Representation Learning in Text Mining: A Systematic Review

Basma Alharbi and Xiangliang Zhang

Abstract. Representation learning is a fundamental subject in data mining and machine learning, where it places emphasis on extracting descriptive feature sets from (mostly) high-dimensional data. Consider the subfield of text mining as an example. Text documents are often represented using Vector Space Models (VSM), a.k.a., "bag-of-words", where each document is described by a vector of unique terms from a corpus-level vocabulary set (dictionary). Given the nature of human language, it is clear that this representation produces very sparse and high-dimensional feature space, which is problematic to most classic data mining algorithms. Hence, it comes the need to reduce the high-dimensional space using various existing and emerging techniques. In this study, we conduct a systematic review of several existing methods of representation learning in text mining. We include both subcategories of this field, shallow dimensionality reduction methods and deep learning methods, from which unsupervised methods are selected for our study. To evaluate the selected methods, we apply Support Vector Machine (SVM) - a classification method - on the new representations and compare the performance.

Keywords. Representation learning, dimensionality reduction, deep learning, text mining, document representation.

1. Introduction

High-dimensional data are considered problematic to most data mining and machine learning techniques. This is mainly due to the curse-of-dimensionality phenomenon, in which higher dimensions exponentially increase the associated computational cost and decrease the performance. In order to overcome this issue, various methods of representation learning are proposed, where they all share a common goal, to reduce the dimension of the feature space. Existing representation learning methods vary from one-layer (shallow) models to multi-layer (deep) models. This includes well-established methods of dimensionality reduction (feature selection and transformation), as well as newly emerging approaches (deep learning).

Text mining is among the numerous fields that suffer from the implications of high-dimensional data. The growth of text documents available on-line has given rise to many critical issues in text mining such as document representation, document retrieval, clustering and classification. There is an ever increasing need for fast and efficient methods that can handle growing collections of text data. This is mainly because Vector Space Models (VSM) are the current practice in representing text documents, where documents are represented by vectors of term frequency. The terms (features) are corpus dependent, and thus very high in dimension. This results in very sparse feature representations, which makes it difficult to apply classic data mining techniques. Thus, document representation is a key factor in many text mining tasks and applications. Good and compact representations allow for less storage space, fast document retrieval and accurate document clustering and classification. In fact, the success of many machine learning methods depends heavily on the choice of feature representations.

M. Kosta and T. Sturm, eds. Proceedings of the MACIS 2013, Nanning, China, December 11-13, 2013.

Current practices to overcome this issue, in text mining, can be categorized into two main schools: 1) developing algorithms and techniques that work efficiently on high dimensional sparse feature space, and 2) attempting to map the high dimensional feature space into lower dimension using representation learning. One example of the first approach includes adapting k-means for sparse and high-dimensional data (e.g., spherical k-means). In this study, we believe that the second approach, representation learning, is more appealing as it facilitates applying existing data mining approaches once the feature space is reduced.

To date, there is a lack of a systematic study that compares existing (both classic and emerging) document representation methods. In this paper, our main objective is to conduct a systematic review and an exhaustive experimental study of the main existing representation methods for text documents. We aim to identify, compare, and evaluate top representation learning methods for text mining. To assess the quality of the representations, we study the effect of document representation on document classification and evaluate the results achieved.

In order to have a fair comparison of the methods, we focus in this paper on unsupervised representation learning techniques. Both supervised and semi-supervised methods are not considered in this research. We study the following methods: Singular Value Decomposition (SVD), Principal Component Analysis (PCA), Independent Component Analysis (ICA), Latent Dirichlet Allocation (LDA), and deep Auto-Encoders (AE). From these techniques, SVD, PCA, and ICA are linear feature transformation methods, LDA is a nonlinear topic modeling algorithm, and deep AE is an unsupervised multi-layer model. We use feature selection by document frequency as the baseline for comparing the above methods.

SVD is also known as Latent Semantic Indexing (LSI) in text mining. It is a matrix factorization method which decomposes a matrix into singular values and singular vectors (right and left). SVD can be used to reduce noise from the dataset, by eliminating small singular values and reconstructing the original matrix. The right and left singular vectors can be used to learn a new representation for both documents and terms. In that case, documents will be represented by the new set of singular values (latent topics) instead of the terms (VSM).

