Fast Text Classification using Lean Gradient Descent Feed Forward Neural Network for Category Feature Augmentation

Joseph Attieh
E.C.E Dept., School of Engineering
Lebanese American University (LAU)
36 Byblos, Lebanon
joseph.attieh@lau.edu

Joe Tekli
E.C.E Dept., School of Engineering
Lebanese American University (LAU)
36 Byblos, Lebanon
joe.tekli@lau.edu.lb

***Abstract*—Text classification is a key task of the Natural Language Processing (NLP) field that aims at assigning predefined categories to textual documents. Performing text classification requires features that effectively represent the content and the meaning of textual documents. Selecting a suitable method for term weighting is of central importance and can improve the quality of the classification method. In this paper, we propose to a new text classification solution to perform Category-based Feature Augmentation (CFA) on the document representation. First, a term-category feature matrix is derived from a modified version of the supervised Term-Frequency Inverse-Category-Frequency (TF-ICF) weighting model. This is done by embedding the TF-ICF matrix in a one-layer feed-forward neural network. The latter is trained using the gradient descent algorithm allowing to iteratively update the term-category matrix until reaching convergence. The model produces category-based feature vector representations that are used to augment the document representations and perform the classification task. Experimental results on four benchmark datasets show that our lean model approach improves text classification accuracy and is significantly more efficient compared with its deep model alternatives.**

***Keywords*—Text Classification, Document and Text Processing, Feature Engineering, Supervised Term Weighting, Inverse Category Frequency, TF-IDF, Text Representation.**

I. INTRODUCTION

Text classification has become a key task in the NLP field [2, 28], with applications ranging over different domains, including information retrieval (e.g., classifying customer complaints into predefined categories [15, 32]), information filtering (e.g., identifying spam emails and filtering them for better user experience [3, 19]), and sentiment analysis (e.g., classifying texts into different polarities, e.g., *positive*, *negative*, or affective categories, e.g., *happy*, *angry*, *sad* [12, 13]). Text classification consists of two main phases: i) feature representation phase, and ii) classification phase. State-of-the-art feature representations mainly compute a weighted representation of the terms in the target documents. This stems from the assumption that terms that are more important in describing a given document are assigned a higher weight. The weighted document representations are then run through a trained classifier to categorize the documents against a set of target classes or categories. Therefore, selecting an adequate method for term weighting is important as it affects the effectiveness of the text classification. Here, we distinguish between two types of weighting schemes used to represent the document: i) unsupervised, where the representations rely on the distribution of the terms across the input documents, and ii) supervised, where representations are influenced by the target categories.

In this paper, we propose to perform Category-based Feature Augmentation (CFA). The proposed method aims to improve classification quality by introducing term-category relationships in the document representation. Our solution consists of a supervised weighting scheme derived from a modified version of the Term-Frequency Inverse-Category-Frequency (TF-ICF) scheme (cf. Section III). Different from existing approaches which are designed for document representation (cf. Section II.A), we adapt TF-ICF to produce weighted representations for the target categories. We augment each document with synthetic features that improve category classification quality. This is done by first embedding a term-category TF-ICF matrix in a one-layer Feed-Forward Neural Network (FFNN). This model then produces category representations and updates these representations using the gradient descent algorithm until reaching convergence. Compared with existing deep learning solutions (cf. Section II.B), the main contributions of our study are summarized as follows: i) this is the first approach that employs the TF-ICF weighting scheme to represent text categories, while existing solutions employ this scheme to represent the input documents rather than the target categories; ii) we introduce a new set of features inferred from the proposed weighting scheme for a more effective classification; and iii) we introduce a new classification model, GradientDescentFFNN, with a lean architecture consisting of a one-layered structure compared with its more complex deep learning alternatives for a more efficient classification. Experimental results on four benchmark datasets show that our solution improves text classification accuracy while requiring significantly fewer model parameters and computation time compared with its deep learning and deep attention alternatives.

The remainder of the paper is organized as follows. Section II reviews the related works. Section III introduces our supervised TF-ICF weighting scheme. Section IV describe our approach. Section V presents the complexity analysis. Section VI describes our experimental evaluation, before concluding in Section VII.

II. RELATED WORKS

A. Feature Representation

State-of-the-art text features mainly rely on a weighted representation of the terms in the target documents, e.g., [4, 11]. The underlying idea is that terms that are more important in describing a given document are assigned a higher weight. We distinguish between two kinds of weighting schemes: i) unsupervised and ii) supervised.

Unsupervised term weighting methods compute the weights of the terms in the specific document based on the distribution of the terms in the source documents. An example of such methods

is the standard Term Frequency – Inverse Document Frequency (TF-IDF) of the Vector Space Model (VSM) [27, 33]. While effective in many applications [32], the main drawback of such methods is that they only focus on the term distributions within the collection of documents, without considering the relationship between the source terms/documents and the target categories. Supervised term weighing methods solve this limitation by using statistical information extracted from the text documents and the corresponding categories. Various supervised term-weighting schemes have been suggested to replace the IDF factor of TF-IDF, including schemes like chi-squared ($\chi 2$) [9], information gain (IR) [9], and odd ratio (OR) [29]. More recently, the authors in [42] introduced a supervised version of the IDF called ICF (Inverse Category Frequency). The ICF scheme describes the importance of terms in describing target categories, e.g., [11, 37], and creates a different representation of each document based on its associated category. Experimental results in [10, 11, 38] indicate that the ICF weighting schemes exhibit superior performance in comparison to conventional supervised and unsupervised weighting. In this study, we adopt the ICF weighting scheme and propose a supervised scheme based on a variant of TF-ICF.

