

ADEP WORKING PAPER SERIES

**Ranked short text classification using co-occurrence features
and score functions**

Brian Dumbacher
U.S. Census Bureau

Daniel Whitehead
U.S. Census Bureau

Working Paper 2024-06
June 2024

Associate Directorate for Economic Programs
U.S. Census Bureau
Washington DC 20233

Disclaimer: Any views expressed are those of the author(s) and not necessarily those of the U.S. Census Bureau. Results were approved for release by the Census Bureau's Disclosure Review Board, authorization number CBDRB-FY22-ESMD001-008.

Ranked short text classification using co-occurrence features and score functions

Brian Dumbacher, U.S. Census Bureau, brian.dumbacher@census.gov

Daniel Whitehead, U.S. Census Bureau, daniel.whitehead@census.gov

ADEP Working Paper 2024-06

June 2024

Abstract

This article explores the use of co-occurrence features and score functions to perform ranked classification of short text. Unlike features based on word sequences, co-occurrence features are based on word combinations with no restrictions on word order or distance. Co-occurrence features are appropriate for short text because documents in this setting contain very few words. We consider a variation of the Vector Space Model called the “umbrella” vectorization that emphasizes textual details and reduces feature redundancy. We also propose a complementary score function based on a weighted average of the features’ class distributions in the corpus. For validation, the methods are applied to four short text datasets and compared to baseline classifiers. The proposed score function performs better than a modified BM25 classifier and achieves a level of accuracy similar to that of logistic regression.

Keywords:

Co-occurrence features; Information retrieval; Ranked classification; Short text; Umbrella vectorization; Vector Space Model

JEL Classification Codes:

C. Mathematical and Quantitative Methods

- C1. Econometric and Statistical Methods and Methodology: General
 - C15 Statistical Simulation Methods: General

Ranked short text classification using co-occurrence features and score functions

Brian Dumbacher^{1,*} and Daniel Whitehead¹

¹U.S. Census Bureau, 4600 Silver Hill Road, Washington, DC 20233, USA

*Corresponding author. Email: brian.dumbacher@census.gov

Abstract

This article explores the use of co-occurrence features and score functions to perform ranked classification of short text. Unlike features based on word sequences, co-occurrence features are based on word combinations with no restrictions on word order or distance. Co-occurrence features are appropriate for short text because documents in this setting contain very few words. We consider a variation of the Vector Space Model called the “umbrella” vectorization that emphasizes textual details and reduces feature redundancy. We also propose a complementary score function based on a weighted average of the features’ class distributions in the corpus. For validation, the methods are applied to four short text datasets and compared to baseline classifiers. The proposed score function performs better than a modified BM25 classifier and achieves a level of accuracy similar to that of logistic regression.

Keywords: Co-occurrence features, Information retrieval, Ranked classification, Short text, Umbrella vectorization, Vector Space Model

1. Introduction

Text classification is the problem of classifying documents according to a pre-defined set of categories. It is an important task with many applications such as online content tagging, search engine optimization (Lei et al., 2020), digital marketing (Salminen et al., 2019), news article categorization (Elnagar et al., 2020), sentiment analysis (Ghiassi et al., 2013; Melville et al., 2009), spam filtering (Nagwani and Sharaff, 2017), survey response classification (Tarnow-Mordi, 2017; Giorgetti and Sebastiani, 2003), occupation coding (Schierholz and Schonlau, 2021), and electronic health record classification (Mascio et al., 2020). With the explosion of digital information, the needs and opportunities for text classification have increased significantly. Text classification is interdisciplinary and lies at the intersection of machine learning (Aggarwal, 2018), natural language processing (Jurafsky and Martin, 2009), and information retrieval (Goswami, 2014; Cunningham et al., 1997). Consequently, there are various ways in which to frame the problem and build a model.

A common approach to text classification uses the Vector Space Model (VSM) representation of text (Turney and Pantel, 2010; Salton et al., 1975). Under the VSM, a document is represented as a high-dimensional vector of weights corresponding to terms in the corpus. For example, the

terms could be individual words, which are also known as single features, s-features (Figueiredo et al., 2011), and unigrams. This is the so-called “bag-of-words” (BOW) representation. More generally, the terms could be longer word sequences of length n, which are commonly referred to as n-grams. The weights in the VSM reflect term importance and serve as the feature values. Possible weighting methods include binary (simply indicating the presence of terms), term frequency, and term frequency-inverse document frequency (TF-IDF). The popular TF-IDF method gives more weight to terms that occur frequently in the given document and infrequently in the corpus (defined by the number of documents containing the term). Figure 1 shows an example VSM representation.

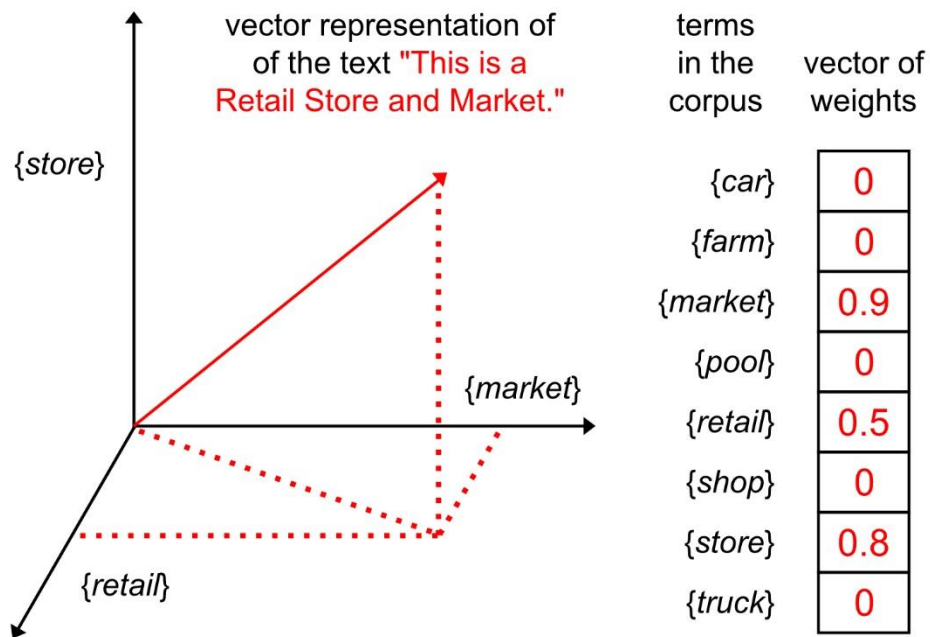


Figure 1. Example Vector Space Model (VSM) representation. The example text is “This is a Retail Store and Market.” The features are individual words and are indicated by curly brackets. Common stop words in the text such as “is” and “a” are ignored.

In the case of multi-class classification (three or more classes), there is an important distinction between “hard” and “ranked” classification (Sebastiani, 2002). In “hard” classification, the model predicts a single class for a given document. However, for information retrieval applications and similar problems, it may be desirable for the model to return multiple classes. In “ranked” classification, the model ranks the classes according to some measure of confidence and then returns the highest-scoring classes. This measure could be a directly estimated probability or a relevance score calculated using a nonparametric score function (Goswami, 2014; Robertson and Zaragoza, 2009).

Another distinction in text classification involves document length. Text classification is broad in definition and could refer to classifying documents ranging from short phrases to entire

books (Worsham, 2018). There are no agreed-upon criteria for what makes a document “short” or “long”, but “short” in this article refers to a single sentence. Examples of short text include search queries, micro-blogs, text messages, news article headlines, and open-ended survey responses. The typical document length in the corpus has implications for the VSM weights. For example, because there tend to be fewer repeated words in short text, the term frequencies behave like binary indicators. In turn, the TF-IDF weights behave like functions of the inverse document frequencies.

Text classification based on short text is more challenging because there are fewer words and phrases on which to base a prediction (Wang et al., 2017). To improve classification in this setting, additional VSM features have been proposed to capture greater semantic and contextual detail (Man, 2014; Pôssas et al., 2002). For example, some VSMs use word co-occurrence features based on word combinations. Unlike word sequences, word combinations place no restrictions on the order of words or the distances between them. Words are considered co-occurring even if they are the first and last words of the text. Co-occurrence features are also referred to as compound features, c-features (Figueiredo et al., 2011), termsets (Badawi, 2015), and itemsets. For consistency with the n-gram terminology, we introduce “n-combs” as another synonym for word combinations. N-grams and n-combs are examples of so-called composite features because they are composed of multiple words.

This article focuses on the problem of ranked classification of short text. We consider a vectorization that uses co-occurrence features to emphasize textual details and reduce feature redundancy. A complementary score function is proposed that computes a weighted average of the features’ class distributions in the corpus. The rest of the article is organized as follows. Section 2 reviews related work on short text classification, co-occurrence features, and score functions. Section 3 introduces notation for the problem and describes the methodology. Next, four datasets suitable for ranked short text classification are described in Section 4. Section 5 outlines a comparative study of classifiers using these datasets, and Section 6 presents results. Lastly, Section 7 summarizes findings and describes ideas for future work.

2. Related work

As stated in Wang et al. (2017), “short text is considerably different from traditional long text documents due to its shortness and conciseness, which somehow hinders the applications of conventional machine learning and data mining algorithms in short text classification.” To provide the BOW-based VSM more semantic context, composite features such as n-grams and n-combs can be added. N-grams may seem more attractive because they are based on words appearing in the same part of the document (Badawi, 2015). However, documents such as single sentences are too short to have multiple parts. Even the first and last words of short text have some relation. Therefore, in this setting n-combs may pick up on useful associations that n-grams do not. There are typically a large number of possible n-combs in the corpus that can

be used as features. Adding too many n-combs leads to model overfitting (Badawi, 2015), so it is best to augment the VSM with a subset.

