Contextual Feature Weighting Using Knowledge beyond the Repository Knowledge

Kazem Qazanfari*, Abdou Youssef
The George Washington University, Washington, DC, USA.

* Corresponding author. Tel.: +1 703 981 4640; email: kazemmit@gwu.edu
Manuscript submitted January 7, 2018; accepted April 1, 2018.
doi: 10.17706/ijcce.2018.7.3.45-57

Abstract: Bag of words, bigram, or more complex combinations of words are the most among general and widely used features in text classification. However, in almost all real-world text classification problems, the distribution of the available training dataset for each class often does not match the real distribution of the class concept, which reduces the accuracy of the classifiers. Let $W(f)$ and $R(f)$ be the discriminating power of feature f based on the world knowledge and the repository knowledge, respectively. In an ideal situation, $W(f) = R(f)$ is desirable; however, in most situations, $W(f)$ and $R(f)$ are not equal and sometimes they are quite different, because the repository knowledge and the world knowledge do not have the same statistics about the discriminating power of feature f. In this paper, this phenomenon is called *inadequacy of knowledge* and we show how this phenomenon could reduce the performance of the text classifiers. To solve this issue, a novel feature weighting method is proposed which combines two bodies of knowledge, world knowledge and repository knowledge, using a particular transformation T. In this method, if both the world knowledge and the repository knowledge indicate a significantly high (resp., low) discriminating power of feature f, the weight of this feature is increased (resp., decreased); otherwise, the weight of the feature will be determined by a linear combination of the two weights. Experimental results show that the performance of classifiers like SVM, KNN and Bayes improves significantly if the proposed feature weighting method is applied on the contextual features such as bigram and unigram. It is shown also that pruning some words from the dataset using the proposed feature weighting method could improve the performance of the text classifier when the feature sets are created using Doc2vec.

Key words: Feature weighting, feature extraction, text classification, transfer learning.

1. Introduction

Text classification is a field of study in data mining, web mining, and text mining. The task is to assign a document or a piece of text to one or more classes. Recently, text classification techniques have been applied widely to a variety of areas like personality insights, social networks, news and politics, economics, target marketing, recommender systems, and medical diagnosis. Therefore, having a highly accurate text classification system is highly useful.

The problem of text classification is defined as follows. Given a training set D of documents, $D = \{D_1, D_2, \dots, D_N\}$, such that each document is labeled with a class value drawn from a set of K different values $\{C_1, C_2, \dots, C_K\}$, a document classification model is trained, which relates each document to one of the class labels. Once the classification model is trained, it is used to predict the class label of a new document or a new piece of text.

Like in other pattern recognition applications, the design of text classifiers relies on feature extraction. Extracted features should preserve as much of the original document information as possible, while keeping the time and space complexity of the extraction process reasonable. Different features for representing a document have been proposed. Contextual features like unigram and bigram [1], [2] are among the simplest, and are applied in a variety of text mining applications. Other types of features, such as conceptual features [3], as well as document-structure features and statistical features like total number of words, number of sentences, average length of sentences [4], are proposed for some special applications.

Many document classification techniques which use such kinds of features have been developed for text classification. These include probabilistic models under the naïve Bayes framework [5], [6], SVM (with reports of it being one of the currently most accurate techniques for text classification [7]-[9]), Expectation Maximization (EM) [10], KNN [11], [12], Artificial neural network [13]-[17], and decision trees [18].

As mentioned earlier, in text mining, the most general and widely used features are unigram, bigram, or more complex combinations of words, described as follows. Let $T_1, T_2, \dots, T_n$ denote distinct terms (unigrams, bigrams, ...) used for indexing documents $D_1, D_2, \dots, D_m$ of a problem P. Document D_i is represented by a term vector defined as:

$$D_i = \left(a_1^i, a_2^i, \dots, a_n^i\right)^T \tag{1}$$

where a_j^i is a weight of the term T_j in the document D_i. The values a_j^i can be just simple frequencies of the term T_j in the document D_i, either normalized or unnormalized. For classification purposes, this document model is extended so a document is represented as a tuple:

$$D_i' = < D_i, C_k > \tag{2}$$

where D_i is the just defined document vector and, C_k, for $k = 1,2 \dots, K$, is the document class. Since these terms would be used as the feature set for creating the classifier model, the discriminating power of each feature might change the performance of the final document classifier.

However, in almost all real-world text classification problems, the training sets do not fully and accurately capture the characteristics of the problem. In particular, the distribution of the available training data for each class often does not match the real distribution of the class concept. Consider w_i to be a word that appears in the training dataset. Let $WK(w_i, \mathrm{c})$ and $\mathrm{Rep}(w_i, \mathrm{c})$ be the relevance of word w_i to the concept of class c based on the world knowledge and repository (training dataset) knowledge, respectively. In an ideal situation, it is desirable to have $WK(w_i, \mathrm{c}) = \mathrm{Rep}(w_i, \mathrm{c})$ or at least $WK(w_i, \mathrm{c}) \cong \mathrm{Rep}(w_i, \mathrm{c})$; however, in most situations, $WK(w_i, \mathrm{c})$ differ significantly from $\mathrm{Rep}(w_i, \mathrm{c})$, because the repository knowledge and world knowledge do not have the same judgement about the relevance of word w_i to class c concept.

