class documents from the newsgroup corpus. We calculated the term weight of every word in a document and generated a corresponding vector. To ensure that all vectrors had equal lengths we padded the vectors to the the size of the vocabulary. The results are shown in Table 3

Table 1 Analysis of one-gram words with respect to ranking in a corpus with different documents

Algorithms	Cacic	Krapvin2008	Schutz2008	Wicc	WikiNews
tf	**0.25**	**0.3**	**0.26**	**0.33**	**0.43**
tf-idf	0.22	0.25	0.21	0.14	0.12
mtf-idf	0.08	0.02	0.04	0.05	0.12
tf-midf	0.13	0.31	0.07	0.15	0.12
mtf-midf	0.07	0.021	0.04	0.05	0.12
YAKE	0.23	0.23	0.2	0.32	0.43

Table 2 Measuring accuracy of different models for classification

Algorithms	NewsGroup	BBC news
tf-idf	**0.85**	**0.975**
tf-pf	0.82	0.965
tf-rf	0.815	0.965
tf-icf	0.827	0.974
tf-chi2	0.814	0.966
tf-binicf	0.79	0.964
tf-rrf	0.816	0.971
CEW	0.59 ±0.01	—

9 Discussion

We have tried to analyse the different term weighting schemes on different parameters. Some algorithms show acceptable performance across all the datasets e.g., tf-idf. Tf-idf is the most popular and versatile algorithm we have studied during our research. The algorithms that extend the tf-idf inherit this consistency. As noticed from rank–based term weighting results on normal size documents, i.e, documents of 7 to 10 pages. Tf works very well after pre-processing of the document. In pre-processing we removed the stop-words and punctuation from the document and split the words into one-gram for term weighting. The reason tf outperforms every algorithm is because most of the algorithms are corpus oriented. On the other hand, idf based algorithms assume that a corpus shall be made up of similar or at least thematically similar kind of documents. It is not possible for every corpus to have similar documents so tf and YAKE outperform every other algorithm as they focus on a single document. It's worth noting that the term frequency which is considered a very primitive feature still provides very effective results. Moreover it remains extremely easy to compute. Yake! is design primarily for three-gram phrase extraction but it works very well on one-grams also. Other than tf and YAKE, tf-idf and tfmidf also provide some satisfying results as it is just a multiplicative extension of tf. In conclusion, it can be said that any algorithm employinf the tf mechanism is bound to produce good term weighting results as far as single documents are concerned.

We also analysed various supervised algorithms and found all of them offered similar performance results. As most of the algorithms are variants of tf-idf, and the schemes have modeled the task of term weighting as a classification task only, this is something which we can expect. We used newsgroups corpus with 20 labels for testing term weighting using a supervised model. In our pre- processing phase we removed only the punctuation from the document and the header of the document. The results were encouraging and still have a scope for improvement on such corpus.

Lastly, we analysed semantic similarity between documents also known as information retrieval analysis represented in Table 3. For this analysis we used four classes the from newsgroup corpus. The classes are related in some manner but were also very distinct from one another. We chose rec.sport.baseball and rec.sport.hockey from sports and talk.politics.mideast and talk.politics.guns from politics. The main target of this analysis was to show the relationship between similar or distinct classes based on the vector representation of a document. We presented the average cosine similarity between two class. During this analysis we used both the approaches of word2vec but found skip-gram to outperform every other algorithm. It effectively represented the similarity and distinct nature

Table 3 Average cosine similarity between different labels in newsgroup corpus is presented