B. Classification Techniques

Text classifiers use machine learning algorithms adapted to deal with textual features by employing a two-step process consisting of a documentation representation stage and a document classification stage. For most text classifiers, the information about the target categories is only utilized at the classification stage and is not considered during the document representation stage. Therefore, recent approaches propose to replace the one-hot vectors representing the target categories by an embedding vector which integrates information about the term/document/category relationships, e.g., [41, 43, 46]. The authors of [43] present LEAM (Label Embedding Attentive Model). LEAM treats the text classification problem as a category-term joint embedding problem where each category is embedded in the same space as the term vectors. The text representation is computed using a weighted average of the term embeddings, where the weights correspond to the category-based attention scores. These scores are learned on a training set of categorized samples. Similarly, the authors in [46] formulate the text classification problem as a vector matching problem, in which they compute a matching score between the embedding vector representation of the input text and the embedding representation of each category vector. The embedding representation of the input text is generated using an input encoder, while the representation of each category is generated using a category label encoder. The authors in [41] first compute a category-based text representation from both the input terms and the target category label embeddings. Then, a convolutional neural network (CNN) is used to compute the weights of the terms from the convolution operation of both the category-based text representation and the term-based text representation (this is similar to the attention score obtained by the attention mechanism proposed in [43]). Finally, fully connected softmax layers are used to perform the classification task. In [45], the authors propose a Text Graph Convolutional Network (Text GCN). This network embeds the document into a single graph, where the nodes represent the documents and terms, and edges represent the document-term and term-term weights. TextGCN is initialized with a one-hot representation for each node. Then, TextGCN jointly learns the embeddings for both terms and documents using the labelled documents. In [26], the authors introduce another graph-based approach, GraphStar, which adds a virtual "star" node to propagate global information to all nodes. This approach learns better representations by introducing topological modifications of the original graph. In [23], the authors suggest that most category-augmented embedding solutions suffer from partial semantic loss, as they ignore the interaction between terms and sentences in the source text. Therefore, they propose LAHAN (Label-Attentive Hierarchical Attention Network) which extracts better text embeddings using a hierarchical architecture integrating category information at both term and sentence levels.

C. Discussion

As mentioned previously, traditional text classification methods leverage information about the target categories only in the classification phase, ignoring their role in the document representation phase, e.g., [2, 20]. Therefore, recent approaches have proposed supervised term weighting solutions to augment and transform the document representation with category information, using ICF (Inverse Category Frequency) weighting schemes and category label embeddings, e.g., [4, 10, 42]. They have produced better results compared with their traditional and unsupervised counterparts [11, 23, 41]. In this study, we adopt a supervised weighting scheme based on a variant of the ICF model and introduce a lean Feed-Forward Neural Network (FFNN) architecture to update this scheme.

III. ICF WEIGHTING MODEL

Besides representing each document by its individual TF-IDF vector, we represent each category by a TF-ICF vector where the dimensions represent distinctive terms and the weight of each dimension reflects the frequency of occurrence of the term in the documents belonging to the category. We then embed the new weighting scheme in our GradientDescentFFNN classification model to capture the relationships between document terms and target categories.

Table 1. Variant of the TF-ICF model adopted in our study.

Variable	Description
TF	$TF_i^j = freq_{c_j}(t_i) = \sum_{d_k \in c_j} freq_{d_p}(t_i)$
CF	$CF(t_i) = freq_{t_i}(C)$
ICF	$ICF_i = \log\left(\frac{\lvert C \rvert + 1}{CF(t_i) + 1}\right) + 1$
TF-ICF	$TF-ICF_i^j = TF_i^j \times ICF_i$

We designate by $D = \{d_1, d_2, \ldots, d_{|D|}\}$ the set of training documents, $T = \{t_1, t_2, \ldots, t_{|T|}\}$ the set of terms that occur in the documents in D (the vocabulary of D), and $C = \{c_1, c_2, \ldots, c_{|C|}\}$ the set of predefined target categories (classes or labels). We compute the TF-ICF of a term $t_i \in T$ in category $c_j \in C$ as shown in Table 1. TF represents the frequency of a term inside the set of documents pertaining to the category c_j, where more recurrent terms are assigned higher TF scores. ICF represents the fraction of

categories that contain term t_i , where less recurrent terms are assigned higher ICF scores. The less categories term t_i occurs in, the more descriptive it will be in distinctively describing the categories it occurs in, and vice versa (the more categories term t_i occurs in, the less expressive it will be in distinguishing the categories).

IV. Gradient Descent FFNN Classification

We aim to augment each document representation with a feature vector that can help identify the correct category for the document. This feature vector is of size $|C|$ where every feature dimension represents how likely the document will be assigned to a category $c_j \in C$. We aim to identify features that satisfy the following expression:

$$F(d_k) = \{f_{c_1}, ..., f_{c_{|C|}}\} \ / \ \arg\max_{c_j \in C}(F(d_k)) = c_k \quad \textbf{(1)}$$

where $F(d_k)$ represents the set of features representing document d_k, f_{c_i} is the feature representation of category c_i, and C is the set of target categories. We simplify this problem by assuming that the features inferred for every document are an aggregation of the features of every term in the document. This assumption follows the bag-of-words model and can be expressed as an arithmetic addition of the features of every term in the document. Therefore, the problem can be formulated as follows:

$$F(d_k) = \{\sum_{t_i \in d_k} f_{c_1}^{t_i}, ..., \sum_{t_i \in d_k} f_{c_{|C|}}^{t_i}\} \ / \ \arg\max_{c_j \in C}(F(d_k)) = c_k \quad \textbf{(2)}$$

Accordingly, we identify a term-category matrix of size $|T|\times|C|$ where every vector in the matrix corresponds to a category-representation of a term, and every value in the matrix corresponds to the TF-ICF weight of a term $t_i \in T$ w.r.t. a category $c_j \in C$. We introduce two term-category matrices: a seed matrix M computed using our TF-ICF weighting scheme described earlier; and a dynamic version of M denoted by M', computed using the gradient descent algorithm. The matrix computation process consists of: i) linguistic preprocessing, ii) matrix initialization, and iii) matrix update.