There are various examples in the literature of using n-combs and feature selection methods along these lines. In the context of short text classification with support vector machines, Man (2014) augments the conventional VSM with 2-combs that occur frequently in the corpus. The author finds that these additional features improve model performance, especially when the number of training documents is small. Similarly, Soumya and Shibily (2014) use the chi-squared statistic to determine the most discriminating words and then augment the VSM with associated, frequently occurring n-combs. The authors report a slight improvement in their naïve Bayes model and note that co-occurrence features may also benefit information retrieval.

Figueiredo et al. (2011) propose a feature extraction method that applies a threshold based on the concept of dominance (discriminative ability) to determine what 2-combs to add. The authors conduct a comparative study with three feature types: words, words and 2-combs (proposed method), and words and 2-grams. They also consider three machine learning algorithms: k-nearest neighbors, naïve Bayes, and support vector machines. An increase in F1-type measures (Tan et al., 2019), which balance recall and precision, is observed when the proposed 2-comb extraction method is used.

Wan et al. (2019) propose a method for selecting composite features called Syntax Augmented Bigram. Their method applies to 2-grams and 2-combs and is based on a metric called relevance category frequency. This metric accounts for both the discriminative ability and redundancy of composite features. Redundancy refers to the undesirable correlation between the composite feature and its sub-features. The authors provide **{machine, learning}** as an example of an n-comb that does not introduce redundancy; it provides useful discriminative information not already provided by the words **{machine}** and **{learning}**. The authors apply their method to three datasets using naïve Bayes and support vector machines.

Complementary to these “hard” text classification studies is research on what weights to assign the features. Pekar et al. (2004) conduct a comparative study of weighting methods for co-occurrence features. The authors distinguish between methods identifying features that discriminate classes (odds ratio, gain ratio, and mutual information) and methods favoring features that characterize classes (term strength). They conclude it is possible to obtain consistent improvement over unweighted features if the method and corresponding parameters are chosen carefully. On a related note, Carvalho and Guedes (2020) and Erenel et al. (2011) study the performance of various supervised and unsupervised weighting methods for text classification. Unsupervised weighting methods such as TF-IDF do not consider class information, whereas supervised weighting methods do. It is observed that the supervised weighting methods perform better.

Weighting methods such as TF-IDF can also be used by ranked text classifiers and retrieval models. These models are based on a score function, which calculates a relevance score for each class. This score serves as a measure of confidence that the corresponding class is true. The classes are then ranked according to score. Score functions can be theoretically motivated or more nonparametric in nature. Many effective score functions can be expressed as a sum of contributions from the various features associated with the given document (Aggarwal, 2018; Goswami, 2014). Two examples are the score functions used by the binary independence model (Aggarwal, 2018) and the popular BM25, or Okapi, model (Robertson and Zaragoza, 2009). The sum of contributions can incorporate weights to account for TF-IDF-related information and give the features appropriate influence in calculating relevance scores. In the context of information retrieval, Pôssas et al. (2002) explore set-based models based on co-occurrence features called closed and maximal termsets. Association rules are used to calculate weights for these features. The authors apply their proposed method to three datasets and demonstrate improvement in retrieval performance over the BOW-based VSM.

3. Methodology

This section introduces notation for the ranked short text classification problem and describes the “umbrella” vectorization and weighted dominance score function.

3.1 Setup and features

Consider a classification scheme with $D \geq 3$ classes. Denote the raw training data consisting of N documents by $\{(r_i, y_i)\}_{i=1}^N$, where r_i and y_i are the raw text and class for document i , respectively. Furthermore, denote by \mathcal{S} the set of all possible strings that can be constructed using keyboard characters. There is a practical limit on the length of strings in \mathcal{S} so that they can be considered short text. The r_i are cleaned by an algorithm $clean: \mathcal{S} \rightarrow \mathcal{S}$ that performs tasks such as stop word removal, stemming, lemmatization, and spelling correction (Jurafsky and Martin, 2009). The resulting clean text for document i is denoted $t_i = clean(r_i)$.

Next, the feature type is chosen. For example, possible feature types include individual words (BOW); words and 2-combs; or words, 2-grams, and 3-grams. All F possible features of the chosen type are identified in the clean text. Let these features be indexed by $f \in \mathcal{B} = \{1, 2, \dots, F\}$. Furthermore, denote by x_{if} the binary variable indicating the presence of feature f in t_i :

$$x_{if} = \begin{cases} 1, & \text{if feature } f \text{ appears in } t_i \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

To reduce the number of features used in modeling, a frequency threshold τ is used (Man, 2014; Pôssas et al., 2002). Define

$$\mathcal{B}^{dict} = \left\{ f \in \mathcal{B}: \sum_{i=1}^N x_{if} \geq \tau \right\} \quad (2)$$

to be the set of features that occur in at least τ documents. These features are said to form the dictionary. Let $F^{dict} = |\mathcal{B}^{dict}|$ denote the dictionary size.

3.2 Weights

For each feature $f \in \mathcal{B}^{dict}$, a supervised weight w_f (Carvalho and Guedes, 2020; Erenel et al., 2011) is calculated based on the feature's class distribution. This distribution is defined by the following proportions. Denote by

$$p_{fd} = \frac{\sum_{i=1}^N 1(y_i = d)x_{if}}{\sum_{i=1}^N x_{if}} \quad (3)$$

the proportion of documents in the training data containing feature f that are in class d . This quantity is also referred to as dominance by Figueiredo et al. (2011). The weight w_f takes on values in the range 0 to 1, inclusive, and quantifies how concentrated the feature's class distribution is. Larger values of w_f mean the feature's distribution is more concentrated and, therefore, that the feature is more discriminative. This concept is equivalent to leaf node purity in decision trees (Tan et al., 2019). As in the decision tree setting, w_f can be defined in multiple ways:

$$w_f^{Max} = \left(\frac{D}{D-1} \right) \left[\max_d(p_{f1}, \dots, p_{fD}) - \frac{1}{D} \right] \quad (4)$$

$$w_f^{Range} = \max_d(p_{f1}, \dots, p_{fD}) - \min_d(p_{f1}, \dots, p_{fD}) \quad (5)$$

$$w_f^{Gini} = \left(\frac{D}{D-1} \right) \left[\left(\sum_{d=1}^D p_d^2 \right) - \frac{1}{D} \right] \quad (6)$$

$$w_f^{Entropy} = 1 + \sum_{d=1}^D p_{fd} \log_D(p_{fd}) \quad (7)$$

For the entropy definition, the logarithm is taken with respect to base D , and $0 \times \log_D(0)$ is defined to be 0. For these four definitions, $w_f = 0$ if and only if all proportions are equal, $p_{fd} = 1/D$ ($d = 1, \dots, D$). Conversely, $w_f = 1$ if and only if one proportion equals 1 and the other $D - 1$ proportions equal 0.

The weight can also be defined in terms of the feature’s class support, $D_f = |\{d: p_{fd} > 0\}|$. This is the number of classes in which feature f appears. The following are two definitions of w_f based on this concept:

$$w_f^{Support} = \left(\frac{D}{D-1}\right)\left(\frac{1}{D_f} - \frac{1}{D}\right) \quad (8)$$

$$w_f^{LogSupport} = 1 - \frac{\ln(D_f)}{\ln(D)} \quad (9)$$

For these two definitions, $w_f = 0$ if and only if $D_f = D$. Conversely, $w_f = 1$ if and only if $D_f = 1$. Setting the weight equal to a constant value is also an option:

$$w_f^{Equal} = 1 \quad (10)$$

3.3 Standard and umbrella vectorizers

To put the data in a format suitable for modeling, a vectorizer is applied to the clean text. The vectorizer is a function of the form $vectorize: \mathcal{S} \rightarrow [0,1]^{F^{dict}}$ that expresses the clean text as a vector of feature values incorporating the weights w_f . The vector for document i is given by

$$\mathbf{z}_i = vectorize(t_i) = \left[z_{i,h(1)}, z_{i,h(2)}, \dots, z_{i,h(F^{dict})} \right]^T \quad (11)$$

where the z_{if} , $f \in \mathcal{B}^{dict}$, are the feature values and $h: \{1,2, \dots, F^{dict}\} \rightarrow \mathcal{B}^{dict}$ is a bijection that makes the ordering of the features in vector \mathbf{z}_i well-defined.

For the “standard” vectorization, we have

$$z_{if} = w_f x_{if} \quad (12)$$

That is, \mathbf{z}_i simply identifies the dictionary features appearing in t_i with weights applied. We consider an alternative “umbrella” vectorization that can be used for feature types involving n-combs and n-grams. The umbrella vectorization restricts attention to higher-order n-combs and n-grams to reduce redundancy (Wan et al., 2019) and focus on detail. These higher-order composite features act like umbrellas that cover sub-features. They are similar to the closed and maximal termsets introduced by Pôssas et al. (2002), but the umbrella vectorization described here applies to both n-grams and n-combs. For this vectorization,

$$z_{if} = w_f u_{if} \quad (13)$$

where

$$u_{if} = \begin{cases} 1, & \text{if } x_{if} = 1 \text{ and } f \not\subseteq g \forall g \in \mathcal{B}^{dict} \text{ such that } x_{ig} = 1 \\ 0, & \text{otherwise} \end{cases} \quad (14)$$

is the umbrella indicator. The notation feature $f \not\subseteq$ feature g is meant to suggest that the n-comb (or n-gram) f is not a sub-feature of some larger n-comb (or n-gram) g .