Given a repository, define the following term:

$$P_{C_k}^{w_i} = \frac{f_{w_i, C_k}}{N_{C_k}} \tag{3}$$

where N_{C_k} is the number of documents in class C_k, and f_{w_i, C_k} is the number of documents (of class C_k) that have word w_i. Basically, $P_{C_k}^{w_i}$ is the probability of word w_i in class C_k and it could be considered as modeling the repository knowledge about word w_i when the bag of words is used as the feature set.

To illustrate the use of this probability, consider Fig. 1. This figure shows $P_{C_k}^{w_i}$ for 9 words of 20-Newsgroups dataset for two classes: “comp.graphics” and “rec.autos”. Among these 9 words, it is clear that some words such as “paint” and “design” are more related to the “comp.graphics” class, whereas certain other words such as “device” and “machine” are more related to the “rec.autos” class; however, Fig. 1 shows that, based on the repository knowledge, these words figure more prominently in the opposite class. There

might be also some other words, like “across”, “bottle” and “complex” which clearly belong to neither of these two classes in the world knowledge, but, based on the repository knowledge, there is a strong connection between these words and one of the classes. Therefore, by relying on the repository knowledge alone, some words like those mentioned above, might degrade the accuracy of the classifier. In this paper, this phenomenon is called *inadequacy of knowledge*. Needless to say, the same phenomenon occurs when bigrams or more complicated combinations are used as features.

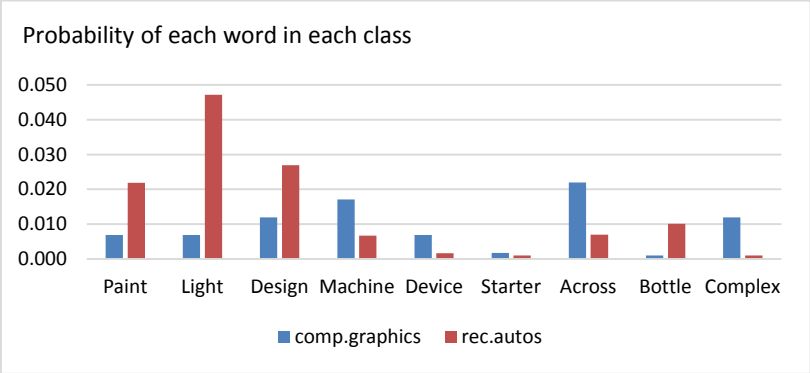


Fig. 1. Relevance of some words to two different classes in the repository.

Also, consider the list of distinctive words of classes. The ones which fall in the same class have related meanings. Using this fact, we are going to increase the knowledge of the classifier about the repository dataset. Although the similarity of two words could be calculated using the repository dataset, this knowledge – the knowledge of similarity of two words using repository dataset - is already considered by the classifier and it will not provide additional information about the words. Therefore, an external –and superior- source of knowledge about the similarity of words is needed. The details of this external source of knowledge will be introduced later.

Finally, to address the issue of *inadequacy of knowledge*, two bodies of knowledge are considered at the same time: World knowledge, which is an external source of knowledge that carries the general meanings of the words, and the repository knowledge, which normally has the domain-confined meaning of each word for a given problem P. To do so, a special transform T on these two bodies of knowledge is proposed, which combines the contributions of both bodies of knowledge. After applying the transform, some feature weighing functions will be introduced to assign a different weight to each feature. In the next section, the details of the proposed method will be presented and discussed.

An important question that comes to mind is that since the world knowledge is presumably superior to the repository knowledge, why is the classifier not trained on the world knowledge directly? The reason is twofold:

1. The repository knowledge normally has valuable domain-knowledge that is often largely masked by the (much bigger, generic) world knowledge, and thus its domain-specific contributions should not be wasted.

2. The world knowledge may not be actually available. Instead, information derived (by some other entity) from the world knowledge is what is available. Our approach is to use that derived information, not the raw text in the world knowledge.

The rest of this paper is organized as follows. The proposed method for improving the knowledge of the classifier for each term (feature) will be explained in Section 2. In Section 3, we present experimental results on two well-known datasets, that indicate our feature weighting method can significantly improve the performance of text classifiers. The paper concludes in Section 4 with a discussion of the results

achieved and some suggestions for future directions.

2. The Proposed Method

If a body of knowledge B is viewed as world knowledge, we denote it as U, and if it is the repository (training dataset), we denote it by R. Also, given a problem P and an N-gram g (as a feature), denote by $W_R(g)$ the weight assigned to the feature g by a weight function W_R based on R in the domain of problem P. Basically, $W_R(g)$ models the knowledge of R about the N-gram g in the domain of problem P. For example, W_R could be *Tf-idf*. Also, we view $W_U(g)$ as a weigh assigned to the N-gram g by a weight function W_U based on world knowledge U and regardless of problem P.