Algorithms	Datasets	rec.sport. baseball	talk. politics. mideast	talk.politics. guns	rec.sport. hockey
tf-idf	rec.sport.baseball	0.0194	0.016	0.014	0.016
	talk.politics.mideast	0.016	0.0272	0.018	0.0175
	talk.politics.guns	0.014	0.018	0.022	0.163
	rec.sport.hockey	0.016	0.017	0.016	0.023
mtf-idf	rec.sport.baseball	0.11	0.082	0.09	0.101
	talk.politics.mideast	0.08	0.09	0.08	0.07
	talk.politics.guns	0.09	0.08	0.1	0.08
	rec.sport.hockey	0.101	0.07	0.08	0.109
tf-midf	rec.sport.baseball	0.31	0.34	0.34	0.306
	talk.politics.mideast	0.34	0.42	0.4	0.346
	talk.politics.guns	0.34	0.4	0.407	0.342
	rec.sport.hockey	0.306	0.346	0.342	0.316
mtf-midf	rec.sport.baseball	0.44	0.467	0.483	0.449
	talk.politics.mideast	0.467	0.536	0.533	0.474
	talk.politics.guns	0.483	0.533	0.533	0.489
	rec.sport.hockey	0.449	0.474	0.489	0.464
BM25 score	rec.sport.baseball	9.47	9.125	7.098	7.856
	talk.politics.mideast	6.982	27.97	12.62	7.002
	talk.politics.guns	6.55	15.105	15.38	6.499
	rec.sport.hockey	6.876	8.9	6.832	11.97
word2vec skip-gram	rec.sport.baseball	0.209	0.074	-0.004	0.139
	talk.politics.mideast	0.074	0.207	0.168	0.142
	talk.politics.guns	-0.004	0.168	0.222	0.103
	rec.sport.hockey	0.139	0.142	0.103	0.197
word2vec CBOW	rec.sport.baseball	0.369	0.33	0.299	0.253
	talk.politics.mideast	0.33	0.365	0.33	0.266
	talk.politics.guns	0.299	0.332	0.407	0.328
	rec.sport.hockey	0.253	0.266	0.328	0.335

For BM25 we measure the score instead of cosine similarity

of different classes. It also finds the hidden relationship between hockey and mideast country. Mostly, middle eastern countries play hockey more than basketball. The similarity score with basketball is negative to show the distinct nature of the classes. However, as group of sports and politics have very higher score as compared to other classes.

We also analysed BM25 algorithm on BM25 score for every documents. As an input to the model we pre-processed all the documents and removed all the punctuation and stopwords from the document. However, even though the algorithm was quite fast, but it did not provide as good a result as a skip-gram implementation of word2vec.

10 Recent advancements

Continuing the journey from word embedding and moving into the domain of language understanding BERT (Bidirectional Encoder Representations from Transformers) [21] is now considered as the state-of-the-art. BERT uses a bidirectional approach to generate language models. It uses only an encoder mechanism to create the model. The model's success is attributed primarily due to the use of a probing classifier for internal vector representation. As the model is based on the attention mechanism [74] the internal attention weights also help the model to be very powerful. The model is trained on 800M Books corpus and 2500M token English Wikipedia. Because of its use of the attention model BERT has been found to be extremely effective in a lot of different contexts [74].

BERT as a component has also been used for contextual based term weighting. The DeepCT (Deep contextual term weighting) [17] framework use BERT for primary extraction of contextual feature of a term. Using transformers it gradually learns the context of the text and DeepCT helps to score the term. DeepCT is also represented using tf as td_{DeepCT} [17], to provide more meaningful extraction of frequent words in a document.

Efforts have also been made to club a modern approach like BERT with tf-idf. The development of a supervised model based on both techniques has the ability to provide very effective term weighting in both the local and global context. Considering relevance term weighting for class determination explained in Section 6.1.3. Tf-idfc-rf (term frequency inverse document frequency class relevance factor) [14] has been used to solve problems in sentiment analysis by modeling the task as a classification problem.

11 Conclusion

It's not possible to cover all the weighting techniques proposed in history, but int this paper we have tried to provide a detailed review of all the major methods for term weighting. Statistical term weighting is not as efficient as supervised term weighting in some sense, but the processing speed of statistical term weighting techniques are much faster as compared to supervised methods. The supervised term weighting is efficient if they are applied to a specific domain. Choosing the correct word embedding technique is very important in the field of natural language processing as it paves the way for an efficient semantic interpretation of a document. In our analysis we found that both statistical and supervised methods can be effective depending on the nature of the problem.

Funding No funding was received by any of the authors to support the research that resulted in this manuscript.

Declarations

Conflict of Interest The authors have no conflicts of interest to declare that are relevant to the content of this article.