As a demonstration of the two vectorizers, consider the following dictionary of words and 2-combs: **{market}**, **{pet}**, **{retail}**, **{store}**, **{market, retail}**, **{pet, retail}**, and **{pet, store}**. Tables 1 and 2 list the values of $w_f x_{if}$ and $w_f u_{if}$, respectively, for four hypothetical documents. For example, the first document has raw text “This is a Retail Market” and clean text “**retail market**”. The words **{market}** and **{retail}** and 2-comb **{market, retail}** appear in the clean text, so $w_f x_{if} = w_f \times 1 = w_f$ for these three features (see Table 1). However, **{market}** and **{retail}** are subsets of the 2-comb **{market, retail}**, so $w_f u_{if} = w_f \times 0 = 0$ for these two words (see Table 2).

Table 1. Example standard vectorizations: values of $w_f x_{if}$

f	Word or 2-comb	Raw text and corresponding clean text			
		This is a Retail Market	Market/Store	Store for Pets.	Retailer – Pet Store & Market
		retail market	market store	store pet	retail pet store market
1	{market}	w_1	w_1	0	w_1
2	{pet}	0	0	w_2	w_2
3	{retail}	w_3	0	0	w_3
4	{store}	0	w_4	w_4	w_4
5	{market, retail}	w_5	0	0	w_5
6	{pet, retail}	0	0	0	w_6
7	{pet, store}	0	0	w_7	w_7

Table 2. Example umbrella vectorizations: values of $w_f u_{if}$

f	Word or 2-comb	Raw text and corresponding clean text			
		This is a Retail Market	Market/Store	Store for Pets.	Retailer – Pet Store & Market
		retail market	market store	store pet	retail pet store market
1	{market}	0	w_1	0	0
2	{pet}	0	0	0	0
3	{retail}	0	0	0	0
4	{store}	0	w_4	0	0
5	{market, retail}	w_5	0	0	w_5

6	{pet, retail}	0	0	0	w_6
7	{pet, store}	0	0	w_7	w_7

3.4 Weighted dominance score function

Using the class proportions and choice of vectorizer, relevance scores can be calculated. Consider a new, previously unseen document with raw text r_{new} and unknown class y_{new} . This document goes through the same cleaning and vectorization process as the training data, yielding clean text $t_{new} = clean(r_{new})$ and feature vector $\mathbf{z}_{new} = vectorize(t_{new})$. The proposed weighted dominance score function calculates the relevance score for class d , denoted θ_d^{WgtDom} , to be the weighted average of the class proportions p_{fd} :

$$\theta_d^{WgtDom} = \frac{\sum_{f \in \mathcal{B}^{dict}} z_{new,f} p_{fd}}{\sum_{f \in \mathcal{B}^{dict}} z_{new,f}} \quad (15)$$

The θ_d^{WgtDom} , $d = 1, \dots, D$, take on values between 0 and 1, inclusive, and sum to 1. [In the case $\sum_{f \in \mathcal{B}^{dict}} z_{new,f} = 0$, the θ_d^{WgtDom} are defined to equal 0.] For ranked classification, the θ_d^{WgtDom} are used to rank the classes. For hard classification, the predicted class is the one with the highest relevance score:

$$\hat{y}_{new} = \underset{d}{\operatorname{argmax}}(\theta_d^{WgtDom}) \quad (16)$$

Conceptually, the weighted dominance score function averages the class distributions of the features, with more weight given to the distributions that are more concentrated. Figure 2 illustrates the relevance score calculation. It displays hypothetical class distributions of {**retail**}, {**market**}, and the 2-comb {**market, retail**} with weights equal to 0.125, 0.375, and 0.75, respectively. These weights are calculated according to the w_f^{Max} definition (Equation 4). The co-occurrence feature {**market, retail**} has the most concentrated distribution.

In the context of the standard vectorization, {**market, retail**} has the most influence in the weighted average of the features' distributions. In this case, the weighted dominance score function produces the following class ranking: class 3 ($\theta_3^{WgtDom} = 0.66$), class 4 ($\theta_4^{WgtDom} = 0.215$), class 1 ($\theta_1^{WgtDom} = 0.065$), class 5 ($\theta_5^{WgtDom} = 0.035$), and class 2 ($\theta_2^{WgtDom} = 0.025$). Note that every class is assigned a positive score.

However, if the umbrella vectorization is used, the distributions of the words {**retail**} and {**market**} are not considered. The relevance scores simply equal the class proportions of the umbrella feature {**market, retail**}. In this case, the weighted dominance score function produces the following class ranking: class 3 ($\theta_3^{WgtDom} = 0.8$) and class 4 ($\theta_4^{WgtDom} = 0.2$).

The proportions for classes 1, 2, and 5 equal 0, so these classes are assigned a score of 0. Compared to the standard vectorizer, the weighted dominance score function with the umbrella vectorizer assigns a higher score to class 3 and is more confident in its being the true class.

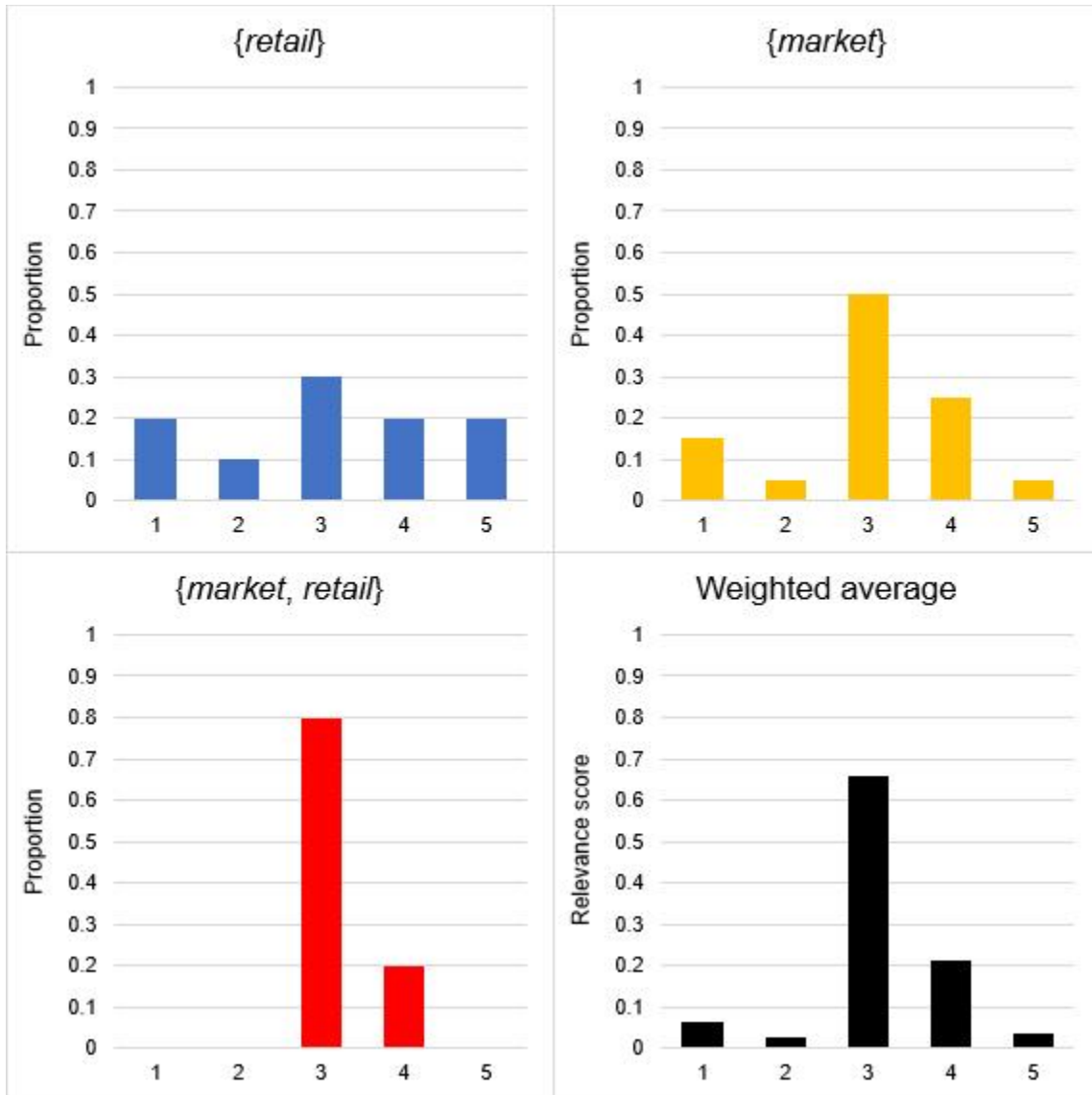


Figure 2. Illustration of relevance score calculation. This calculation is based on the features {retail}, {market}, and {market, retail} with weights 0.125, 0.375, and 0.75, respectively. The class distribution of {market, retail} over the five classes is the most concentrated and thus has the most influence in the weighted average.

4. Datasets

This section describes four datasets suitable for ranked short text classification. We use these datasets in a comparative study of proposed and baseline methods

4.1 HuffPost

HuffPost is an online news aggregator and blog. A related dataset is available on Kaggle that consists of 200,853 HuffPost article headlines from 2012 to 2018 (Misra, 2018; Grover and Misra, 2021). The headlines are classified according to 41 topics including politics, entertainment, travel, comedy, sports, crime, and world news. The dataset also contains a text variable that summarizes the news article, but we focus on using the shorter headline to classify the topic.