In this section, a method will be proposed which combines $W_R(g)$ and $W_U(g)$ to create $W_{RU}(g)$ for each N-gram g. The resulting $W_{RU}(g)$ models the discriminating contribution of N-gram g (as a feature) to the classification problem P based on both the world knowledge and the repository knowledge. To do so, a graph representation is used to visualize how $W_R(g)$ could use U (world knowledge) to create $W_{RU}(g)$. Also, to facilitate the understanding of our method, N is considered to be 2 (bigram).

Goal: The goal is to derive a graph $G_{RU}(V, E)$, where the nodes in V are the words in the repository, and each edge (w_i, w_j) between two words w_i and w_j is given a weight $W_{RU}(g = w_i w_j)$. These weights could later be used for feature reduction, or feature weighting in the training phase of the classifiers.

Input: There are two input graphs.

The first is $G_U(V, E)$, which is purely based on the world knowledge regardless of problem P. Each edge $e = (w_i, w_j)$ of $G_U(V, E)$ is assigned a weight $W_U(w_i w_j)$, a value representing the similarity between the two words w_i and w_j based on world knowledge. Since there is no standard comprehensive similarity metric between two words at this time, we will use Google's pre-trained model [19], which includes word vectors for a vocabulary of 3 million words and phrases that they trained on roughly 100 billion words from a Google News dataset.

The second graph is $G_R(V, E)$ which is purely based on the repository knowledge. Each edge $e = (w_i, w_j)$ of $G_R(V, E)$ is assigned a value $W_R(e)$ representing the similarity between two words w_i, w_j relatively to the repository:

$$W_R(e) = \text{var}\left(\left\{P_{C_1}^{w_i w_j}, P_{C_2}^{w_i w_j}, \dots, P_{C_K}^{w_i w_j}\right\}\right) \tag{4}$$

$$P_{C_k}^{w_i w_j} = \frac{f_{C_k}^{w_i w_j}}{N_{C_k}} \tag{5}$$

where N_{C_k} is the number of documents in class C_k, and $f_{C_k}^{w_i w_j}$ is the number of documents (of class C_k) that have both words w_i and w_j. Finally, $W_R(e)$ is the variance of all $P_{C_k}^{w_i w_j}$ for $k = 1,2 \dots, K$.

Note that all the three graphs $G_{RU}(V, E), G_R(V, E)$ and $G_U(V, E)$ are fully connected graphs with the same nodes.

Transform: A transform function T is needed to merge the problem P knowledge graph (i.e., repository knowledge graph, $G_R(V, E)$) with the world knowledge graph $G_U(V, E)$:

$$G_R(V, E) + G_U(V, E) \xrightarrow{T} G_{RU}(V, E) \tag{6}$$

$$\left(W_R(g), W_U(g)\right) \rightarrow W_{RU}(g) \tag{7}$$

where the details of the transform, specifically the value of $W_{RU}(g)$, will be provided later in this section. To illustrate visually the desired effect of the transform, consider an example of a classification problem P with two classes C_1 and C_2. After some preprocessing (like tokenizing, stopword removal, etc.) on the repository

samples, seven words have remained: $V = \{w_1, w_2, w_3, w_4, w_5, w_6, w_7\}$. In a very simple scenario, suppose that based on $W_{RU}(g), V_1 = \{w_1, w_2, w_3\}$ are the words related to C_1 and $V_2 = \{w_4, w_5, w_6, w_7\}$ are the words related to the C_2 class. The graph $G_R(V, E)$ is shown in Fig. 2. In this figure, the thickness of every edge (w_i, w_j) is based on the weight $W_R(w_i w_j)$: the higher the weight, the thicker the edge. Observe in Fig. 2 that there are some low-weight edges between some pairs of words of the same class, such as (w_5, w_6), and that there are some high-weight edges between some words from different classes, such as (w_1, w_7). Such phenomena occur due to the *inadequacy of knowledge* (or, more accurately, inadequate representation) of R about problem P.

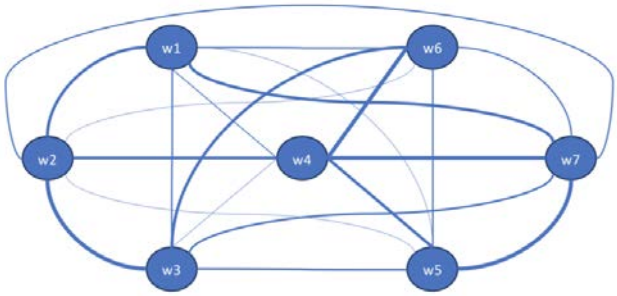


Fig. 2. A typical graph $G_R(V, E)$.

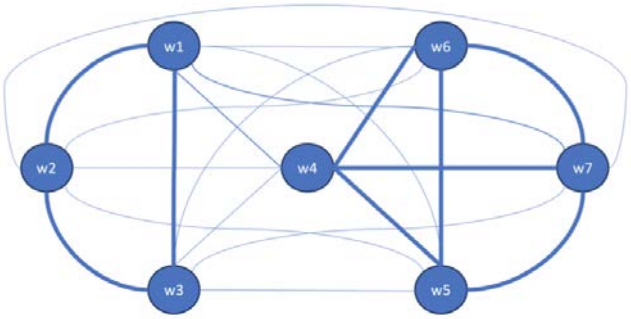


Fig. 3. A typical graph $G_{RU}(V, E)$ after transformation T.