PCA is a linear feature transformation method, that aims at finding a set of orthogonal (uncorrelated) Principal Components (PCs) from the original feature space. PCs are linear combinations of the original features. The importance of PCs is measured by their corresponding variance. PCs thus can be ordered by their variances. This means that the 1st PC captures the largest variability in the data, and the 2nd PC captures the 2nd largest variability, and so on. PCA is used as a dimensionality reduction method by ignoring the PCs with the lowest variance (thus, lowest variability). The number of selected PCs is usually less than the original feature space. In text data, this means that we are eliminating unnecessary concepts, which may be noise or outliers, and capturing the main conceptual components, which can be used to represent each document in a lower dimensional space with little information loss.

ICA is very similar to PCA, although it places more strict assumptions. Instead of finding orthogonal latent variables, ICA finds non-gaussian and statistically independent hidden variables. Statistical independence is a much stronger assumption than un-correlation. In text mining, the Independent Components (ICs) represent latent topics in the dataset, where the independent assumption means that topics are statistically independent.

Emerging approaches employed the concept of deep learning to text corpora, where the idea is to automatically produce multiple levels of representations for the data. In this stream of approaches, Ranzato and Szummer used a system of deep auto-encoders to automatically learn multiple representations of documents in a corpus [6]. This study proposed a semi-supervised method, and showed that even a few labeled samples can enhance the achieved results (when compared to unsupervised methods). For a fair comparison, we adopt an unsupervised deep auto-encoder method in this work.

The rest of this paper is organized as follows. Section 2 provides a comprehensive summary of representation learning in general and detailed discussion of selected methods. Section 3 describes the

experimental settings adopted in this study. Section 4 presents the dataset used and results achieved. Finally, a conclusion is presented in section 5.

2. Representation Learning

A common pipeline of any data mining task is composed of the following subtasks: data collection, preprocessing (removing outliers, handling missing values, and cleaning noise), representation learning (extracting a good feature representation from the data), algorithm deployment (e.g., clustering, classification, pattern recognition), and finally result evaluation. Experiments show that even with the best machine learning algorithms, good performance can be achieved only when the investigated data is a good representation of the problem under study.

Generally speaking, representation learning methods can be grouped into two main categories: single layer models (SLM)/shallow learning, and multi-layer models (MLM)/deep learning. In this context, a multi-layered model (deep learning model) is a representation learning model which is comprised of several layers of transformation nodes. In each layer, the nodes map the current feature space into a lower dimension. Single layer models, as the name implies, contain only one layer of transformation.

Methods of representation learning can also be grouped as either supervised, semi-supervised or unsupervised. In supervised feature learning, methods use sample labels in order to acquire better representation for the data. Examples of such methods include Information Gain (IG), Mutual Information (MI), and χ^2 statistic. On the other hand, unsupervised feature learning does not make use of sample labels during feature learning. Ranzato and Szummer provide evidence that a semi-supervised approach is better than a completely unsupervised approach, in deep learning [6]. In this study, we focus on unsupervised representation learning methods for the sake of fair comparison (being independent from other tasks). Further details of different methods are provided in the sections below.

2.1. Single Layer Models

Classic dimensionality Reduction (DR) techniques are often regarded to as SLMs. In literature, DR methods are grouped into two main subcategories: feature selection and feature transformation.

2.1.1. Feature Selection. Feature selection is the process of reducing original feature space into a lower dimension by eliminating less important features and selecting more important ones. There exist a number of methods that measure the importance of each feature in the feature space, and thus rank them by the corresponding importance value. Given sample labels, the importance of one feature is usually measured by its "relevance" to the labels, which falls into the category of supervised feature learning. In unsupervised feature learning, the importance is measured based on the occurrence frequency.

Yang and Pedersen conducted a systematic comparison of five feature selection methods, including both supervised and unsupervised techniques, which are: Document Frequency (DF), Information Gain (IG), Mutual Information (MI), χ^2 statistic (CHI), and Term Strength (TS) [10]. For details on the selected methods and the experimental design and settings, please refer to [10]. In general, the study assessed the validity of the methods based on the results achieved on classification on the reduced sets. The reported results favored both IG and CHI, while DF produced comparable results but with much lower time and space complexity. From this study, it can be concluded that -among the five feature selection methods -, DF is an adequate feature selection approach given its low computational cost and relatively high reported accuracy.