4.2 NAICS

The North American Industry Classification System (NAICS) is a hierarchical coding system used by government agencies to classify the principal economic activity of business establishments (U.S. Census Bureau, 2021a). For example, Economic Census (U.S. Census Bureau, 2021b) data products released by the U.S. Census Bureau are based on NAICS. NAICS codes are six digits long with the first two digits identifying the economic sector. There are 20 sectors including construction, manufacturing, wholesale trade, and accommodation and food services. To research sector classification based on short text, we compiled business descriptions and corresponding NAICS codes from two data sources: the Economic Census (2002, 2007, 2012, and 2017) and the Internal Revenue Service's SS-4 form (Internal Revenue Service, 2021) (2002 through 2016), which is used by employers for tax purposes. Both the Economic Census questionnaire and SS-4 form allow respondents to write an open-ended description of their business (for example, products sold and services performed). The dataset consists of 2,192,275 documents.

4.3 NAICS Wholesale

To examine NAICS classification at a more granular level, we focus on one economic sector from the NAICS dataset and consider the more challenging problem of classifying at the 6-digit level. Wholesale trade is one of the largest sectors in the NAICS dataset with 271,025 documents and 71 classes, making this dataset interesting for evaluating ranked classifiers.

4.4 Stack Overflow

Stack Overflow is a popular question-and-answer website for computer programmers. A related dataset is made available by Xu et al. (2015 and 2017) and Kaggle that represents a sample of 20,000 Stack Overflow thread titles evenly distributed over 20 topics. The topics touch on computer applications, operating systems, and programming languages. Examples include Oracle, SVN, Apache, Excel, MATLAB, SharePoint, and Ajax. The problem is to classify the topic under discussion based on the thread title.

4.5 Summary of datasets

Table 3 presents descriptive statistics for the four datasets. The class imbalance degree is the measure described by Ortigosa-Hernández et al. (2017) and is based on Euclidean distance. We

divide this imbalance degree by the number of classes to allow comparison among datasets. A related measure is the class imbalance ratio, which equals the maximum class proportion divided by the minimum class proportion. The HuffPost and two NAICS datasets exhibit class imbalance, whereas the Stack Overflow dataset is, by design, perfectly balanced. In terms of characters and words, the documents for the two NAICS datasets are shorter on average than the documents for the HuffPost and Stack Overflow datasets. Words are defined as sequences of non-whitespace characters.

Table 3. Dataset descriptive statistics

Dataset	Number of documents	Number of classes	Class imbalance degree	Class imbalance ratio	Raw text length (characters)		Raw text length (words*)	
					Average	Max	Average	Max
HuffPost	200,853	41	0.71	32.61	57.94	320	9.54	44
NAICS	2,192,275	20	0.47	323.58	26.33	268	3.52	49
NAICS Wholesale	271,025	71	0.65	121.99	28.43	259	3.76	43
Stack Overflow	20,000	20	0.00	1.00	48.69	213	8.00	34

*Defined here as strings of non-whitespace

5. Evaluation

We compare the performance of the weighted dominance score function to that of three other classifiers: a modified BM25 score function, multinomial logistic regression, and decision tree. The modified BM25 score is given by

$$\theta_d^{BM25} \propto \sum_{f \in \mathcal{B}^{dict}} \left[z_{new,f} \ln \left(\frac{D+1}{D_f+0.5} \right) \frac{(m+1)N_{fd}}{m \left[(1-b) + b \left(\frac{N_d D}{N} \right) \right] + N_{fd}} \right] \quad (17)$$

where $N_{fd} = \sum_{i=1}^N 1(y_i = d)x_{if}$ and $N_d = \sum_{i=1}^N 1(y_i = d)$. The argument to the natural logarithm has been increased by one to ensure a positive score, and the BM25 factor accounting for within-query term frequencies has been omitted (Robertson and Zaragoza, 2009). The values of m and b are set to 1.2 and 0.75, respectively. The logistic regression and decision tree classifiers produce class probabilities, which can be used like scores to rank classes. The weighted dominance and modified BM25 classifiers are implemented using base Python functions and data structures. The logistic regression and decision tree classifiers, on the other hand, are fit using the Natural Language Toolkit (NLTK) (Bird et al., 2009) and Scikit-learn (Pedregosa et al., 2011) modules. For these two classifiers, most parameters are set to their default values. Results are not believed to be sensitive to software implementation.

In terms of features, we study all combinations of feature type (BOW, words and 2-combs, and words and 2-grams) and vectorizer (standard and umbrella). There are only five such combinations because the standard and umbrella vectorizers are the same for BOW. We do

not consider n-combs and n-grams longer than two because of diminishing returns for the increase in complexity; Choueka and Lusignan (1985) show that simply adding a second word of context can greatly help with term disambiguation. We also consider three weight types representative of the different weight definitions: w_f^{Equal} , $w_f^{LogSupport}$, and w_f^{Max} . There are 15 ($= 5 \times 3$) feature/weight combinations in total.

The same cleaning algorithm is applied to the four datasets. This algorithm consists of converting text to lowercase, removing punctuation, removing extraneous whitespace, removing stop words, and stemming (stripping suffixes to reduce the number of word variations). The stop words are based on a default list from NLTK. NLTK is also used to implement the popular Porter 2/Snowball stemmer (Porter, 2001). In practice, the stop words and stemmer would be tailored to the dataset. For example, additional rules may be added to the stemmer to address known under- and over-stemming errors. In all cases, the frequency threshold τ used to determine the dictionary is set to 3.

To evaluate the various combinations of classifier, features, and weight type, we apply the holdout method (Tan et al., 2019). For each dataset, we select a stratified random sample with classes as strata. The sampling fraction in each stratum is set to 0.9. The selected documents form the training data, and the remaining documents form the test data. The training data are used to determine the dictionary and fit the various classifiers. Finally, the fitted classifiers are applied to and evaluated on the test data. Table 4 describes the metrics used to assess performance. These metrics are mean reciprocal rank (MRR), mean score of the true class (MSCORE), accuracy (TOP- k), time to fit the classifier, and time to apply the classifier.

Table 4. Evaluation metrics

Metric	Description
MRR	Mean reciprocal rank of the true class. If the true class is assigned a score of 0, its reciprocal rank is defined to be 0.
MSCORE	Mean score of the true class. This metric reflects model confidence in predicting the true class.
TOP- k	Accuracy, where success is defined as the true class being among the k highest-scoring classes. This metric is calculated for different values of k . The case $k = 1$ corresponds to “hard” classification accuracy.
Fitting time	Time (in seconds) it takes to fit the classifier on the training data.
Application time	Time (in seconds) it takes to apply the classifier to the test data on a document-by-document basis (i.e., not batch processing).

6. Results

For each metric, we present global values and then focus on a specific dataset to highlight interesting results. The four classifiers are abbreviated WgtDom, BM25, LogReg, and DecTree. The five sets of features are abbreviated BOW, CS, CU, GS, and GU. C and G stand for n-combs

and n-grams, respectively. S and U stand for the standard and umbrella vectorizers, respectively.

6.1 Dictionary size

Table 5 presents the dictionary sizes for the four datasets. As a reminder, the dictionaries are determined using the documents randomly selected for training. The dictionary for the NAICS dataset is the largest with 150,456 2-grams and 313,034 2-combs. In general, there are approximately 2-4 times as many 2-combs as 2-grams. Even though the HuffPost dataset has fewer documents than the NAICS Wholesale dataset, its dictionary is appreciably larger.

Table 5. Dictionary sizes

Dataset	Words	2-grams	2-combs
HuffPost	17,048	49,770	238,768
NAICS	19,324	150,456	313,034
NAICS Wholesale	5,995	27,702	55,622
Stack Overflow	2,515	3,733	14,753

6.2 MRR

Figure 3 presents the largest value of MRR for each combination of classifier and dataset. The corresponding features and weight type are identified by color and symbol, respectively. Between n-combs and n-grams, the n-comb features (CS and CU) occur more frequently in Figure 3 and are prevalent for WgtDom and BM25. The performance of WgtDom is most similar to that of BM25, with WgtDom producing slightly larger MRR values across nearly all feature/weight combinations for each dataset. This is understandable given that WgtDom has a similar model structure to BM25 but, arguably, makes better use of the features' class distributions. There are mixed results regarding weight type, although $w_f^{LogSupport}$ tends to perform the best for DecTree.

WgtDom with CS features produces the largest value of MRR for the Stack Overflow dataset. LogReg produces the largest value of MRR for the other three datasets. In terms of MRR, LogReg has an advantage over the other classifiers. It always assigns a positive score to the true class, and thus the reciprocal rank is always positive for a given document. WgtDom and BM25 are limited by their features' support, resulting in a reciprocal rank equal to 0 for some documents. Similarly, DecTree is limited by the classes represented in the leaf node at which classification occurs.

Classifier	Dataset			
	HuffPost	NAICS	NAICS Wholesale	Stack Overflow
BM25	○ 0.6705	● 0.8032	● 0.5516	● 0.9036
DecTree	◐ 0.5771	○ 0.7867	◐ 0.5392	◐ 0.8692
LogReg	○ 0.7199	● 0.8229	○ 0.5888	● 0.9093
WgtDom	● 0.6747	○ 0.8173	● 0.5867	○ 0.9126

Features and weight type

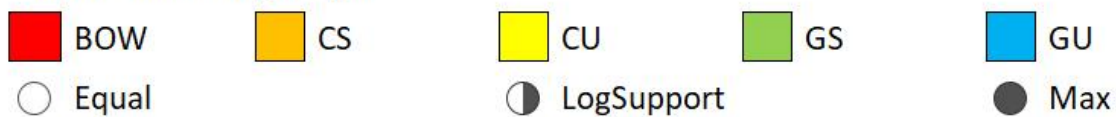


Figure 3. Largest value of MRR for each combination of classifier and dataset. The corresponding features and weight type are identified by color and symbol, respectively.