A. Linguistic Preprocessing

Linguistic preprocessing cleans the content of the documents through a series of steps that include tokenization, removal of stop words, removal of capitalization and punctuation, and stemming. This is crucial to form an appropriate vocabulary that will be used to represent the features of the documents in the vector space model.

Running example: Consider training documents $D = \{d_1, d_2, d_3, d_4, d_5\}$ and their desired categories $C = \{c_1, c_2, c_3\}$ in Table 2.a. The preprocessed documents and their TF vectors are shown in Table 2.b. The vector dimensions represent the terms $T = \{t_1, \ldots, t_8\}$ which will be utilized to compute the seed TF-ICF matrix M.

B. Matrix Initialization

Matrix initialization computes the seed matrix M using the TF-ICF statistic introduced previously. The resulting matrix describes the initial impact of each term $t_i \in T$ on every predefined category $c_j \in C$ (cf. running example in Table 3).

Table 2. Sample training documents used in our running example.

a. Training documents and their category labels.

	Documents D		Categories C
d_1	"*He ate a green apple then cooked an apple pie*"	c_1	*Food*
d_2	"*He wrote it on his green book*"	c_2	*Study*
d_3	"*She cooked one apple pie and another apple pie*"	c_1	*Food*
d_4	"*The grass was green*"	c_3	*Nature*
d_5	"*The apple cook book*"	c_2	*Study*

b. Preprocessed documents and their term-frequency vectors.

Preprocessed Documents D	Term-document vector dimensions							
	t_1 *eat*	t_2 *green*	t_3 *apple*	t_4 *write*	t_5 *book*	t_6 *cook*	t_7 *pie*	t_8 *grass*
<"*eat*", "*green*", "*apple*", "*cook*", "*apple*", "*pie*">	1	1	2	0	0	1	1	0
<"*write*", "*green*", "*book*">	0	1	0	1	1	0	0	0
<"*cook*", "*apple*" ,"*pie*", "*apple*" ,"*pie*">	0	0	2	0	0	1	2	0
<"*grass*", "*green*">	0	1	0	0	0	0	0	1
<"*apple*", "*cook*", "*book*">	0	0	1	0	1	1	0	0

Table 3. TF-ICF seed matrix M computed based on our running example from Table 2.

a. Term-Frequency (TF) weight matrix.

Categories C		Term-category vector dimensions							
		t_1 *eat*	t_2 *green*	t_3 *apple*	t_4 *write*	t_5 *book*	t_6 *cook*	t_7 *pie*	t_8 *grass*
c_1	*Food*	1	1	4	0	0	2	3	0
c_2	*Study*	0	1	1	1	2	1	0	0
c_3	*Nature*	0	1	0	0	0	0	0	1

b. Inverse Category Frequency (ICF) weight vector.

Category collection	Term-category vector dimensions							
	t_1 *eat*	t_2 *green*	t_3 *apple*	t_4 *write*	t_5 *book*	t_6 *cook*	t_7 *Pie*	t_8 *grass*
C	1.7	1	1.29	1.7	1.7	1.29	1.7	1.7

c. Combined TF-ICF seed matrix *M*.

Categories C		Term-category vector dimensions							
		t_1 *eat*	t_2 *green*	t_3 *apple*	t_4 *write*	t_5 *book*	t_6 *cook*	t_7 *pie*	t_8 *grass*
c_1	*Food*	1.7	1	5.16	0	0	2.58	5.1	0
c_2	*Study*	0	1	1.29	1.7	3.4	1.29	0	0
c_3	*Nature*	0	1	0	0	0	0	0	1.7

C. Matrix Update

1) Matrix Model

We propose to refine the seed matrix M into a matrix M' using the gradient descent algorithm. Note that other optimization solutions can be used to refine the matrix, such as evolutionary-developmental algorithms, e.g., [1, 7]. We propose to model the term-category matrix using a 1-layer FFNN, where the inputs correspond to the terms in the vocabulary, and the outputs correspond to the predefined categories (Fig. 1). The FFNN has a weight matrix of size $|T|\times|C|$ which represents the term-category matrix M. We adopt a 1-layered FFNN as the simplest possible

solution to the problem, yet deeper neural structures can be considered. We utilize softmax as the activation function of the final layer (i.e., the only layer in our current 1-layered network), which is suitable with our decision function (i.e., the category with the highest weight in the output determines the category of the document).

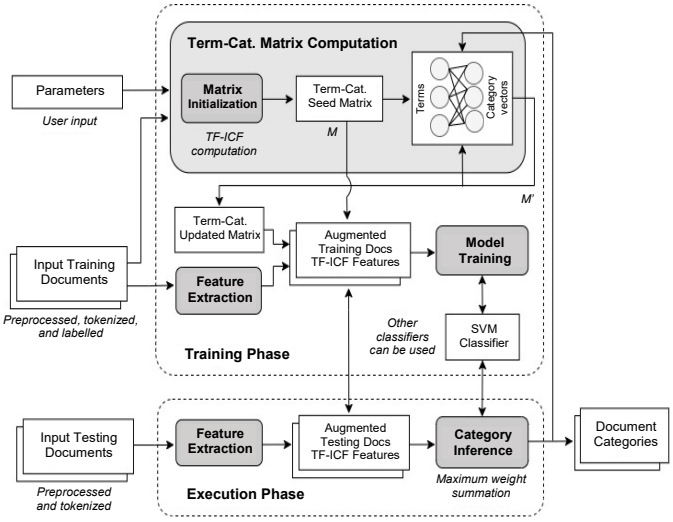

Fig. 1. Simplified diagram describing GradientDescentFFNN.