There are other methods that learn features combinatorially, such as forward feature selection and sequential feature selection. Instead of ranking each feature individually by different criteria, these methods evaluate the importance of a set of features as a union. These approaches are often computationally expensive, especially in large and sparse data, though with a chance to achieve better performance as redundancy among selected features is reduced.

2.1.2. Feature Transformation. Feature transformation methods transform the original features into a set of new features that span a lower dimension, by using either linear or non-linear functions on the original features. Typical examples of linear transformations include SVD, PCA, and ICA. Non-linear transformations in text mining include - among others, LDA.

Singular Value Decomposition (SVD). SVD is a matrix factorization technique, which decomposes a rectangular matrix to the following:

$$X_{m,n} = U_{m,r} * S_{r,r} * V_{n,r}^T \tag{1}$$

where r is the rank of X, each column of $U_{m,r}$ is the left singular vector of X (eigenvector of XX^T), each column of $V_{n,r}$ is the right singular vector of X (eigenvector of X^TX), and $S_{r,r}$ is a diagonal matrix of singular values of X. Matrix U and V are both column orthonormal, while entries of S are sorted in decreasing order.

Let X be a text dataset, where m is the number of documents, and n is the feature space (terms). The SVD results imply that there are r latent concepts in X, and the strength of the i-th concept is the singular value $S_{i,i}$. The i-th column of U shows how each document is similar to the i-th concept, while the i-th column of V shows how each feature (term) is relevant to the i-th concept. Eliminating small singular values in S and the corresponding columns in U and V, results in a reduced matrix, $\hat{X}$, with k singular values and singular vectors: $U_{m,k}$, $S_{k,k}$ and $V_{n,k}$.

$$\hat{X}_{m,n} = U_{m,k} * S_{k,k} * V_{n,k}^T \tag{2}$$

The reconstruction $\hat{X}_{m,n}$ is a rank k matrix that is the best approximation of X for minimizing the approximate error. In other words,

$$||X_{m,n} - \hat{X}_{m,n}||_{Frobenius} = min_B||X_{m,n} - B_{m,n}||_{Frobenius}$$

where $B_{m,n}$ is any rank k matrix. Reduced SVD, a.k.a., Latent Semantic Indexing (LSI), is used in text mining to reduce the dimensions of the original dataset X, as shown in eq. [3]. $\hat{D}_{m,k}$ is a new document representation, and is often used to measure similarity between documents. $\hat{T}_{n,k}$ is a new term representation, and can be used to measure similarity between terms.

$$\hat{D}_{m,k} = U_{m,k} * S_{k,k} \tag{3}$$

$$\hat{T}_{n,k} = V_{n,k} * S_{k,k} \tag{4}$$

Principal Component Analysis (PCA). PCA is an unsupervised, non-parametric, linear dimensionality reduction technique. In PCA, the feature vector space is re-defined using a set of uncorrelated (i.e., orthogonal) principal components (PCs). The set of PCs are ranked based on how much variance each PC captures from the original feature vector space, in which variance is assumed to indicate importance. In other words, the first principal component captures the largest variance in the data, the second principal component captures the second largest variance in the data, and so on. As a result, the original feature space can be reconstructed by projecting the data over all PCs. More importantly, the original feature space can be reduced by projecting the data over an eliminated set of PCs, where PCs with lowest variances are eliminated first. This assures that less-important information (i.e., noise) are removed from the data.

To understand the basic intuition behind PCA, let's consider the following definitions from linear algebra. Recall that, in a given vector space, every feature vector can be represented by a linear combination of some finite *orthonormal basis vectors*. The *naive* set of orthonormal bases, in an m-dimensional vector space, is the $m \times m$ identity matrix, in which each row represents an orthonormal basis vector. In PCA, the goal is to find a new set of orthonormal bases which *best* represents the original data. In which, the *best* representation is interpreted as a representation which eliminates noise, redundancy, and finds rotations which captures the maximum variance in the set. The notion of variance and covariance is essential in understanding the assumptions behind PCA. It is assumed that variables (or directions) with maximum variances capture interesting structure, and

directions with minimum variances hold less relevant knowledge and are mostly regarded as noise. At this point, it is clear that the goal of PCA is to capture directions with maximum variance, and the naive set of bases -more often than not-, does not capture the largest variances. As a result, it is important to find the best rotation of the naive bases which maximizes the variance. In addition to that, a basic intuition behind dimensionality reduction is to eliminate redundancy in the set, that is, highly correlated variables among which, one can be easily predicted from others. Redundancy in a multivariate data is measured using the magnitude of the covariance.