References

1. Alaya M, Bussy SG (2017) Binarsity: a penalization for one-hot encoded features in linear supervised learning 39:4760–4768. arXiv:1703.08619

2. Aljaber B, Stokes N, Bailey J, Pei J (2010) Document clustering of scientific texts using citation contexts. Inf Retr 13:101–131
3. Alt (2010) Analytical evaluation of term weighting schemes for text categorization. Pattern Recogn Lett 31:1310–1323
4. Aquino G (2015) Keyword identification in spanish documents using neural networks. J Comput Sci Technol 15:465–473
5. Azam N, Yao J (2012) Comparison of term frequency and document frequency based feature selection metrics in text categorization. Expert Syst Appl 39:4760–4768
6. Bafna P, Pramod D, Vaidya A (2016) Document clustering: TF-IDF approach. Encycloped Stat Sci 33:61–66
7. Bengio Y (2008) Sen, Adaptive importance sampling to accelerate training of a neural probabilistic language model. IEEE Trans Neural Netw 19:713–722
8. Bengio Y, Ducharme R (2003) A neural probabilistic language model. J Mach Learn Res 3:1137–1155
9. Bojanowski P, Grave E, Joulin A, Mikolov T (2017) Enriching word vectors with subword information. Trans Assoc Comput Linguist 5:135–146
10. Bougouin A, Boudin F, Dailles B (2013) Topicrank: Graph-based topic ranking for keyphrase extraction. Master's thesis, vol 24. National University of Ireland, pp 1532–1543
11. Brinker K, Moerchen F, Glomann B, Neubauer C (2010) Online document clustering using TFIDF and predefined time windows 39:4760–4768. arXiv:1703.08619
12. Brown P, Della Pietra V, Desouza P, Lai J, Mercer R (1992) Class-based n-gram models of natural language. Comput Linguist 18:467–480
13. Campos R, Mangaravite V (2020) YAKE! Keyword extraction from single documents using multiple local features. Inf Sci 509:257–289
14. Carvalho F, Guedes G (2020) TF-IDFC-RF: a novel supervised term weighting scheme. arXiv:2003.07193
15. Chen K, Zhang Z, Long J, Zhang H (2016) Turning from TF-IDF to TF-IGM for term weighting in text classification. Expert Syst Appl 66:245–260
16. Chirawichitchai N, Sa-nguansat P, Meesad P (2010) Developing an effective Thai document categorization framework base on term relevance frequency weighting. Inf Sci 509:19–23
17. Dai Z, Callan J. (2020) Context-aware term weighting for first stage passage retrieval. In: Proceedings of the 43rd international ACM SIGIR Conference On Research And Development In Information Retrieval, pp 1533–1536
18. Dai A, Olah C, Le Q (2015) Document embedding with paragraph vectors 58:239–243arXiv:1507.07998
19. Debole F, Sebastiani F (2004) Supervised term weighting for automated text categorization. J Amer Soc Inf Sci 27:81–97
20. Deng Z, Tang S, Yang D, Li M, Xie K (2004) A comparative study on feature weight in text categorization. Adv Neural Inf Process Syst 21:588–597
21. Devlin J, Chang M, Lee K, Toutanova K (2018) Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv:1810.04805
22. Ferguson P, O'Hare N, Lanagan J, Phelan O, McCarthy K (2012) An investigation of term weighting approaches for microblog retrieval. J Amer Soc Inf Sci 27:552–555
23. Gao Y, Xu Y, Li Y, Liu B (2013) A two-stage approach for generating topic models. Expert Syst Appl 38:221–232
24. Goldberg Y, Levy O (2014) word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method 31:721–735. arXiv:1402.3722
25. Huang C, Tian Y, Zhou Z, Ling C, Huang T (2006) Keyphrase extraction using semantic networks structure analysis. Adv Neural Inf Process Syst 21:275–284
26. Huang W, Wu Z, Liang C, Mitra P, Giles C (2015) A neural probabilistic model for context based citation recommendation. IEEE Trans Pattern Anal Mach Intell 31:721–735
27. Jimenez S, Cucerzan S, Gonzalez F, Gelbukh Aer (2018) BM25-CTF: Improving TF and IDF factors in BM25 by using collection term frequencies. J Intell Fuzzy Syst 34:2887–2899
28. Kombrink S, Mikolov T (2011) Recurrent neural network based language modeling in meeting recognition. Comput Speech Lang 21:269–274
29. Krapivin M, Autaeu A. rM. (2009) Large dataset for keyphrases extraction. Inf Process Manag 24:1532–1543
30. Lan M, Tan C, Su J, Lu Y (2008) Supervised and traditional term weighting methods for automatic text categorization. IEEE Trans Pattern Anal Mach Intell 31:721–735
31. Lau J, Baldwin T (2016) An empirical evaluation of doc2vec with practical insights into document embedding generation 20:723–730. arXiv:1607.05368