Figure 4 shows MRR for each combination of feature and weight type for the NAICS Wholesale dataset. The features and weight type are identified by color and shading, respectively. Note that the y-axis starts at 0.4 to emphasize differences. BM25, WgtDom, and LogReg perform the worst when $w_f^{LogSupport}$ is used. Weight type has some effect on DecTree, but this is not consistent across all datasets.

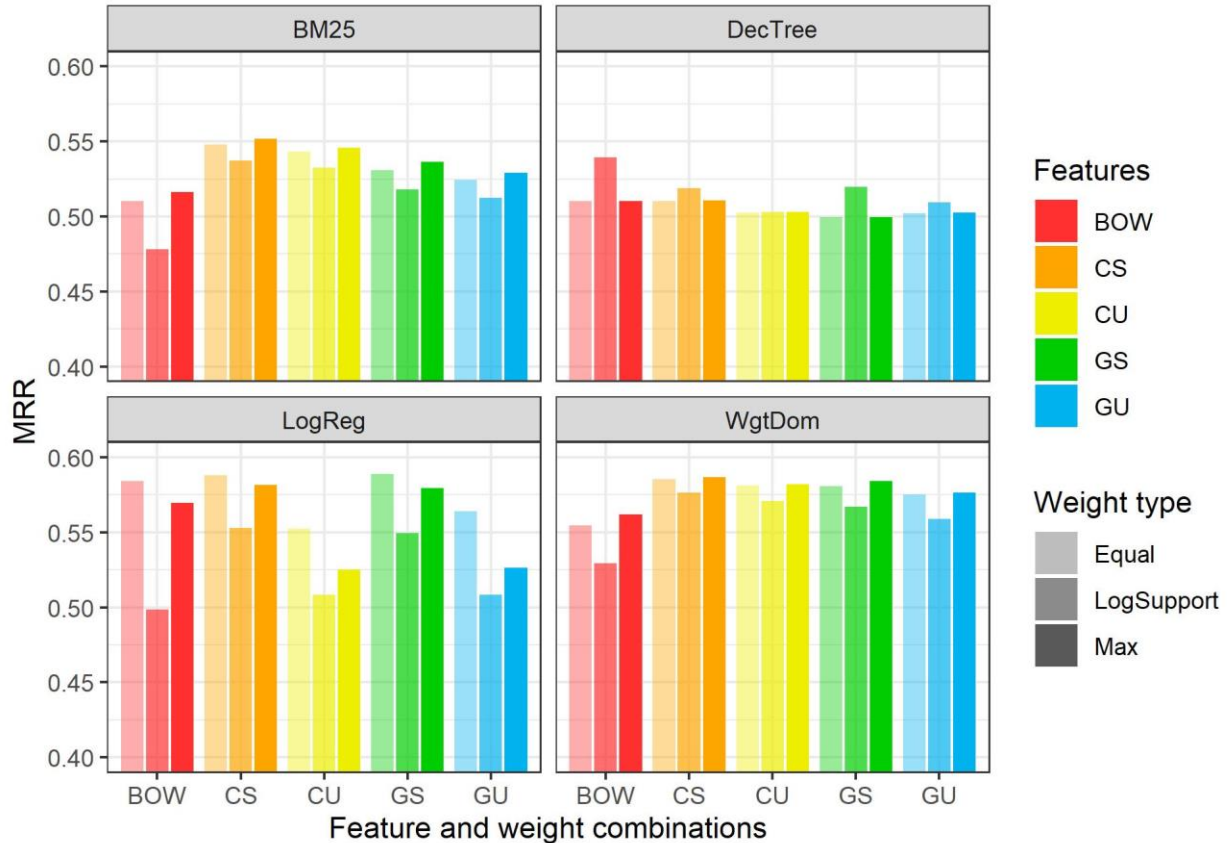


















Figure 4: MRR by feature and weight type for the NAICS Wholesale dataset. The features and weight type are identified by color and shading, respectively. Note that the y-axis starts around 0.4 to emphasize differences.

6.3 MSCORE

Larger values of MSCORE indicate higher confidence in predicting the true class. This is an important consideration if results are filtered using a score threshold. Figure 5 presents the largest value of MSCORE for each combination of classifier and dataset. The CU and GU features are prevalent for WgtDom and BM25. By restricting attention to higher order composite features, the umbrella vectorization helps the two score functions focus on detail and assign higher scores to the true class. On the other hand, the CS and GS features are prevalent for LogReg and DecTree. For the MSCORE metric, there is more of a pattern regarding weight type. The weight type w_f^{Max} works best for WgtDom, w_f^{Equal} for LogReg, and $w_f^{LogSupport}$ for BM25 and DecTree.

For all four datasets, LogReg and DecTree produce the largest and second largest values of maximum MSCORE in Figure 5; WgtDom produces the third largest value. However, for particular datasets and combinations, WgtDom does outperform LogReg. For example, see Figure 6, which shows MSCORE for each combination of feature and weight type for the Stack

Overflow dataset. WgtDom produces a higher MSCORE than LogReg for two of the three combinations involving the CU features. For WgtDom and BM25, the CU features perform better than CS for all weight types. WgtDom and BM25 perform the worst when w_f^{Equal} is used. This demonstrates that, on average, the supervised weights help the score functions assign large scores to the true class. LogReg performs the worst when $w_f^{LogSupport}$ is used. In contrast, the weight type has little effect on DecTree.

Classifier	Dataset			
	HuffPost	NAICS	NAICS Wholesale	Stack Overflow
BM25	 0.1933	 0.2044	 0.1131	 0.5679
DecTree	 0.3913	 0.5977	 0.3071	 0.8151
LogReg	 0.4791	 0.6117	 0.3144	 0.7963
WgtDom	 0.3326	 0.5469	 0.2704	 0.6813

Features and weight type

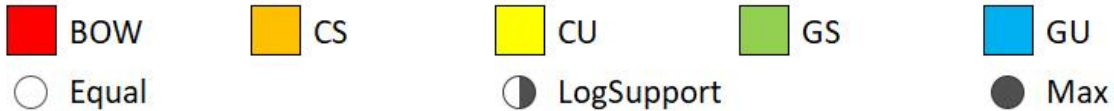


Figure 5. Largest value of MSCORE for each combination of classifier and dataset. The corresponding features and weight type are identified by color and symbol, respectively.

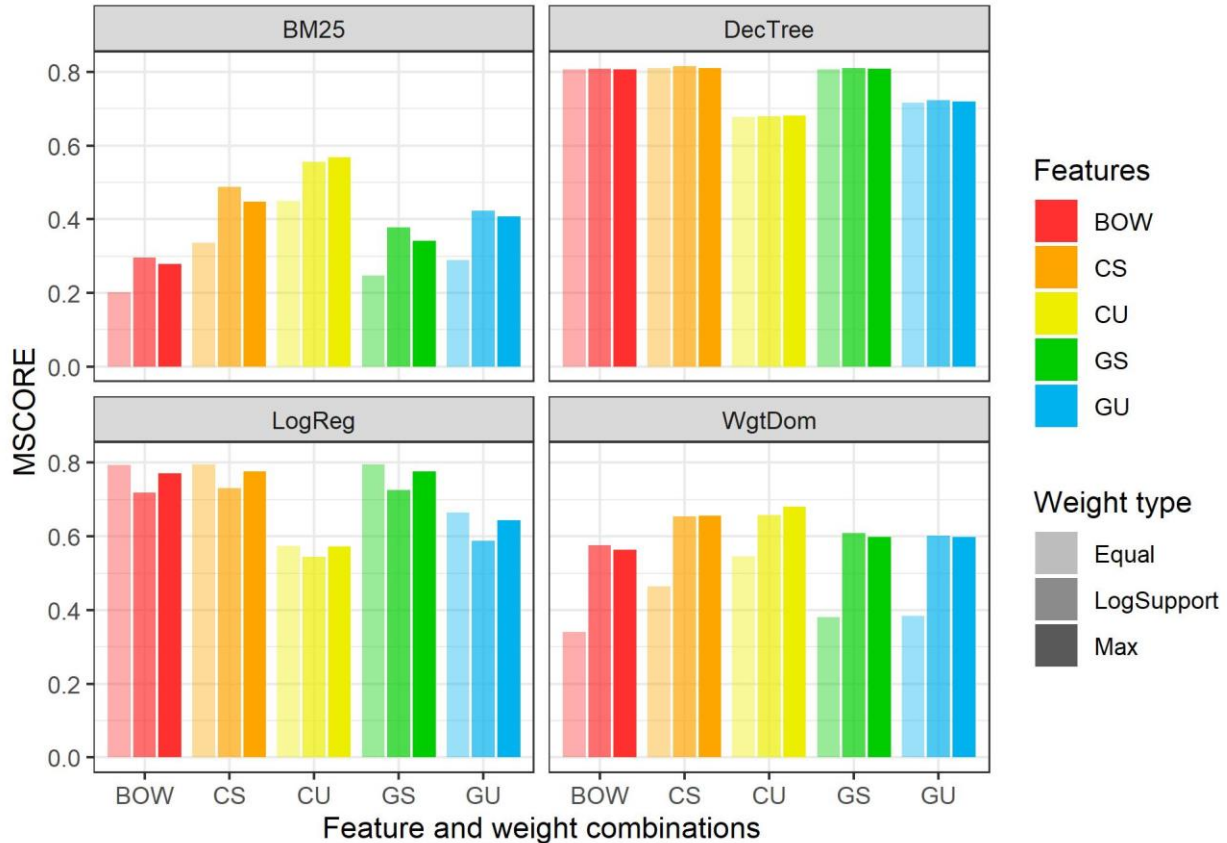


Figure 6: MSCORE by feature and weight type for the Stack Overflow dataset. The features and weight type are identified by color and shading, respectively.