By applying a good graph transform T on $G_R(V, E)$ and $G_U(V, E)$, the resulting graph $G_{RU}(V, E)$ will be like the graph in Fig. 3: the edge weights between the words of the same class are high, while the edge weights between the words from different classes are low.

Before detailing the graph transform, some terms will be defined first, relative to a body of knowledge B, where B can be the world knowledge U or the repository knowledge R. Consider an edge $e_k = (w_i, w_j)$ between two words w_i and w_j:

$$W_B^{norm}(e_k) = \frac{W_B(e_k)}{\max(\text{all } W_B(e_l) | \, e_l \in E)} \tag{8}$$

The term $W_B^{norm}(e_k)$ is the normalized similarity weight between two words w_i and w_j relative to the body of knowledge B.

$$\text{EquiSim}_B(e_k, w) = \frac{\min\{W_B^{norm}(w_i, w), \quad W_B^{norm}(w_j, w)\}}{\max\{W_B^{norm}(w_i, w), \quad W_B^{norm}(w_j, w)\}} \tag{9}$$

which is meant to represent the extent of equidistance (or rather equi-similarity) between a word w and the two words w_i and w_j, relative to the body of knowledge B. Observe that the more equal the similarity

between w and w_i is to the similarity between w and w_j, the greater the value of $\text{EquiSim}_B(e_k, w)$. Note also that $\text{EquiSim}_B(e_k, w)$ is always between 0 and 1.

$$W_B^{ANE}(e_k) = \sum_{w \in \text{neighborhood}_B(\{w_i, w_j\})} \text{EquiSim}_B(e_k, w) \tag{10}$$

The term W_B^{ANE} is the overall (i.e., cumulative) equi-similarity to the two words w_i and w_j from all the words in the neighborhood of $\{w_i, w_j\}$ in the body of knowledge B. Since the graph of B is fully connected, the neighborhood of $\{w_i, w_j\}$ is $B - \{w_i, w_j\}$. Because the weight of the majority of edges in $G_R(V, E)$ and $G_U(V, E)$ are close to zero, and to reduce the computation time, we reduce the neighborhood of $\{w_i, w_j\}$ to the top 20% of the most highly weighted edges incident to the two words $\{w_i, w_j\}$.

$$W_B^{NCE}(e_k) = \frac{W_B^{ANE}(e_k)}{\max(\text{all } W_B^{ANE}(e_l) \mid e_l \in E)} \tag{11}$$

which is the normalized cumulative equi-similarity (NCE), so that it is always between 0 and 1.

Using the above definitions, the graph transform procedure is defined next. In this algorithm, γ_1 and γ_2 are two thresholding parameters that should be experimentally optimized. The optimized values might be different for different repositories.

Algorithm 1: Graph transform T

Result: $W_{RU}(e_k)$
initialization:

- creating of $G_R(V, E)$ and $G_U(V, E)$
- calculating of $W_U^{norm}(e_k)$, $W_R^{norm}(e_k)$, $W_U^{NCE}(e_k)$ and $W_R^{NCE}(e_k)$

foreach $e_k = (w_i, w_j) \in G_{RU}(V, E)$ **do**
  $S_U = W_U^{norm}(e_k)$; *// normalized similarity between the two words, w.r.t. U*
  $S_R = W_R^{norm}(e_k)$; *// normalized similarity between the two words, w.r.t. R*
  $ES_U = W_U^{NCE}(e_k)$; *// normalized cumulative equi-similarity to the two words, w.r.t. U*
  $ES_R = W_R^{NCE}(e_k)$; *// normalized cumulative equi-similarity to the two words, w.r.t. R*
  if $(S_U < \gamma_1$ *and* $S_R < \gamma_1) or (S_U > \gamma_2$ *and* $S_R > \gamma_2)$ **then**
    $W_{RU}(e_k) = \frac{(W_U(e_k) + W_R(e_k))}{2}$;
  else
    $M_U = max(S_U, ES_U)$;
    $M_R = max(S_R, ES_R)$;
    if $(M_U > M_R)$ **then**
      $W_{RU}(e_k) = \alpha * W_U(e_k) + \beta * W_R(e_k)$;
    else
      $W_{RU}(e_k) = \beta * W_U(e_k) + \alpha * W_R(e_k)$;
    end
    //where $\alpha > \beta$ *and* $\alpha + \beta = 1$
  end
end

Based on the first condition of this algorithm, i.e. $(S_U < \gamma_1 \text{and } S_R < \gamma_1) \text{or} (S_U > \gamma_2 \text{and } S_R > \gamma_2)$, if both world knowledge and repository knowledge indicate a significantly high or low similarity between two words $\{w_i, w_j\}$, then the mean average of $W_U(e_k)$ and $W_R(e_k)$ will be assigned to $W_{RU}(e_k)$ which indicates that both sources of knowledge have the same amount of effect on $W_{RU}(e_k)$. However, if this condition is not satisfied, then the similarity between the two words based on their neighborhood is also taken into consideration to find the maximum similarity between two words, by considering the words themselves and their neighborhood. To do so, first, the maximum normalized similarity weight between weights $W_B^{\text{norm}}(e_k)$ and $W_B^{\text{NCE}}(e_k)$ relative to the body of knowledge B is selected. Then, a weighted combination of the weights $W_B(e_k)$'s is assigned to $W_{RU}(e_k)$ which combines the contributions of both bodies of knowledge U and R.