2) Matrix Update

The matrix update step is responsible for updating the matrix *M*-into-*M'* to answer our problem, and consists of training the FFNN using the gradient descent algorithm. The matrix update ends when the gradient descent algorithm reaches convergence (i.e., the matrix *M'* is of desired quality), or when reaching a maximum number of (user or system specified) iterations. We propose to evaluate the quality of the updated matrix by splitting the set of training documents *D* into i) a reference subset (70%), and ii) a validation subset (30%)[1]. The reference subset is used to optimize the FFNN and update the category-term matrix, while the validation subset is used to evaluate the quality of the updated matrix. The quality of the matrix is evaluated using the category inference accuracy, by comparing the categories inferred by the FFNN for every document in the evaluation subset with the expected document categories.

This is suitable with the problem formulation in Equation (2). Following our TF-ICF weighting scheme, the weights inside matrix *M'* reflect the likelihood of occurrence of each term in each category. Hence, performing category inference comes down to finding the category that is most described by the terms inside the document. This is reflected by computing the summation of the TF-ICF weights of the terms occurring in the document, for each category in *C*. This is analogous to the operation performed by the FFNN.

D. Augmented TF-ICF Document Features for Classification

As mentioned previously, producing the matrix *M'* using the FFNN and the gradient descent algorithm maximizes category inference accuracy. We formulate the category inference:

$$Category_{Inference}(d_k, C) = arg\ max_{c_j \in C}\left(\sum_{t_i \in d_k}\left(TF - ICF_i^j\right)\right) \quad (3)$$

where d_k is the document being processed for category inference, and c_j is the category which is assigned to d_k as a result of the category inference process. After computing the TF-ICF seed matrix *M* and generating its updated version *M'*, we produce the augmented TF-ICF document features needed for the classification phase. The overall process is visualized in Fig. 2. From both *M* and *M'*, we extract the following aggregate feature vectors for each document:

- The summation of the weights of the terms occurring in the document per category (i.e., the soft scores per category using the category inference):

$$\sum_{t_i \in d_k} TF\text{-}ICF_i^k \quad (4)$$

- The maximum of the weights of the terms occurring in the document per category:

$$\max_{t_i \in d_k}\left(TF\text{-}ICF_i^k\right) \quad (5)$$

The above features are concatenated to form the new aggregate TF-ICF feature vector for the document, which length is equal to four times the number of categories (since each feature vector has one dimension per category). We extract these aggregate TF-ICF features for all the documents in the training set, where each document is now associated with: i) a traditional TF-IDF vector representation, and ii) the aggregate TF-ICF vector representation described above. Both features are used to train the classifier model. The classifier adopted in this study is a Linear Support Vector Machine (SVM) due to its quality in performing text classification (SVM is specifically designed to handle sparse feature vectors, which is the case with high-dimensional text data). Nonetheless, other classifiers can be used following the system designer's preferences. Once trained, the classifier predicts the category of a new input document based on its traditional TF-IDF and learned TF-ICF feature vectors.

Table 4. Feature augmentation based on the TF-ICF seed matrix *M* from Table 3.c.

Categories *C*		Term-Category vector dimensions								Σ	max
		t_1 *eat*	t_2 *green*	t_3 *apple*	t_4 *write*	t_5 *book*	t_6 *cook*	t_7 *Pie*	t_8 *grass*		
c_1	*Food*	1.7	1	5.16	0	0	2.58	5.1	0	**7.74**	**5.16**
c_2	*Study*	0	1	1.29	1.7	3.4	1.29	0	0	5.98	3.4
c_3	*Nature*	0	1	0	0	0	0	0	1.7	0	0

Running example: Consider training document d_5 ="*The apple cook book*" and its desired category c_2 = *Study*. Using seed matrix *M* from our running example (Table 3), the initial *category inference* process applied on d_5 produces category c_1 = *Food* (cf. inference computation in Table 4). This assignment is incorrect since the document is supposed to be assigned with its desired category c_2 = *Study*. This discrepancy in category assignment is solved by updating the seed matrix by training the FFNN through the gradient descent method.

[1] We adopt a random 70/30 split (other forms of cross-validation splitting can be used, like *k*-fold or Monte-Carlo).

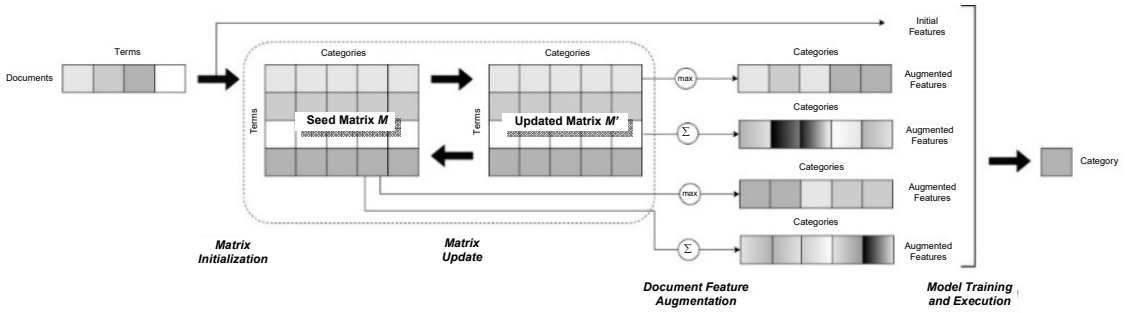

Fig. 2. Simplified diagram describing the document features computation, and the features' usage within the classification process.

V. Complexity Analysis

The computational complexity of the proposed solution is O($t \times c$) where t is the number of tokens in the vocabulary (forming the document-term vector dimensions, $t = |T|$), and c is the number of target categories (forming the document-category matrix dimensions, i.e., $c = |C|$). Table 5 compares our method with existing solutions in the literature in terms of the number of compositional parameters and the computational complexity.