Given the above definitions, the problem of PCA is to find a transformation matrix P such that $Y = XP$, where X is an $m \times n$ data matrix of m samples and n features and Y is a new de-correlated and ordered feature vector. This means that the off-diagonal values in $cov(Y)$ are zeros (no correlation), and that the diagonal variances in $cov(Y)$ are ordered from high to low. There are two possible solutions to diagonalize the covariance matrix, using either eigenvector decomposition or SVD.

The steps required to reduce the data dimensions of a dataset $X_{m,n}$ with m samples and n features using PCA are described in Alg. 1. First, X is centralized by subtracting the mean of each feature, resulting $\hat{X}$. Then, the covariance matrix C_X of the centralized data is computed, where C_X is an $n \times n$ matrix, whose diagonal elements are the variances and off-diagonal elements are the covariances. After that, the eigenvalues and eigenvectors of the covariance matrix are obtained. The set P are composed of orthogonal eigenvectors ranked by the variances (i.e., eigenvalues). The eigenvector with the largest eigenvalue (the first column of P) is the first principal component. To reduce the dimensions in the dataset, eigenvectors with low eigenvalues are eliminated. The number of selected PCs can be determined by a threshold on the portion of variance captured. For example, the first k PCs are selected if k is the smallest value to make the sum of the first k eigenvalues greater than 95%. This implies that the selected PCs, $\hat{P}$, can capture 95% of the variance in the data. Y can be finally obtained by projection over the reduced set of principal components, $\hat{P}$.

Algorithm 1 Principal Component Analysis

Require: $\mathbf{X}$

1: $\hat{\mathbf{X}} = \mathbf{X} - \bar{\mathbf{X}}$ // centralize the data
2: $\mathbf{C_X} = \frac{1}{m-1}\hat{\mathbf{X}}^{\mathbf{T}}\hat{\mathbf{X}}$ // covariance of the centralized data
3: $\mathbf{P} = eig(\mathbf{C_X})$ // eigenvector of covariance
4: $\mathbf{Y} = \hat{\mathbf{X}}\hat{\mathbf{P}}$ // project the data on selected PCs

Independent Component Analysis (ICA). ICA is an unsupervised statistical method that aims at identifying latent variables from a set of observed variables (features). The ICA model has three main components: random variables ($\mathbf{x}$), hidden variables ($\mathbf{s}$), and a mixing matrix ($\mathbf{A}$). Random variables are the observed data variables, which are often referred to as data features. In this model, it is assumed that random variables are a linear mixture of some hidden components (see eq. [5]).

$$\mathbf{x} = \mathbf{As} \tag{5}$$

where $\mathbf{x}$ is a vector of random variables, $\mathbf{s}$ is a vector of independent components, and $\mathbf{A}$ is the mixing matrix.

Hidden components are also known as independent components, latent variables, hidden sources, and/or factors. The mixing matrix determines the weights of the linear combination in the equation. Hidden variables and the mixing matrix are unknown and need to be estimated. In ICA, it is assumed that the hidden components are non-gaussian and statistically independent. These assumptions simplify the estimation of variables, as will be seen later on.

As mentioned previously, $\mathbf{x}$ is the only observed (known) variable, both $\mathbf{A}$ and $\mathbf{s}$ are unknown. If we assume that the mixing matrix $\mathbf{A}$ is known and has an inverse, then finding the independent

components can be achieved by using equation [6]. From this equation we can see that ICA produces new representation, which is a linear transformation of the original feature variables.

$$\begin{aligned} \mathbf{W} &= \mathbf{A}^{-1} \\ \mathbf{s} &= \mathbf{W}\mathbf{x} \end{aligned} \tag{6}$$

Now, in order to solve for $\mathbf{s}$, we first have to estimate $\mathbf{W}$. Let $y = \mathbf{w}^T\mathbf{x}$, where $\mathbf{w}$ is an unknown vector. Note that if $\mathbf{w}$ is a row from $\mathbf{A}^{-1}$, then y is actually one of the independent components. Then, substitute the value of $\mathbf{x}$ from [5], and let $\mathbf{z} = \mathbf{A}^T\mathbf{w}$. This results in $y = \mathbf{z}^T\mathbf{s}$, see [7]. Thus, y is a linear combination of the independent components, given the weights in $\mathbf{z}$.