3. Experiments and Discussion

3.1. Datasets

In our experiment, we use two corpora: Reuters [20] and 20 Newsgroups [21] data sets. The 20 Newsgroups data set is a collection of about 20,000 newsgroup documents, divided across 20 different newsgroups in variety of topics such as computer, religion and politics. The Reuters-21578 dataset contains documents collected from the Reuters newswire in 1987. It is a standard text categorization benchmark and contains 21,578 samples in 135 categories. As a preprocessing step on the Reuters dataset, all categories that have less than 100 documents in the training set and the test set have been removed. The remaining dataset has 20 categories with a training set of 5,887 documents and a test set of 2,323 documents.

3.2. World Knowledge

Since there is no standard comprehensive similarity metric between two words at this time, we will use Google's pre-trained model [19], which includes word vectors for a vocabulary of 3 million words and phrases that they trained on roughly 100 billion words from a Google News dataset. This model has been provided using Gensim, which is a mature well-known open-source vector space modeling and topic modeling toolkit implemented in Python.

3.3. Feature Weighting

In this section, a feature weighting method will be introduced for when unigrams and bigrams are used as the feature set. However, it is also possible to weight any N-grams, e.g. trigrams.

Weighting of one-gram features: the weight $\mathcal{W}(w_i)$of each word w_i, could be calculated using the following formulas:

$$\mathcal{W}(w_i) = \frac{N(w_i)}{\max\{N(w_j) \,|\, w_j \in \text{ bag of words }\}} \tag{12}$$

$$N(w_j) = \sum_{w_k \in \text{ bag of words}} W_{RU}(w_j, w_k) \tag{13}$$

Also, the weight $\mathcal{W}(b_{ij})$of each bigram b_{ij} could be calculated using the following formula:

$$\mathcal{W}(b_{ij}) = \frac{W_{RU}(w_i, w_j)}{\max\{W_{RU}(w_k, w_l) \,|\, (w_k, w_l) \in E\}} \tag{14}$$

3.4. Experiments

The first experiment shows how the proposed method of using the world knowledge could improve the

repository knowledge about the words which were mentioned in Section 2. The initial knowledge of repository about the word w_i could be defined as following:

$$\mathcal{W}_{init}(w_i) = \frac{N_R(w_i)}{\max\{N_R(w_j) \mid w_j \in \text{ bag of words}\}} \tag{15}$$

$$N_R(w_j) = \sum_{w_k \in \text{ bag of words}} W_R(w_j, w_k) \tag{16}$$

Based on the above definitions, $\mathcal{W}_{init}(w_i)$ shows the discriminating power of word w_i based on repository knowledge. This weight is higher as the similarity (i.e. co-occurrences in the same text) of the word w_i and its neighbor words is higher. Also, $\mathcal{W}(w_i)$ is the discriminating power of word w_i based on both world and repository knowledge. Fig. 4 shows the discriminating power of the words that were mentioned in Section 2. As it is shown in this figure, although some words like "paint", "design", "starter" and "machine" (consider them as group A) have highly conceptual correlation to either "comp.graphics" class or "rec.autos" class, based on the repository knowledge they have the same discriminating power as other words like "across", "bottle" and "complex" (consider them as group B). However, it is shown in this figure that by considering both the repository and the world knowledge, the discriminating power of the words in group A has been increased, while it has been reduced for the words in group B.

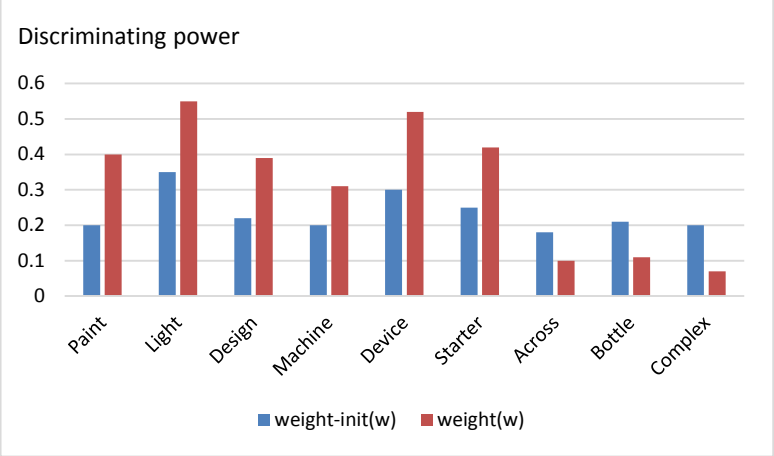


Fig. 4. The discriminating power of some words before the transform $\mathcal{W}_{init}(w_i)$ (weight-init(w)) and after the transform $\mathcal{W}(w_i)$ (weight(w)).