Table 5. Number of parameters and computational complexity of existing solutions.

t represents the number of tokens in the vocabulary (i.e. t=$|T|$), c the number of target categories ($c = |C|$), f the number of filters in the CNN, s the size of the CNN filter, h as the number of dimensions of hidden units in the LSTM, n and e respectively represent the number of nodes and the number of edges in GCN, and p the number of dimensions of the sequence (length of the embedding).

Approach	Parameters	Computational Complexity
CNN-non-static [21]	$s \times f \times p$	$O(s \times f \times p \times t)$
Bi-LSTM [45]	$4h \times (h + p)$	$O(t \times h^2 + t \times h \times p)$
SWEM [35]	$\emptyset$	$O(t \times p)$
TextGCN [45]	$p^2 + n^2$	$O(n \times p^2 + e \times p)$
LEAM [43]	$c \times p$	$O(c \times t \times p)$
GradientDescentFFNN	$t \times c$	$O(t \times c)$

On the one hand, SWEM [35] requires the least number of parameters, followed by our approach and LEAM [43] requiring $t \times c$ and $c \times p$ respectively, where p is the number of dimensions of the sequence representation (i.e., length of the embedding). On the other hand, our solution requires the least amount of computational complexity, since the number of target categories is usually much smaller than the size of the embedding vector compared with existing solutions including SWEM, i.e., $c << p$. This is also evident in the time performance results reported in Section VI.C, where our model is clearly more efficient than existing solutions.

VI. Empirical Evaluation

A. Experimental Setup

We experiment on multiple benchmark datasets from the text classification literature: i) R52[2] is a subset of Reuters-21,578 including 9,100 documents organized in 52 categories (e.g, earn, trade, fuel, coffee), ii) Ohsumed[3] is an extract of the MEDLINE database where every document represents an abstract of a medical paper from the database, and is categorized in one of 23 diseases (e.g., endocrine diseases, eye diseases, and virus diseases),), iii) 20 News-Group[4] consists of 18,846 short news post organized into 20 categories (e.g., comp.graphics, sci.med, rec.autos, misc.forsale), and iv) AG News[5] is a collection of short news articles collected from more than 2,000 news sources, and organized into 4 categories (e.g., world, sports, business, and sci/Tec). We adopted the standard split between training and testing sets as provided by the datasets. The dataset characteristics are shown in Table 6. The experimental prototype, test data, and test results are available online[6].

Table 6. Characteristics of experimental datasets used in our study.

Dataset	Training set (# of docs)	Testing set (# of docs)	Avg. size of doc (# of terms)	# of categories
R52	6,532	2,568	113	52
Ohsumed	3,357	4,043	185	23
20NewsGroup	11,314	7,532	318	20
AGNews	120,000	7,600	39	4

We train our 1-layered gradient descent FFNN considering a maximum 100 epochs, with an early stopping rule if the validation accuracy converges and is stable for 10 consecutive epochs. We use softmax as the activation function.

B. Classification Quality

1) Weighting Scheme Evaluation

This first experiment evaluates four variants for our approach: i) one trained on our supervised document-category TF-ICF weighting scheme, and three others trained on ii) unsupervised Boolean TF [22, 34], ii) unsupervised TF-IDF [18, 45], and iv) supervised document TF-ICF [11, 38]. Table 7 presents the mean accuracy for these variants. As shown in the table, the Supervised Document-Category TF-ICF (our scheme) consistently produces better results compared with the other existing weighting models. This can be attributed to the supervised nature of the algorithm as well as the optimized document-category weighting used. As mentioned in Section II.A, supervised schemes consider the relationship between the

[2] https://ana.cachopo.org/datasets-for-single-label-text-categorization

[3] http://disi.unitn.it/moschitti/corpora.htm

[4] http://qwone.com/~jason/20Newsgroups/

[5] https://www.kaggle.com/amananandrai/ag-news-classification-dataset

[6] http://sigappfr.acm.org/Projects/CFE/

source terms/documents and the target categories, unlike the unsupervised schemes that only focus on the term distributions within the collection of documents. This explains why the proposed algorithm outperforms its unsupervised counterparts (i.e., TF and TF-IDF schemes), as it incorporates both document and category information in the document representation. This observation is also confirmed as the other supervised weighting scheme presented (i.e., the supervised document TF-ICF scheme) outperforms other unsupervised schemes on all datasets except AGNews. Even though both supervised schemes presented (i.e., original TF-ICF scheme and our scheme) are based on TF-ICF, our scheme operates on the level of the category collection. Additionally, our scheme uses the gradient descent for iterative refinement of the weights used, which leads to optimized and improved features compared to the vanilla TF-ICF scheme.

Table 7. Mean accuracy results for CFA classification approaches.

Red color refers to the best score, and green color refers to the second best for each dataset. We run all models 10 times and report mean accuracy and standard deviation results (between parentheses).

Weight Scheme	Benchmark Dataset			
	R52	Ohsumed	20NewsGroup	AGNews
Unsupervised Boolean TF [22, 34]	92.37 (0.00)	56.79 (0.00)	79.07 (0.01)	85.80 (0.11)
Unsupervised TF-IDF [18, 45]	90.16 (0.47)	50.10 (2.9)	78.11 (0.90)	**88.00 (0.06)**
Supervised Document TF-ICF [11, 38] (original scheme)	**92.7 (0.06)**	**66.9 (0.00)**	**79.14 (0.01)**	85.61 (0.34)
Supervised Document-Category TF-ICF (our scheme)	**93.85 (0.00)**	**68.79 (0.05)**	**86.02 (0.05)**	**91.75 0.01)**

2) *Varying Dimensionality and Training Data Size*

We analyze the effect of varying i) the number of dimensions representing the document feature vectors, and ii) the size of the training data. We vary the number of dimensions by considering the top weighted feature dimensions ranging over: 1,000, 2,000, 5,000, 10,000 and 15,000 dimensions per vector. We also vary the size of the training data by performing k-fold cross validation, for k= 2, 4, 5 and 10. For every model trained with a number of folds k, we compute and report average accuracy on the corresponding testing dataset. Results on the R52 dataset are shown in Fig. 3.