$$\begin{aligned} y &= \mathbf{w}^T\mathbf{x} \\ y &= \mathbf{w}^T\mathbf{A}\mathbf{s} \\ y &= \mathbf{z}^T\mathbf{s} \end{aligned} \tag{7}$$

where

$$\mathbf{z} = \mathbf{A}^T\mathbf{w}$$

At this point, we make use of the Central Limit Theorem, which states that the distribution of the standardized sum of N independent variables approaches gaussian as N approaches infinity. Given the assumption that the independent components are non-gaussian, and using the rule of the Central Limit Theorem, the linear combination y is more gaussian than any of the individual independent components. This conclusion holds in all cases, given the assumptions, except in the case where there is only one nonzero element in $\mathbf{z}$. In that case, the distribution is non-gaussian since y is actually one of the independent components.

As a result, in order to find the independent components, we need to solve for $\mathbf{w}$ that maximizes non-gaussianity of y. Note that a $\mathbf{w}$ which maximizes non-gaussianity, minimizes the independence of hidden variables, and will result in a $\mathbf{z}$ with only one nonzero element. From the above, we can see the importance of having non-gaussian independent components. In fact, the assumptions are critical to the success of ICA. However, ICA can still perform well if there exists only one gaussian independent component.

Now, ICA can be thought of as an optimization problem, with an objective function that maximizes non-gaussianity (or minimizes independence), and an optimization algorithm that searches for the optimum values. Theoretically, any measure of non-gaussianity can be used as an objective function in ICA. Main examples of these measures include: Kurtosis measure, Negentropy measure, and approximation of negentropy. The Kurtosis method is given in [8], where nonzero values indicate non-gaussianity. It is a classical method, though it is not widely used due to its sensitivity to outliers. Negentropy is a method based on differential entropy, which measures the entropy of continuous random variables. Findings from information theory show that larger entropies indicate gaussian random variables, while small values indicate non-gaussianity. Negentropy is a reliable measure of gaussianity, however, it is computationally expensive and thus not widely used. An approximation of negentropy is a reasonable alternative for a non-gaussianity measure, and is used widely as an objective function for ICA.

$$kurt(y) = E\{y^4\} - 3(E\{y^2\})^2 \tag{8}$$

$$H(y) = -\int f(y) log f(y) dy \tag{9}$$

Latent Dirichelet Allocation (LDA). LDA, proposed by Blei et al. in [1], is an unsupervised, statistical topic modeling technique, in which it assumes that documents are composed of a mixture of hidden topics, and topics are represented by a probability distribution over words. Thus, the general goal of LDA is to represent each document in a corpus by a set of topics, rather than by the entire set of corpus vocabulary. To do so, LDA adopts a generative model which is presented in Alg. 2.

This generative model assumes that, a bag-of-words' document is generated by first selecting the distribution of topics (e.g., 30% topic a, and 70% topic b), then generating words constrained by the selected topic distribution and the word distribution associated with each topic.

Algorithm 2 Latent Dirichlet Allocation: Generative Model

for each document **do**
  Select the number of words, N, according to a poisson distribution
  Select a mixture of topics for the document
  for word $\in N$ **do** // to generate every word in a document
    Select a topic, according to the multinomial distribution in step **(3)**
    Generate a word, given the multinomial word distribution for the selected topic
  end for
end for

Assuming that this generative model represents how documents are generated, LDA attempts to find the two important distributions: word distributions for topics, and topic distributions for documents. In other words, LDA attempts to find the set of topics, that generated the observed collection of documents, where this is achieved through backtracking the generative model. More precisely, to learn these distributions, a number of statistical inference techniques exist where any can be employed. Examples of possible methods includes Gibbs sampling and Expectation-Maximization algorithm (EM).