As mentioned before, the proposed method has four parameters, namely, $\gamma_1, \gamma_2, \alpha$ and β. Since γ_1 and γ_2 could be optimized separately from α and β, γ_1 and γ_2 are experimentally optimized first, and then α and β, using the 20-Newsgroups and Reuters datasets separately, and based on the F-measure as the performance metric.

To train the classifiers, a weighted feature set of unigrams and bigrams were used. Also, the performance of the different values for these parameters were evaluated using the F-measure, and assumed the SVM classifier (due to its often-superior performance).

Also, the experimental results showed that to have optimum F-measure, the optimized values for $\gamma_1, \gamma_2, \alpha$ and β parameters are different based on the selected repository. Table 1 shows the optimized values of these parameters based on the F-measure on Reuters [20] and 20-Newsgroups [21] data sets.

In the next experiment, the performance of the proposed feature weighting method will be evaluated on several widely-used text classification algorithms on the mentioned datasets. Specifically, SVM, KNN and Bayes classifiers are trained using the same feature set, i.e., the tf-idf of unigrams and bigrams. However,

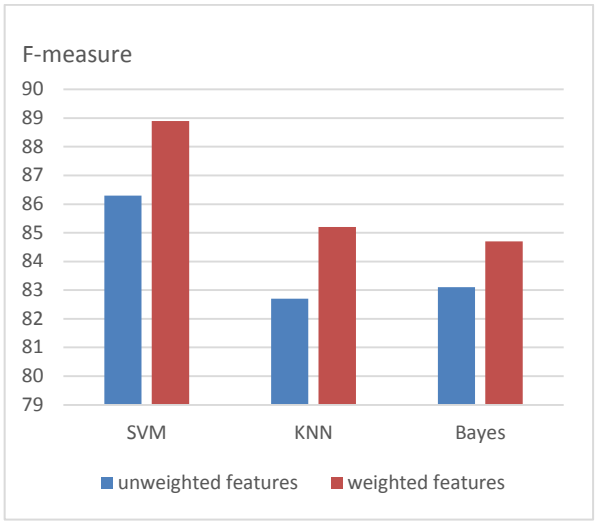
when our feature weighting method is not used, the feature set type is considered to be unweighted, and when our feature weighting method is used, the feature set type is considered to be weighted. Table 2 and Fig. 5 show the performance of these classifiers using different types of feature sets as mentioned above. As it is shown in this table, the performance of the mentioned classifiers has been improved after using the weighted feature sets.

Table 1. Optimized Values of the $\gamma_1, \gamma_2, \alpha$ and β Parameters Based on F-Measure on Reuters and 20-Newsgroups Datasets

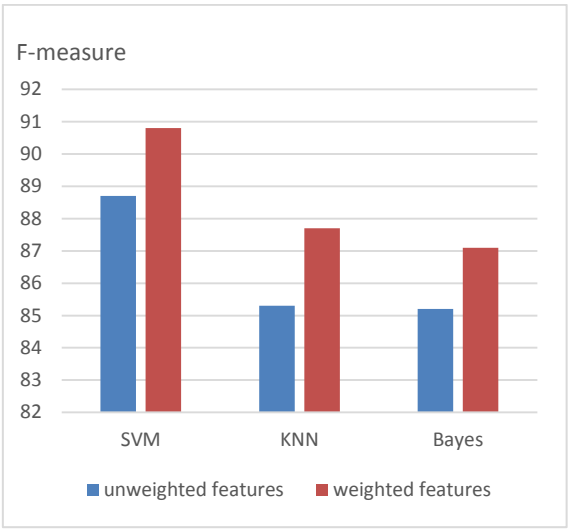
Dataset	γ_1	γ_2	α	β
Reuters	0.08	0.91	0.61	0.39
20-Newsgroups	0.09	0.89	0.65	0.35

Table 2. Performance Evaluation of Some Classification Algorithms on Reuters and 20-Newsgroups Datasets Using Different Feature Set Type

Feature set + Algorithm	Feature set type	F-measure	
		Reuters	20-Newsgroups
(Unigram,bigram)+SVM	unweighted	86.3	88.7
(Unigram,bigram)+SVM	weighted	89	90.6
(Unigram,bigram)+KNN	unweighted	82.7	85.3
(Unigram,bigram)+KNN	weighted	85.6	87.9
(Unigram,bigram)+Bayes	unweighted	83.1	85.2
(Unigram,bigram)+Bayes	weighted	84.5	87.3



(a) 20-Newsgroups dataset



(b) Reuters dataset

Fig. 5. Performance evaluation of some recently classification algorithms on 20-Newsgroups (a) and Reuters (b) datasets using different feature set type (unweighted means our method is not used and weighted means our method is used).

Also, to evaluate the performance of the proposed method on some Neural Network methods, the two following well-known solutions are considered:

- Using Doc2vec [22] to generate the feature vectors and train an optional classifier like SVM.

Doc2vec is an unsupervised algorithm to generate feature sets for sentences/documents using Neural Network. In this paper, a pre-trained Doc2vec model [23] which has been trained on the English Wikipedia was used to generate the feature set of each textual sample.