6.4 TOP- k

The TOP- k accuracy measures how successful the classifier is in ranking the true class among the k highest-scoring results. Ranking the true class first is more difficult than ranking it among the top k results when k is large. Figure 7 presents the largest value of TOP-5 for each combination of classifier and dataset. Results differ for different values of k . The choice $k = 5$ seems reasonable given the number of classes in each dataset. The CS and BOW features are the most prevalent in Figure 7, with the GS features yielding the largest value of TOP-5 for LogReg for two datasets. The CU and GU features do not correspond to any values in Figure 7. Regarding weight type patterns, w_f^{Max} performs the best for WgtDom, whereas w_f^{Equal} performs the best for LogReg. According to TOP-5, the classifiers can be ranked from most to least accurate roughly as LogReg, WgtDom, BM25, and DecTree. LogReg, WgtDom, and BM25 perform similarly on the Stack Overflow dataset. LogReg and WgtDom perform similarly on the NAICS Wholesale dataset.

Classifier	Dataset			
	HuffPost	NAICS	NAICS Wholesale	Stack Overflow
BM25	○ 0.8294	○ 0.9417	● 0.6992	● 0.9520
DecTree	◐ 0.6976	○ 0.9020	◐ 0.6705	○ 0.9065
LogReg	○ 0.8755	○ 0.9529	○ 0.7449	○ 0.9560
WgtDom	● 0.8466	● 0.9485	● 0.7423	● 0.9560

Features and weight type

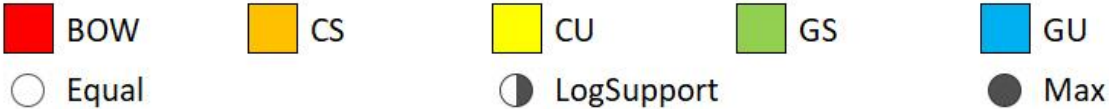


Figure 7. Largest value of TOP-5 for each combination of classifier and dataset. The corresponding features and weight type are identified by color and symbol, respectively.

Figure 8 displays TOP- k of each classifier on the NAICS Wholesale dataset for selected values of k . We focus on results using the weight type w_f^{Max} . Although WgtDom either performs the best or essentially as well as any other classifier for this particular dataset, that is not always true. As shown previously, for other datasets, LogReg is the most accurate, depending on the weight type. DecTree is generally the least accurate. WgtDom tends to improve upon BM25 consistently in terms of TOP- k accuracy.

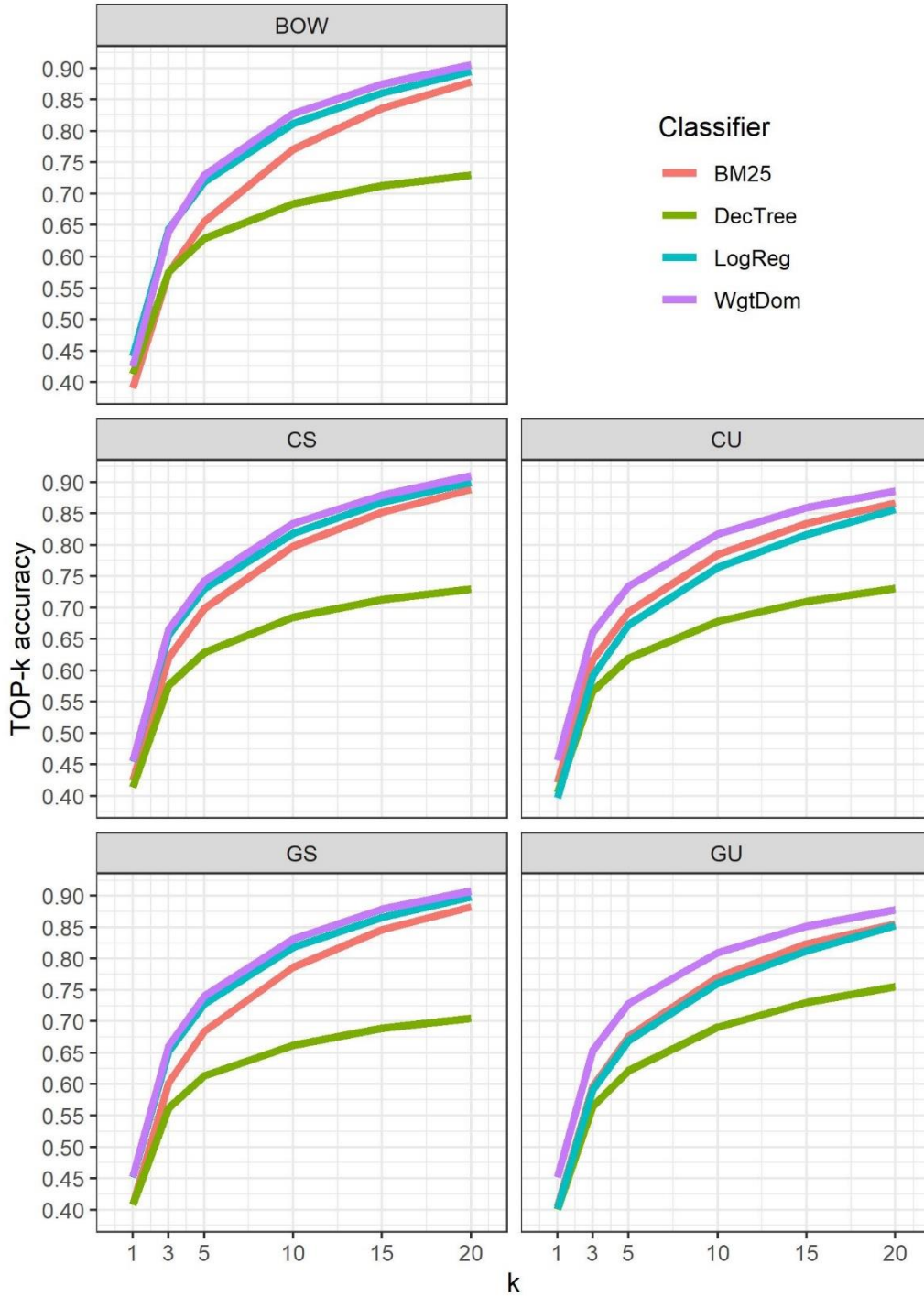


Figure 8. TOP- k by classifier and feature for the NAICS Wholesale dataset. Results are displayed for the weight type w_f^{Max} and for selected values of k .

6.5 Fitting time

Fitting WgtDom is defined as calculating the proportions p_{fd} appearing in Equation 15 for θ_d^{WgtDom} . Similarly, fitting BM25 is defined as calculating the quantities D_f , N_{fd} , and N_d

appearing in Equation 17 for θ_d^{BM25} . As expected, for all datasets, LogReg and DecTree fit the slowest. The fitting times for WgtDom and BM25 are negligible in comparison. DecTree fits more slowly for the CU and GU features compared to CS and GS. The opposite is true for LogReg. Whereas the choice of weight has little impact on the fitting time of DecTree, the effect of using w_f^{Equal} is much more apparent for LogReg. Figure 9 displays fitting times for DecTree and LogReg on the NAICS dataset, the largest of the four datasets. The maximum values for WgtDom, BM25, LogReg, and DecTree for this dataset are 17.8, 17.6, 2,729, and 5,328 seconds, respectively.