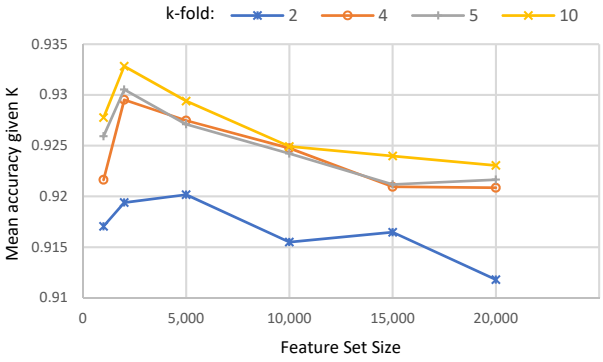

Fig. 3. Mean accuracy results with varying dimensionality and training data size applied on the R52 dataset (similar results were produced with the other datasets [5]).

We highlight the following observations: First, our solution suffers from a drop in accuracy levels when increasing the number of feature dimensions. This implies that better category features are inferred when using a smaller set of top weighted terms. We can attribute the drop in accuracy to the curse of dimensionality. In fact, increasing the number of dimensions will increase the sparsity of the data. This will make it more difficult for the gradient descent algorithm to converge, affecting the quality of the features produced in the matrix update phase. Perhaps, choosing the size of the feature set should be a hyperparameter that can be tuned before applying the CFA approach. We aim to investigate this behavior in a dedicated empirical study. Second, results clearly show that classification accuracy improves when increasing the number of folds k. This means that we were able to learn better document-category mappings with larger training data size.

3) *Comparative Study*

We also compare our CFA approaches against state-of-the-art text classification solutions and benchmark datasets. We utilize the same parameter settings described in Section VI.A, and we set the number of feature dimensions to be 15,000 for all models.

Table 8. Comparative mean accuracy results on benchmark datasets.

Red color refers to the best score, and green color refers to the second best. Results for alternative approaches are verified from their respective papers. Standard deviation results are shown between parentheses.

Approach		Benchmark Dataset			
		R52	Ohsumed	20NewsGroup	AGNews
Non-parametric	TF-IDF+LR	86.95 (0.00)	54.66 (0.00)	83.19 (0.00)	-
	FastText	92.81(0.09)	57.70 (0.49)	79.40 (0.30)	-
	FastTextBigrams	90.99 (0.05)	55.69 (0.39)	79.67 (0.29)	-
Deep Learning	CNN-non-static	87.59 (0.48)	58.44 (1.06)	82.15 (0.52)	-
	Bi-LSTM	90.54 (0.91)	49.27 (1.07)	73.18 (0.18)	-
	SWEM	92.94 (0.24)	63.12 (0.55)	85.16 (0.29)	-
	GraphCNN	92.75 (0.22)	**63.89 (0.53)**	81.42 (0.32)	-
	GraphStar	**95.00** (0.30)	63.86 (0.53)	**86.90** (0.30)	-
Deep Attention	PV-DM	74.92 (0.05)	51.14 (0.22)	29.50 (0.07)	-
	PV-DBOW	78.29 (0.11)	46.65 (0.19)	74.36 (0.18)	-
	LEAM	91.84 (0.23)	58.58 (0.79)	81.91 (0.24)	**91.75** (0.24)
	LAHAN	-	-	-	**92.45** (0.00)
CFA	GradientDescent FFNN	**93.85** (0.00)	**68.79** (0.05)	**86.02** (0.05)	**91.75** (0.01)

Mean accuracy results in Table 8 show that our solution ranks best on one of the benchmark datasets used in our study (i.e., Ohsumed) and second best on the other three datasets (i.e., R52, 20NewsGroup, and AGNews). Our solution computes *term-category* relationships using supervised TF-ICF weighting. This produces more accurate classification results compared with i) FastText and FastTextBigrams which make use of the n-gram model to extract *term-term* relationships, ii) PV-DM and PV-DBOW which are trained to predict terms that are randomly sampled from the document, and iii) SWEM and GraphCNN which make use of *term-document* relationships through their weighted graph and hypergraph structures. Our solution utilizes a shallow (1-layered) architecture which is significantly easier to train and manipulate. This is mainly due to our supervised TF-ICF scheme which is the first proposal to use a variation of TF-ICF weighting for representing the target text categories, while existing solutions are designed for input document representation rather than target category representation. Our lean classification model was designed around our term-category TF-ICF scheme,

producing results which are on a par with and surpass many deep learning and supervised weighting solutions. Nonetheless, our approach is outperformed by its deep learning counterpart GraphStar on three datasets. This can be attributed to the semantic similarities that exist between certain target classes (e.g., "talk.politics.guns" and "talk.politics.misc", and "comp.graphics" and "comp.windows.x" in 20NewsGroup), compared with more distinctive categories. An closer analysis of the results showed that similarities between the target categories in 20NewsGroup produced similarities between the TF-ICF category feature vectors, resulting in confusion and misclassification for our approach. A possible solution would be to fine-tune and calibrate the parameters of our classifier according to the target categories of each dataset. This can be handled as an optimization problem using a number of known techniques that apply linear programming and machine learning to identify the best weights for a given problem class, e.g., [6, 48]. This issue needs to be further investigated in a dedicated study.

The reader can refer to [5] for a more detailed description of the experimental results, as well as the whole framework.