32. Le Q, Mikolov T (2014) Distributed representations of sentences and documents. Expert Syst Appl 36:1188–1196
33. Li X, Zhang A, Li C, Ouyang J, Cai Y (2018) Exploring coherent topics by topic modeling with term weighting. Inf Process Manag 54:1345–1358
34. Li J, Zhang K et al (2007) Keyword extraction based on tf/idf for Chinese news document. Wuhan Univ J Natural Sci 12:917–921
35. Lilleberg J, Zhu Y, Zhang Y (2015) Support vector machines and word2vec for text classification with semantic features 39:136–140. arXiv:1703.08619
36. Litvak M, Last M (2008) Graph-based keyword extraction for single-document summarization. ACM SIGKDD Explor Newslett 7:17–24
37. Liu Y, Loh H, Sun A (2009) Imbalanced text classification: a term weighting approach. Expert Syst Appl 36:690–701
38. Lv Y, Zhai C (2011) When documents are very long. Int J Artif Intell Tools 13:1103–1104
39. Manning C (2008) Sch, Introduction to information retrieval. University Press, Cambridge, pp 239–243
40. Matsuo Y, Ishizuka M (2004) Keyword extraction from a single document using word co-occurrence statistical information. Int J Artif Intell Tools 13:157–169
41. Mikolov T (2011) Strategies for training large scale neural network language models. J ACM (JACM) 9:196–201
42. Mikolov T (2013) Linguistic regularities in continuous space word representations 20:746–751. arXiv:1607.05368
43. Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space 34:2887–2899. arXiv:1301.3781
44. Mikolov T, Sutskever I, Chen K, Corrado G, Dean J (2013) Distributed representations of words and phrases and their compositionality. Adv Neural Inf Process Syst 26:3111–3119
45. Mladeni'c D, Grobelnik M (1998) Feature selection for classification based on text hierarchy. Comput Speech Lang 21:492–518
46. Mnih A, Hinton G (2008) A scalable hierarchical distributed language model. Adv Neural Inf Processi Syst 21:1081–1088
47. Mnih A, Teh Y (2012) A fast and simple algorithm for training neural probabilistic language models 12:917–921. arXiv:1206.6426
48. Mooney R, Bunescu R (2005) Mining knowledge from text using information extraction. ACM SIGKDD Explor Newslett 7:3–10
49. Morin F, Bengio Y (2005) Hierarchical probabilistic neural network language model. Int J Artif Intell Tools 5:246–252
50. Pennington J, Socher R, Manning C (2014) Glove: Global vectors for word representation. Proceedings Of The 2014 Conference On Empirical Methods In Natural Language Processing (EMNLP), pp 1532–1543
51. Peters M, Neumann M, Iyyer M, Gardner M, Clark C, Lee K, Zettlemoyer L (2018) Deep contextualized word representations. arXiv:1802.005365
52. Piantadosi S (2014) Zipf's word frequency law in natural language: A critical review and future directions. Psychonom Bullet Rev 21:1112–1130
53. Polettini N (2004) The vector space model in information retrieval-term weighting problem. Entropy 34:1–9
54. Quinlan J (1986) Induction of decision trees. Mach Learn 1:81–106
55. Ramos J et al (2003) Using tf-idf to determine word relevance in document queries. J ACM (JACM) 242:133–142
56. Robertson S, Jones K (1976) Relevance weighting of search terms. J Amer Soc Inf Sci 27:129–146
57. Robertson S, Walker S (1994) Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. J Amer Soc Inf Sci 27:232–241
58. Robertson S, Walker S, Jones S, Hancock-Beaulieu M, Gatford M et al (1995) Okapi at TREC-3. Nist Special Publ Sp 109:109
59. Robertson S, Zaragoza H (2009) The probabilistic relevance framework: BM25 and beyond. Now Publishers, Inc., pp 98–103
60. Rong X (2014) word2vec parameter learning explained 31:1103–1104. arXiv:1411.2738
61. Sabbah T, Selamat A, Selamat M, Al-Anzi F, Viedma E, Krejcar O, Fujita H (2017) Modified frequency-based term weighting schemes for text classification. Appl Soft Comput 58:193–206
62. Salton G, Buckley C (1988) Term-weighting approaches in automatic text retrieval. Inf Process Manag 24:513–523
63. Salton G, Lesk M (1968) Computer evaluation of indexing and text processing. J ACM (JACM) 15:8–36