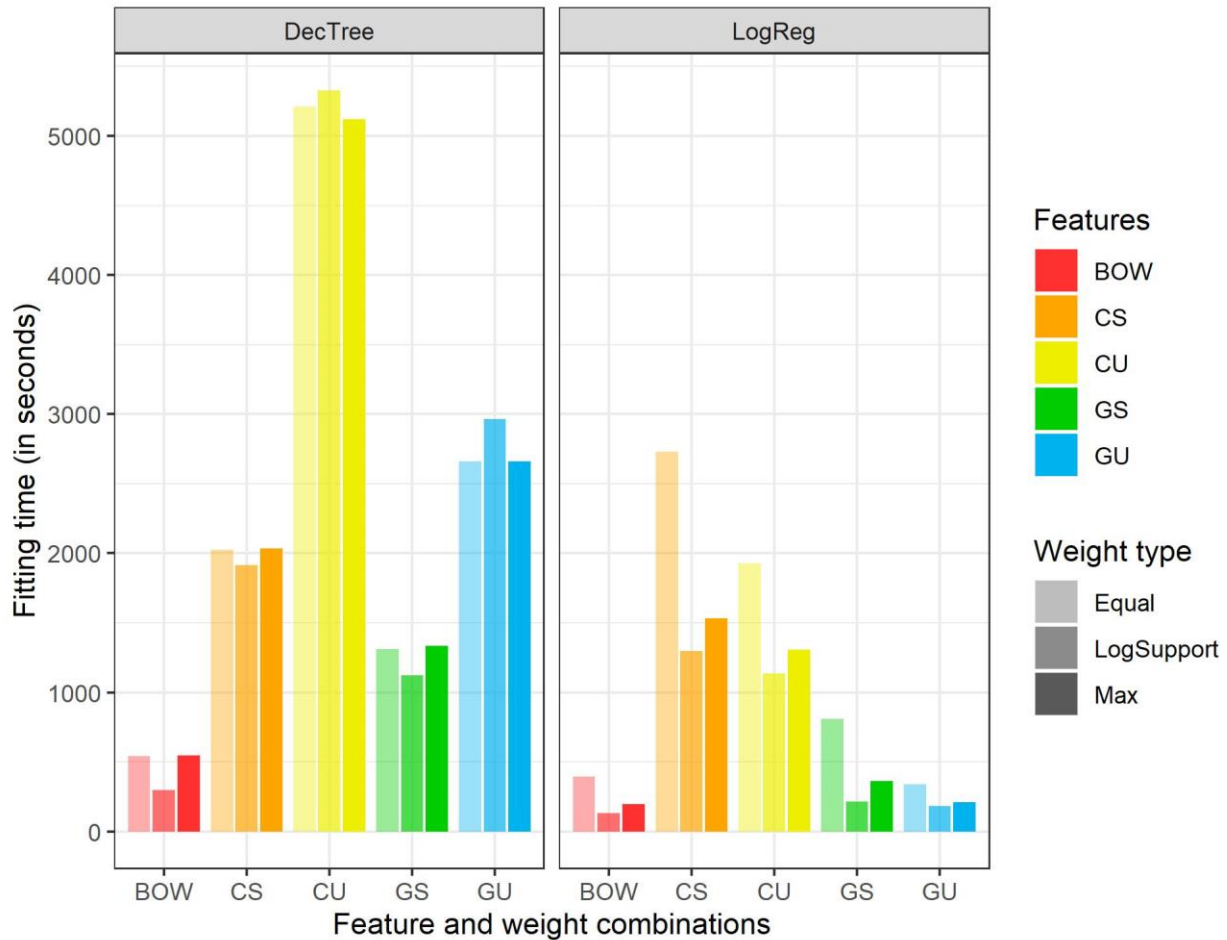


Figure 9. Fitting times (in seconds) for DecTree and LogReg by feature and weight type on the NAICS dataset. The features and weight type are identified by color and shading, respectively. The fitting times for WgtDom and BM25 are negligible in comparison to those for DecTree and LogReg.

6.6 Application time

The classifiers are applied to the test data one document at a time, as opposed to using a batch method Scikit-learn method available to LogReg and DecTree. This allows for a fairer

comparison and better simulates information retrieval applications. Applying WgtDom and BM25 is fast for all datasets. The application time for LogReg is much longer. Figure 10 displays application times for DecTree and LogReg on the NAICS dataset. The maximum values for WgtDom, BM25, LogReg, and DecTree for this dataset are 9.9, 23.7, 2,475, and 117 seconds, respectively.

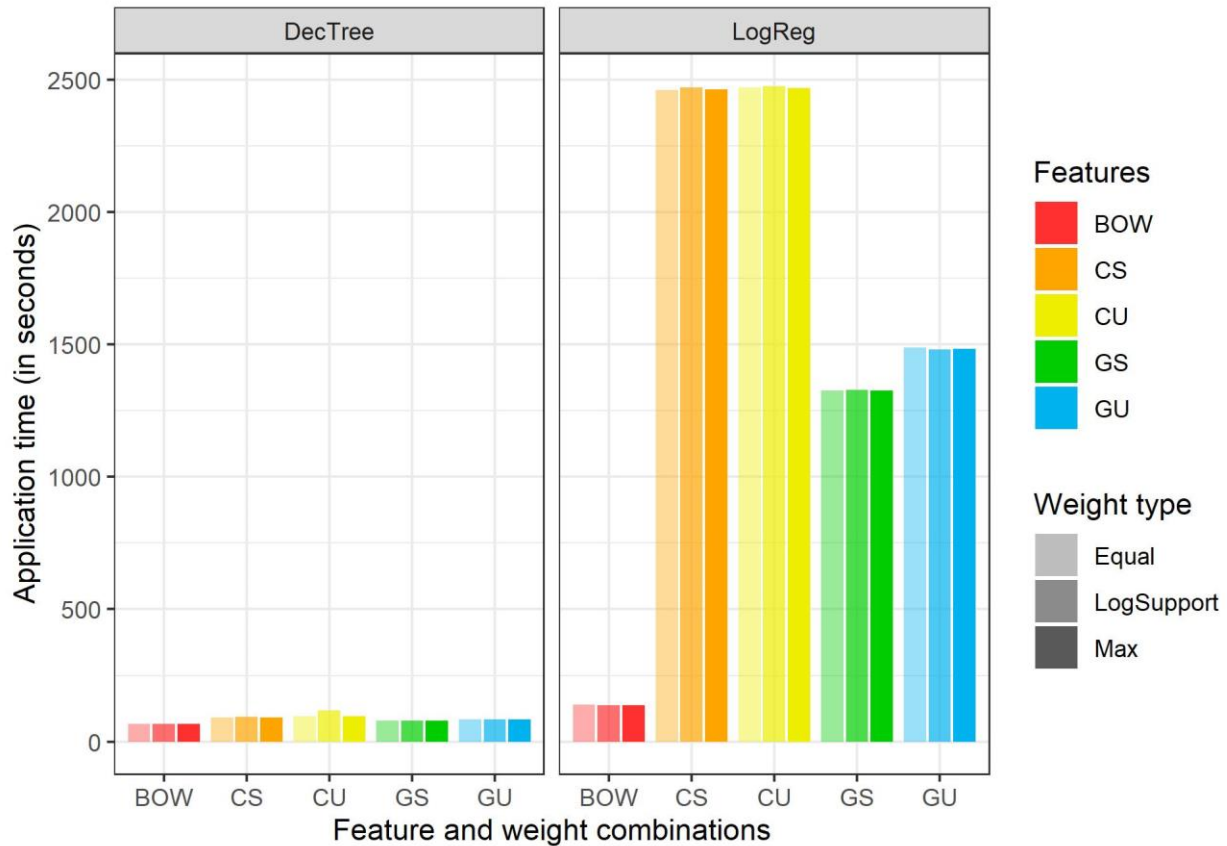


Figure 10. Application times (in seconds) for DecTree and LogReg by feature and weight type on the NAICS dataset. The features and weight type are identified by color and shading, respectively. The application times for WgtDom and BM25 are negligible in comparison to those for LogReg.

7. Conclusions and future work

In the context of ranked short text classification, we examine the use of co-occurrence features and an alternative VSM and score function. N-combs are both feasible and appropriate for short text, making them an interesting option in this setting. An evaluation was conducted in which four classifiers were compared across four datasets that represent news headlines, business descriptions, and online forum postings. We considered 15 combinations of features and weight type. The classifiers were evaluated according to five metrics. Three metrics quantify accuracy, and the other two measure the time required to fit and apply the classifier.

The results show the classifiers have different strengths and weaknesses, but there are also some patterns. The umbrella vectorization appears better suited for score functions than for the more traditional multinomial logistic regression and decision tree classifiers. N-combs performed well in general, but the advantage of using them over n-grams, as measured by MRR, MSCORE, and TOP- k , is clearer for the score functions. As a generalized linear model, logistic regression is versatile and performs very well on all four datasets. However, as expected, it takes a long time to fit and apply. Logistic regression does not scale well like the weighted dominance and modified BM25 score functions. Score functions in general can handle a large number of documents, features, and classes. Weighted dominance outperforms the modified BM25 with respect to the accuracy metrics. For these reasons, the weighted dominance classifier based on n-combs could serve as a useful baseline classifier for ranked short text classification, especially for large datasets.

With regard to the performance of the BM25, logistic regression, and decision tree classifiers, there are potential gains by fine-tuning the model parameters. The decision tree classifier, in particular, has many parameters that govern complexity. The default minimum sample size needed to terminate branches was increased to try to protect against model overfitting, but parameter values are not optimal. Decision tree overfitting is reflected in large values of MSCORE and small values of MRR and TOP- k . Still, the evaluation shows the much simpler weighted dominance score function performs at the same level as the other classifiers.

In terms of improving the weighted dominance score function, it would be interesting to research a hybrid approach based on both CS and CU features. The CS features perform well in terms of MRR and TOP- k , whereas the umbrella vectorization clearly performs the best in terms of MSCORE. CS allows the weighted dominance score function to consider all words and word combinations in a given document. This results in a wider range of classes that are assigned a positive score. CU, on the other hand, restricts positive score assignment to a subset of these classes. Employing a model ensemble (Tan et al., 2019) based on both sets of features could result in a classifier that optimally balances these metrics.

Another idea for future work is using model stacking (Pavlyshenko, 2018; Güneş, 2017), or meta ensembling, to transform and weight the scores from CS- and CU-based classifiers instead of simply averaging them. Such an approach could adaptively account for the differences of each classifier and the unique characteristics of each class. Any insights uncovered through such second-stage modeling could also motivate internal refinements of the weighted dominance score function.

Abbreviations

BOW: bag-of-words; MRR: mean reciprocal rank of the true class; MSCORE: mean score of the true class; NAICS: North American Industry Classification System; NLTK: Natural Language Toolkit; TF-IDF: term frequency-inverse document frequency; TOP- k : accuracy, where success is defined as the true class being among the k highest-scoring classes; VSM: Vector Space Model

Acknowledgments

The authors would like to acknowledge and thank colleagues at the U.S. Census Bureau for reviewing drafts of this article and providing helpful comments.