- Using Convolutional Neural Networks (CNN) [24] to generate the classifier model: CNN is a type of feed-forward artificial neural network. It uses a variation of multilayer perceptron designed to require minimal preprocessing, and has shown good performance in different applications.

Since the unigram and bigram features are not used for two above solutions, our feature weighting method cannot be used for these solutions. Instead, the following pruning process would be applied on the dataset before the training phase of the above solutions:

"Removing all words from the dataset whose

$\mathcal{W}(w_i)$ < threshold T."

Table 3 shows the performance of these two methods using different types of input training dataset, i.e. original dataset or pruned dataset. Looking at Table 3, one can observe the following:

- The performance of the SVM classifier when Doc2vec is used for creating the feature set has been improved after applying the aforementioned pruning process on the dataset. Note also that the highest F-measure is gained when T=0.11 and as the threshold T is increased above 0.11, the F-measure drops.
- The performance of the CNN is not improved. This experiment shows that as the threshold value is increased, the F-measure drops quickly. The reason that our proposed method could not improve CNN is, the input of CNN is a set of features which come from some other pre-trained world knowledge models like word2vec. On the other words, CNN is trained using both the world knowledge and the repository knowledge. So, purely removing some under-weighted words (using the mentioned pruning process), as in our method cannot improve it.

Table 3. Performance Evaluation of Some Recently Classification Algorithms on Reuters and 20-Newsgroups Datasets Using Original Dataset and Pre-processed Dataset

Dataset type + Method	F-measure	
	Reuters	20-Newsgroups
Original dataset + (Doc2vec, SVM)	87.3	90
Pruned dataset (T=0.05) + (Doc2vec, SVM)	89.6	91.6
Pruned dataset (T=0.11) + (Doc2vec, SVM)	90.4	92.3
Pruneddataset (T=0.15) + (Doc2vec, SVM)	89.7	91.4
Original dataset + (CNN)	91.2	93.6
Pruneddataset (T=0.05) + (CNN)	90.4	92.3
Pruneddataset (T=0.10) + (CNN)	88.7	90.5
Pruneddataset (T=0.15) + (CNN)	86.5	88.2

4. Conclusion and Future Work

In almost all real-world text classification problems, the training sets are not comprehensive. Therefore, the classifier models are built using incomplete information, which could reduce their performance. In this paper, to solve this issue, a novel feature weighting method was introduced.

To do so, two bodies of knowledge, world knowledge and repository knowledge, have been combined using a special transform T that was introduced and optimized. If both the world knowledge and the repository knowledge indicate a significantly high/low correlation between a feature and a class, the weight

of the feature is increased/decreased, however if the two bodies of knowledge don't agree, then the weight of the feature will be determined by a linear combination of the two feature weights from the two bodies of knowledge; these values are an indicator of the discriminating power of the feature using the two bodies of knowledge.

The performance of the feature weighing method has been evaluated on some widely-used classification algorithms, namely, SVM, KNN and Bayes classifiers. These classifiers were trained using the same feature set; i.e. bigram and unigram in tf-idf manner. The evaluation results showed that the performance of these classifiers has been improved using our feature weighting method. In a separate experiment, the SVM classifier was also trained using Doc2vec features. This experiment showed that by pruning some unrelated words from the dataset using our feature weighting method, the performance of the classifier could be improved too. In another experiment, when our method was applied to CNN, the results showed that performance of the CNN is not improved. The reason is that the input of CNN is a set of features which are derived from some other pre-trained world knowledge models like word2vec. Therefore, the mentioned method of using the world knowledge could not improve the performance of CNN.

As part of our future research, we will explore alternative ways of using the world knowledge to improve the performance of the CNN classifiers. Also, we will test the improvement in performance when using N-grams for large values of N beyond bigrams.

Also, to generalize our proposed method for using different sources of knowledge, in the future research work, we will focus on transfer learning or inductive transfer to use the stored knowledge while solving one problem and applying it to a different but related problem conclusion section is not required.

References

[1] Fontaine, M., & Matwin, S. (2000). Features extraction techniques of unintelligible texts. *Proceedings of KDD's Workshop on Text Mining*. Boston.

[2] Qazanfari, K., Youssef, A., Keane, K., & Nelson, J. (2017). A novel recommendation system to match college events and groups to students. *Proceedings of AIAAT 2017* (pp. 1-15). Hawaii, USA.

[3] Jensen, L. S., & Martinez, T. (2000). Improving text classification by using conceptual and contextual features. *Proceedings of Workshop on Text Mining at the ACM Sigkdd International Conference on Knowledge Discovery and Data Mining* (pp. 101-102).

[4] Vel, O. D. (2000). Mining e-mail authorship. *Proceedings of Workshop on Text Mining, ACM International Conference on Knowledge Discovery and Data Mining*.

[5] Diab, D. M., & Hindi, K. M. E. (2017). Using differential evolution for fine tuning nave Bayesian classifiers and its application for text classification. *Applied Soft Computing, 54*, 183-199.

[6] Tang, B., He, H., Baggenstoss, P. M., & Kay, S. (2016). A Bayesian classification approach using class-specific features for text categorization. *IEEE TKDE, 28*, 1602–1606.