Note that transformer-based methods, e.g., [24, 25], are not included in our comparative study since they rely on transfer learning and provide additional knowledge that is extrinsic to the dataset. For this reason, we compare with classification methods which infer derived features from the dataset itself without including any external knowledge or transfer learning.

C. Time Performance

We also evaluate our solution's time performance and compare it with existing solutions. Time experiments were carried out on 12th Gen Intel(R) Core(TM) i7-1260P processor with a 2.66 GHz processor and 16 GB of RAM. Table 9 reports computation time as the wall-clock time for 1000 iterations, applied on the Yahoo Answers dataset. Results were scaled and averaged across multiple runs, to ensure consistency across the different models. Results show that our solution uses much less parameters and require significantly less computation time compared with existing solutions.

Table 9. Comparing the number of parameters and computation time with existing solutions.

Approach	Parameters	Computation time (s)
CNN-non-static [21]	541K	171
Bi-LSTM [45]	1.8M	598
SWEM [35]	61K	63
LEAM [43]	65K	65
CFA-GradientDescentFFNN	20K	17

VII. Conclusion

In this paper, we introduce a new approach for Category-based Feature Augmentation (CFA). The proposed method aims to improve classification quality by introducing term-category relationships in the document representation. Our solution consists of a supervised weighting scheme derived from a modified version of the Term-Frequency Inverse-Category-Frequency (TF-ICF) scheme [37]. Different from existing approaches which are designed for document representation, e.g., [11, 38, 42], we adapt TF-ICF to produce weighted representations for the target categories. We augment each document with synthetic features that improve the category classification quality. This is done by first embedding a term-category TF-ICF matrix in a one-layer feed-forward neural network titled GradientDescentFFNN. This model then produces category representations and updates these representations using the gradient descent algorithm until reaching convergence. GradientDescentFFNN provides a lean architecture, compared with its more complex deep learning and deep attention model alternatives, while mostly producing improved and comparable classification results.

We are conducting more experiments by integrating corpus-based statistics (e.g., distributional thesaurus [40], association rule mining [17], unsupervised clustering [16]), and considering structure data augmentation [36, 39], and semantic data augmentation [8, 44], to augment the target feature vectors. We plan to consider man-made knowledge bases like WordNet [31, 47] and DBPedia [14, 30], to compare corpus-based and knowledge-based feature representations. We also aim to generalize this approach to consider more complex architectures like transformer-based models.

References

[1] Abboud R. and Tekli J., *Integration of Non-Parametric Fuzzy Classification with an Evolutionary-Developmental Framework to perform Music Sentiment-based Analysis and Composition.* Springer Soft Computing, 2019. 24(13): 9875-9925

[2] Aggarwal C. and Zhai C., *A Survey of Text Classification Algorithms.* Mining Text Data, 2012. pp. 163-222.

[3] Ahmed H., et al., *Detecting Opinion Spams and Fake News using Text Classification.* Security and Privacy, 2018. 1(1).

[4] Alsaeedi A., *A Survey of Term Weighting Schemes for Text Classification.* International Journal of Data Mining, Modelling and Management, 2020. 12(2): 237-254.

[5] Attieh J. and Tekli J., *Supervised Term-Category Feature Weighting for Improved Text Classification.* Knowledge Based Systems 2023. 261:110215.

[6] Azar D., et al., *A Combined Ant Colony Optimization and Simulated Annealing Algorithm to Assess Stability and Fault-Proneness of Classes Based on Internal Software Quality Attributes.* International Journal of Artificial Intelligence (ISSN 0974-0635), 2016. 14:2.

[7] Byerly A. and Kalganova T., *Homogeneous Vector Capsules Enable Adaptive Gradient Descent in Convolutional Neural Networks.* IEEE Access, 2021. 9: 48519-48530.

[8] Cai L., et al., *A Hybrid BERT Model That Incorporates Label Semantics via Adjustive Attention for Multi-Label Text Classification.* IEEE Access, 2020. 8:152183-152192.

[9] Debole F. and Sebastiani F., *Supervised Term Weighting for Automated Text Categorization.* Text Mining and Its Applications, Springer, Berlin, Heidelberg., 2004. pp .81–97.

[10] Domeniconi G., et al., *A Study on Term Weighting for Text Categorization: A Novel Supervised Variant of TF-IDF.* Inter. Conf. on Data Technologies and Applications (DATA'15) 2015. pp. 26-37.

[11] Domeniconi G., et al., *A Comparison of Term Weighting Schemes for Text Classification and Sentiment Analysis with a Supervised Variant of TF-IDF.* Inter. Conf. on Data Technologies and Applications (DATA'16) 2016. pp. 39-58.

[12] Fares M., et al., *Difficulties and Improvements to Graph-based Lexical Sentiment Analysis using LISA.* IEEE Inter. Conf. on Cognitive Computing (ICCC'19), 2019. pp. 28-35.

[13] Fares M., et al., *Unsupervised Word-level Affect Analysis and Propagation in a Lexical Knowledge Graph.* Elsevier Knowledge-Based Systems, 2019. 165: 432-459.

[14] Flisar J. and Podgorelec V., *Improving Short Text Classification using Information from DBpedia Ontology.* Fundamenta Informaticae, 2020. 172(3): 261-297.

[15] Han J. and Akbari M., *Vertical Domain Text Classification: Towards Understanding IT Tickets Using Deep Neural Networks.* AAAI Conf. on Artificial Intelligence (AAAI'18), 2018. pp. 8202-8203.

[16] Haraty R., et al., *An Enhanced k-Means Clustering Algorithm for Pattern Discovery in Healthcare Data.* Intelligent Journal on Distributed Sensor Networks, 2015. 11: 615740:1-615740:11.