64. Salton G, Wong A, Yang C (1975) A vector space model for automatic indexing. Commun ACM 18:613–620
65. Salton G, Wu H (1980) A term weighting model based on utility theory 31:9–22. arXiv:1411.2738
66. Schutz A, Thorsten er (2008) Keyphrase extraction from single documents in the open domain exploiting linguistic and statistical methods. Master's thesis, vol 24. National University of Ireland, pp 1532–1543
67. Schwenk H (2006) Continuous space language models for statistical machine translation. J Korea Soc Comput Inf 20:723–730
68. Schwenk H (2007) Continuous space language models. Comput Speech Lang 21:492–518
69. Sebastiani F (2002) Machine learning in automated text categorization. ACM Comput Surv (CSUR) 34:1–47
70. Sien (2015) Adapting word2vec to named entity recognition. Appl Soft Comput 58:239–243
71. Soucy P, Mineau G (2005) Beyond TFIDF weighting for text categorization in the vector space model 5:1130–1135. arXiv:1607.05368
72. Tsai R, Hung H, Dai H, Lin Y, Hsu W (2008) Exploiting likely-positive and unlabeled data to improve the identification of protein-protein interaction articles. J ACM (JACM) 9:1–10
73. Tsai F, Kwee A (2011) Experiments in term weighting for novelty mining. Expert Syst Appl 38:14094–14101
74. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez A, Kaiser Ł, Polosukhin I (2017) Attention is all you need. Adv Neural Inf Process Syst:5998–6008
75. Wang J, Liu J, Wang C (2007) Keyword extraction based on pagerank. J Korea Soc Comput Inf 20:857–864
76. Wang Z, Ma L, Zhang Y (2016) A hybrid document feature extraction method using latent Dirichlet allocation and word2vec. Int J Artif Intell Tools 13:98–103
77. Wang X, McCallum A, Wei X (2007) Topical n-grams: phrase and topic discovery. Entropy 34:697–702
78. Whissell J, Clarke C (2011) Improving document clustering using Okapi BM25 feature weighting. Inf Retr 14:466–487
79. Wilson E (1927) Probable inference. J Amer Stat Assoc 22:209–212
80. Wilson A, Chew P (2010) Term weighting schemes for latent dirichlet allocation 12:465–473. arXiv:1206.6426
81. Yang K, Cai Y, Chen Z, Leung H, Lau R (2016) Exploring topic discriminating power of words in latent dirichlet allocation. Expert Syst Appl 42:2238–2247
82. You E, Choi G, Kim S (2015) Study on extraction of keywords using TF-IDF and text structure of novels. J Korea Soc Comput Inf 20:121–129
83. Zhang D, Xu H, Su Z, Xu Y (2015) Chinese comments sentiment classification based on word2vec and SVMperf. Expert Syst Appl 42:1857–1863
84. Zipf G (2016) Human behavior and the principle of least effort: An introduction to human ecology. Ravenio Books