[7] Liu, B. (2011). *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data.* Springer Science & Business Media, Springer.

[8] Jain, A., & Mandowara, J. (2016). Classification by combining text classifiers to improve the efficiency of classification. *Int. J. Comput. Appl.*, 2250-1797.

[9] Haddoud, M., Lecroq, A., Lecroq, T., & Abdeddam, S. (2016). Combining supervised term-weighting metrics for SVM text classification with extended term representation. *Knowl. Inf. Syst., 49(3)*, 909-931.

[10] Zhao, L., Huang, M., Yao, Z., Su, R., Jiang, Y., & Zhu, X. (2016). Semi-supervised multinomial naive Bayes for text classification by leveraging word-level statistical constraint. *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence* (pp. 2877-2884).

[11] Song, J., Huang, X., Qin, S., & Song, Q. (2016). A bi-directional sampling based on K-means method for

imbalance text classification. *Proceedings of IEEE/ACIS International Conference on Computer & Information Science* (pp. 1-5).

[12] Bijalwan, V., Kumar, V., Kumari, P., & Pascual, J. (2014). KNN based machine learning approach for text and document mining. *Int. J. Database Theor. and App., 7(1)*, 61–70.

[13] Kowsari, K., Brown, D. E., Heidarysafa, M., Meimandi, K. J., Gerber, M. S., & Barne, L. E. (2017). HDLTex: Hierarchical deep learning for text classification. *Proceedings of the 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*.

[14] Nam, J., Kim, J., Menca, E. L., Gurevych, I., & Frnkranz, J. (2014). Large-scale multi-label text classification- revisiting neural networks. *Proceedings of the 2014 European Conference on Machine Learning and Knowledge Discovery in Databases* (pp. 437-452).

[15] Lai, S., Xu, L., Liu, K., & Zhao, J. (2015). Recurrent convolutional neural networks for text classification. *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence* (pp. 2267-2273).

[16] Zhang, X., Zhao, J., & LeCun, Y. (2015). Character-level convolutional networks for text classification. *Adv. Neur. In.*, 649-657.

[17] Tang, D., Qin, B., & Liu, T. (2015). Document modeling with gated recurrent neural network for sentiment classification. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)* (pp. 1422-1432).

[18] Colace, F., De Santo, M., Greco, L., & Napoletano, P. (2014). Text classification using a few labeled examples. *Comput. Hum. Behav., 30*, 689–697.

[19] Google's Pre-trained Model. Retrieved from the website: https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit

[20] Lewis, D. D. (1997). Reuters-21578 text categorization test collection. *Distribution 1.0, AT&T Labs-Research*.

[21] Joachims, T. (1997). A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. *Proceedings of International Conference on Machine Learning* (pp. 143-151).

[22] Quoc, L. Mikolov, T. (2014). Distributed representations of sentences and documents. *Proceedings of the 31st International Conference on Machine Learning*.

[23] Pre-trained Doc2vec Model. Retrieved from the website: https://ibm.ent.box.com/s/3f160t4xpuya9an935k84ig465gvy mm2

[24] Yoon, K. (2014). Convolutional neural networks for sentence classification. *Proceedings of Conference on Empirical Methods in Natural Language Processing*.

Kazem Qazanfari received his BSc degree from Birjand University in 2008 (computer software engineering) and the MSc degree from Amirkabir University of Technology (artificial intelligence) in 2010 from Iran, Tehran. He is currently a PhD candidate at the George Washington University, Washington, DC, USA and majored computer science.

He has served as a researcher at some research centers including Amirkabir University of Technology, Sharif University of Technology, Iran Telecom Research Center and Tejarat Bank before starting his PhD studies. During his PhD studies, he joined to Promantus Inc. as a data scientist located in Washington DC, USA. His current research includes data mining, text mining, machine learning, deep learning, deep neural networks, convolutional deep neural networks, deep belief networks and recurrent neural networks. His research in Promantus Inc. company is to use NLP and deep learning to create a novel event recommendation systems for college students. Mr. Qazanfari was awarded a full package scholarship for 4 consecutive year.

Abdou Youssef received his MA and PhD degrees in computer science from Princeton University, Princeton, NJ, in 1985 and 1988, respectively, and his BS in mathematics from the Lebanese University in 1981. He also completed the requirements for a BS degree in statistics in 1992 at the Lebanese University.

He has 30 years of research and teaching experience in the field of computer science. He is currently a tenured professor and former chairman of the Department of Computer Science at The George Washington University, Washington, D.C, which he joined as assistant professor in fall of 1987. His current research interests are applied data science, math search and math language processing, audio-visual data processing, pattern recognition, theory and algorithms.

Dr. Youssef has published over 125 papers in those areas, and co-edited the book *Interconnection Networks for High-Performance Parallel Computers*, published by IEEE Computer Society Press in 1994. His research has been funded by NSF, NSA, and NIST. He has developed applied techniques and systems that have been put to critical use. In the late 1990's, he and his students developed for the US Government a system that recovers from fax errors without retransmission. More recently, he has created for the US National Institute of Standards and Technology (NIST) a math-search engine as part of the Digital Library of Mathematical Functions (DLMF).