Declarations

Disclaimer

Any opinions and conclusions expressed herein are those of the authors and do not reflect the views of the U.S. Census Bureau. The Census Bureau has reviewed this data product for unauthorized disclosure of confidential information and has approved the disclosure avoidance practices applied (Approval ID: CBDRB-FY22-ESMD001-008).

Funding

Not applicable.

Competing interests

The authors declare that they have no competing interests.

Ethics approval

Not applicable.

Consent to participate

Not applicable.

Consent for publication

Not applicable.

Availability of data and materials

The HuffPost and Stack Overflow datasets are publicly available at the references cited. The NAICS dataset, which consists of confidential data collected by the U.S. Census Bureau and Internal Revenue Service, is not publicly available.

Code availability

The code is available upon request.

Authors' contributions

Brian Dumbacher is the primary author and led work on the proposed methodology. Daniel Whitehead helped conduct the evaluation, assess findings, and create visualizations. All authors read and approved the final manuscript.

References

- Aggarwal, C.C. (2018). *Machine learning for text*. Cham, Switzerland: Springer International Publishing.
- Badawi, D. (2015). *Termset selection and weighting in binary classification*. Ph.D. dissertation. Eastern Mediterranean University.
- Bird, S., Klein, E., and Loper, E. (2009). *Natural language processing with Python: analyzing text with the natural language toolkit*. Sebastopol, CA: O'Reilly Media.
- Carvalho, F. and Guedes, G.P. (2020). TF-IDFC-RF: a novel supervised term weighting scheme. *arXiv*. doi:10.48550/arXiv.2003.07193
- Choueka, Y. and Lusignan, S. (1985). Disambiguation by short text contexts. *Computers and Humanities*, 19, 147-157.
- Cunningham, S.J., Littin, J., and Witten, I.H. (1997). Applications of machine learning in information retrieval. Working paper. Hamilton, New Zealand: University of Waikato, Department of Computer Science.
- Elnagar, A., Al-Debsi, R., and Einea, O. (2020). Arabic text classification using deep learning models. *Information Processing & Management*, 57(1), 102121. doi:10.1016/j.ipm.2019.102121
- Erenel, Z., Altınçay, H., and Varoğlu, E. (2011). Explicit use of term occurrence probabilities for term weighting in text categorization. *Journal of Information Science and Engineering*, 27, 819-834.
- Figueiredo, F., Rocha, L., Couto, T., Salles, T., Gonçalves, M.A., and Meira Jr., W. (2011). Word co-occurrence features for text classification. *Information Systems*, 36, 843-858. doi:10.1016/j.is.2011.02.002
- Ghiassi, M., Skinner, J., and Zimbra, D. (2013). Twitter brand sentiment analysis: a hybrid system using n -gram analysis and dynamic artificial neural network. *Expert Systems with Applications*, 40(16), 6266-6282. doi:10.1016/j.eswa.2013.05.057
- Giorgetti, D. and Sebastiani, F. (2003). Automating survey coding by multiclass text categorization techniques. *Journal of the American Society for Information Science and Technology*, 54(14), 1269-1277. doi:10.1002/asi.10335
- Goswami, P. (2014). *Learning information retrieval functions and parameters on labeled collections*. Ph.D. dissertation. Université Joseph Fourier.

- Grover, J. and Misra, R. (2021). *Sculpting data for ML: the first act of machine learning*. January 2021.
- Güneş, F. (2017). Why do stacked ensemble models win data science competitions? *The SAS Data Science Blog*. May 18, 2017.
<https://blogs.sas.com/content/subconsciousmusings/2017/05/18/stacked-ensemble-models-win-data-science-competitions/>. Accessed April 21, 2022.
- Internal Revenue Service. (2021). Form SS-4: Application for Employer Identification Number.
<https://www.irs.gov/pub/irs-pdf/fss4.pdf>. Accessed September 30, 2021.
- Jurafsky, D. and Martin, J.H. (2009). *Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition*. Upper Saddle River, NJ: Pearson Education, Inc.
- Lei, K., Fu, Q., Yang, M., and Liang, Y. (2020). Tag recommendation by text classification with attention-based capsule network. *Neurocomputing*, 391, 65-73.
doi:10.1016/j.neucom.2020.01.091
- Man, Y. (2014). Feature Extension for Short Text Categorization Using Frequent Term Sets. 2nd International Conference on Information Technology and Quantitative Management ITQM 2014. *Procedia Computer Science*, 31, 663-670. doi:10.1016/j.procs.2014.05.314
- Mascio, A., Kraljevic, Z., Bean, D., Dobson, R., Stewart, R., Bendayan, R., and Roberts, A. (2020). Comparative analysis of text classification approaches in electronic health records. *arXiv*.
doi:10.48550/arXiv.2005.06624
- Melville, P., Gryc, W., and Lawrence, R.D. (2009). Sentiment analysis of blogs by combining lexical knowledge with text classification. *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1275-1284.
doi:10.1145/1557019.1557156
- Misra, R. (2018). News category dataset. <https://www.kaggle.com/rmisra/news-category-dataset>. doi:10.13140/RG.2.2.20331.18729.
- Nagwani, N.K. and Sharaff, A. (2017). SMS spam filtering and thread identification using bi-level text classification and clustering techniques. *Journal of Information Science*, 43(1), 75-87.
doi:10.1177/0165551515616310
- Ortigosa-Hernández, J., Inza, I., and Lozano, J.A. (2017). Measuring the class-imbalance extent of multi-class problems. *Pattern Recognition Letters*, 98, 32-38.
doi:10.1016/j.patrec.2017.08.002
- Pavlyshenko, B. (2018). Using stacking approaches for machine learning models. *2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP)*, 2018, 255-258. doi:10.1109/DSMP.2018.8478522
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, É. (2011). Scikit-learn: machine learning in Python. *Journal of Machine Learning Research*, 12, 2825-2830.

- Pekar, V., Krkoska, M., and Staab, S. (2004). Feature weighting for co-occurrence-based classification of words. *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*, 799-805.
- Porter, M.F. (2001). Snowball: a language for stemming algorithms. <http://snowball.tartarus.org/texts/introduction.html>. Accessed April 4, 2022.
- Pôssas, B., Ziviani, N., Meira Jr., W., and Ribeiro-Neto, B. (2002). Set-based model: a new approach for information retrieval. *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 230-237.
- Robertson, S. and Zaragoza, H. (2009). The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4), 333-389. doi:10.1561/15000000019
- Salminen, J., Yoganathan, V., Corporan, J., Jansen, B.J., and Jung, S.-G. (2019). Machine learning approach to auto-tagging online content marketing efficiency: a comparative analysis between methods and content type. *Journal of Business Research*, 101, 203-217. doi:10.1016/j.jbusres.2019.04.018
- Salton, G., Wong, A., and Yang, C.-S. (1975). A vector space model for automatic indexing. *Communications of the ACM*, 18(11), 613-620. doi:10.1145/361219.361220
- Schierholz, M. and Schonlau, M. (2021). Machine learning for occupation coding—a comparison study. *Journal of Survey Statistics and Methodology*, 9(5), 1013-1034. doi:10.1093/jssam/smaa023
- Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1), 1-47.
- Soumya, G.K. and Shibily, J. (2014). Text classification by augmenting bag of words (BOW) representation with co-occurrence feature. *IOSR Journal of Computer Engineering (IOSR-JCE)*, 16, 34-38. doi:10.9790/0661-16153438
- Tan, P.-N., Steinbach, M., Karpatne, A., and Kumar, V. (2019). *Introduction to data mining (2nd edition)*. New York: Pearson Education, Inc.
- Tarnow-Mordi, R. (2017). The intelligent coder: developing a machine-learning classification system. Australian Bureau of Statistics Methodological News 1504.0. Issued September 21, 2017. <https://www.abs.gov.au/ausstats/abs@.nsf/Previousproducts/1504.0Main%20Features5Sep%202017>. Accessed October 1, 2021.
- Turney, P.D. and Pantel, P. (2010). From frequency to meaning: vector space models of semantics. *Journal of Artificial Intelligence Research*, 37, 141-188. doi:10.1613/jair.2934
- U.S. Census Bureau. (2021a). North American industry classification system. <https://www.census.gov/naics/>. Accessed September 30, 2021.
- U.S. Census Bureau. (2021b). Economic Census. <https://www.census.gov/programs-surveys/economic-census.html>. Accessed September 30, 2021.
- Wan, C., Wang, Y., Liu, Y., Ji, J., and Feng, G. (2019). Composite Feature Extraction and Selection for Text Classification. *IEEE Access*, 7, 35208-35219. doi:10.1109/ACCESS.2019.2904602

- Wang, Y., Zhou, Z., Jin, S., Liu, D., and Lu, M. (2017). Comparisons and selections of features and classifiers for short text classification. *IOP Conf. Series: Materials Science and Engineering*, 261, 012018.
- Worsham, J.M. (2018). *Towards literary genre identification: applied neural networks for large text classification*. M.S. thesis. University of Colorado at Colorado Springs.
- Xu, J., Wang, P., Guanhua, T., Xu, B., Zhao, J., Wang, F., and Hao, H. (2015). Short text clustering via convolutional neural networks. *NAACL 2015 Vector Space Modeling for NLP Workshop*, 62-69.
- Xu, J., Wang, P., Guanhua, T., Xu, B., Zhao, J., Wang, F., and Hao, H. (2017). Short text dataset for classification and clustering extracted from StackOverflow. <https://github.com/jacoxu/StackOverflow>. Accessed September 30, 2021.