[17] Haraty R. and Nasrallah R., *Indexing Arabic Texts using Association Rule Data Mining.* Library Hi Tech, 2019. 37(1): 101-117.

[18] Joulin A., et al., *Bag of Tricks for Efficient Text Classification.* Conference of the European Chapter of the Association for Computational Linguistics (EACL'17), 2017. pp. 427-431.

[19] Kaddoura S., et al., *A Spam Email Detection Mechanism for English Language Text Emails Using Deep Learning Approach.* IEEE International Workshops on Enabling Technologies: Infrastracture for Collaborative Enterprises (WETICE'20) 2020. pp. 193-198.

[20] Kadhim A., *Survey on Supervised Machine Learning Techniques for Automatic Text Classification.* Artificial Intelligence Review, 2019. 52(1): 273-292.

[21] Kim Y., *Convolutional Neural Networks for Sentence Classification.* Conference on Empirical Methods in Natural Language Processing (EMNLP'14), 2014. pp. 1746–1751.

[22] Lee J. H., *Properties of Extended Boolean Models in Information Retrieval.* Proceedings of the ACM SIGIR Conference, 1994. Springer-Verlag New York, pp.182-190.

[23] Li X., et al., *Label-Attentive Hierarchical Attention Network for Text Classification.* Proceedings of the 2020 5th International Conference on Big Data and Computing (ICBDC'20), 2020. pp. 90–96.

[24] Lin X., et al., *ET-BERT: A Contextualized Datagram Representation with Pre-training Transformers for Encrypted Traffic Classification.* World Wide Web Conference (WWW'22) 2022. pp. 633-642.

[25] Lin Y., et al., *BertGCN: Transductive Text Classification by Combining GCN and BERT.* Cornell University - Computer Science - Computation and Language, 2021. CoRR abs/2105.05727.

[26] Lu H., et al., *Graph Star Net for Generalized Multi-Task Learning.* Computing Research Repository, 2019. CoRR abs/1906.12330 (2019).

[27] McGill M., *Introduction to Modern Information Retrieval.* 1983. McGraw-Hill, New York.

[28] Mironczuk M. and Protasiewicz J., *A Recent Overview of the State-of-the-Art Elements of Text Classification.* Expert Systems and Applications 2018. 106: 36-54.

[29] Mladenic D. and Grobelnik M., *Feature Selection for Classification based on Text Hierarchy.* Conf. on Automated Learning and Discovery (CONALD'98), 1998.

[30] Mouriño-García M., et al., *Wikipedia-based Hybrid Document Representation for Textual News Classification.* Soft Computing, 2018. 22(18): 6047-6065.

[31] Poostchi H. and Piccardi M., *Cluster Labeling by Word Embeddings and WordNet's Hypernymy.* Australasian Language Technology Association Workshop (ALTA'18) 2018. pp. 66-70.

[32] Revina A., et al., *IT Ticket Classification: The Simpler, the Better.* IEEE Access, 2020. 8:193380-193395.

[33] Salton G., *Automatic text processing: the transformation, analysis, and retrieval of information by computer.* Addison-Wesley Longman, Boston, MA, USA 1989. pp. 530.

[34] Salton G. and Buckley C., *Term-weighting approaches in automatic text retrieval.* Info. Processing and Management, 1988. 24(5):513 -523.

[35] Shen D., et al., *Baseline Needs More Love: On Simple Word-Embedding-Based Models and Associated Pooling Mechanisms.* Annual Meeting of the Association for Computational Linguistics (ACL'18), 2018. pp. 440-450.

[36] Taddesse F.G., et al., *Semantic-based Merging of RSS Items.* World Wide Web Journal, 2010. 13(1-2): 169-207, Springer Netherlands.

[37] Tang Z., et al., *Several Alternative Term Weighting Methods for Text Representation and Classification.* Knowledge Based Systems, 2020. 207:106399.

[38] Tang Z., et al., *An Improved Supervised Term Weighting Scheme for Text Representation and Classification.* Expert Systems and Applications, 2022. 189: 115985.

[39] Tekli J., et al., *Minimizing User Effort in XML Grammar Matching.* Elsevier Information Sciences Journal, 2012. 210:1-40.

[40] Tekli J., et al., *Full-fledged Semantic Indexing and Querying Model Designed for Seamless Integration in Legacy RDBMS.* Data and Knowledge Engineering, 2018. 117: 133-173.

[41] Wang C. and Tan C., *Label-Based Convolutional Neural Network for Text Classification.* International Conference on Control Engineering and Artificial Intelligence (CCEAI'2021) 2021. pp. 136–140.

[42] Wang D. and Zhang H., *Inverse-Category-Frequency based Supervised Term Weighting Schemes for Text Categorization.* Journal of Information Science and Engineering, 2013. 29(2): 209-225.

[43] Wang G., et al., *Joint Embedding of Words and Labels for Text Classification.* Annual Meeting of the Association for Computational Linguistics (ACL'18), 2018. 2321-2331.

[44] Wei J. and Zou K., *EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks.* Conference on Empirical Methods in Natural Language Processing (EMNLP'19), 2019. (1) 2019: 6381-6387.

[45] Yao L., et al., *Graph Convolutional Networks for Text Classification.* AAAI Conference on Artificial Intelligence (AAAI'19), 2019. pp. 7370-7377.

[46] Zhang H., et al., *Multi-Task Label Embedding for Text Classification.* Conference on Empirical Methods in Natural Language Processing (EMNLP'18), 2018. pp. 4545-4553.

[47] Zhu X., et al., *An Improved Class-Center Method for Text Classification Using Dependencies and WordNet.* Natural Language Processing and Chinese Computing (NLPCC'19), 2019. (2): 3-15.

[48] Zou F., et al., *A Reinforcement Learning Approach for Dynamic Multi-objective Optimization.* Information Sciences, 2021. 546: 815-834.