In this study, we provide a brief description of Gibbs sampling. Let w, t, d refer to *word*, *topic* and *document* respectively, and D is the set of all documents in the corpus. Let k be the number of topics existing in D. Note that this parameter has to be specified by users, usually according to prior knowledge about D. An appropriate setting of k is important, as small k results in broad topic distributions and large k breaks a single topic into several refined ones. Given the above, the probability distribution over topics for a document is denoted by $p(t)$, and θ represents the multinomial distribution over topics for all documents, such that $\theta(d) = p(t)$. $p(w|t)$ is the probability distribution over words given topic t, and ϕ is the multinomial distribution over words for all topics, where $\phi^{(j)} = p(w|t = j)$, and j is the jth topic. This being said, the task of Gibbs sampling is to infer ϕ and θ from D. The procedure followed by Gibbs sampling is presented in Alg. 3.

Algorithm 3 Latent Dirichlet Allocation: Gibbs Sampling

Require: k, D
for each document $d \in D$ **do**
  Randomly assign word w to topic t
end for
repeat
  for each document $d \in D$ **do** // improve the random initialization
    for each word $w \in d$ and $t \in k$ **do**
      $p(t|d)$ = probability distribution over topics given document d
      $p(w|t)$ = probability distribution over words given topic t
      $p(w|d) = p(w|t)p(t|d)$
    end for
  end for
until until convergence or maximum number of iterations reached

2.2. Multi-Layer Models

Multi-layer models can be thought of as a stack of learning blocks, in which the output of each block is provided as an input to the next block. In such models, the main objective of each block is to reduce the dimension of the input space and thus produce a much compact representation. The idea of deep models was inspired by the experiment in [9], in which Von Melchner et al. showed that the

processing unit in the brain "*cortex*", is not a special-purpose processor, but rather a multi-purpose general learning block that can be tuned given the right training set. The implications of this study have led to the idea of deep learning where the general belief is: one model can handle any task, given the proper training. Here, we investigate one deep learning method, deep auto-encoders. The details of this method are provided below.

Deep Auto-Encoders (AE). Deep AE is an unsupervised multi-layer model, where in each layer, a learning block, i.e., an *auto-encoder*, is composed of an *encoder* and a *decoder*. The objective of each auto-encoder is to produce a *reduced* and *representative* new feature space. Given this objective, the task of the encoder is evident; it generates the lower dimensional codes from the high-dimensional input space. The task of the decoder, on the other hand, is to assure that the encoded features are representative of the original features. Thus, a decoder in this framework, attempts to reconstruct the original input from the generated codes. The reduced feature space is assumed to be representative if the reconstructed data is similar to the original data.

This being said, when training a deep auto-encoder, the objective is to minimize the *reconstruction error*, i.e., the difference between the original and the reconstructed data, where two sets of weights need to be learned (for encoder and decoder). According to Hinton and Salakhutdinov, using gradient descent - similarly to neural networks - in deep models, where weights are far from a good solution, does not provide good results [3]. As a result, the authors proposed an alternative solution, which uses Restricted Boltzmann Machines (RBM) to initialize the weights of each layer in the model one at a time [3]. The objective of RBM is to make the distribution of the reconstructed input similar to the distribution of the original input. Once the training is done, the model can then be used to reduce the dimension of the input data, one layer at a time. This results in producing new representations with multiple levels of granularity, from each layer in the model.

3. Experimental Design

This section describes the details of the conducted experiment. The experiment is composed of five main layers: preprocessing, term-weighting, representation learning, classification, and evaluation. The input to the model is a bag-of-word representation of text documents.

3.1. Preprocessing

Motivated by the work of Yang and Pedersen [10], we use feature selection as a preprocessing method to eliminate unnecessary terms from the data. Thus, the computation time of the examined algorithms is reduced. We use both Document Frequency (DF) and Term Frequency (TF) as the preprocessing methods, mainly due to their low computation requirements and their equivalent performance to the best methods [10]. DF is defined as the number of documents a word appeared in, and TF is the total number of term occurrences in the entire corpus. We eliminate terms that appeared in only two documents, as well as terms that appeared only twice in the entire corpus. This is based on an assumption that terms with low DF/TF are not important in representing documents.

3.2. Term weighting

In this step, we test the effect of different term weighting approaches on document representation. Term weighting indicates the value associated with each term-document pair in the dataset. The most intuitive approach is the word count, in which the value associated with every pair of term-document is simply the term frequency. Different weighting approaches take into consideration term frequency, document frequency, and a normalization factor. The reason behind adopting both term and document frequency is, on one hand, to penalize very frequent words (e.g., stop words) and, on the other hand, to favor unique keywords.

In this study, we adopt the notations of the SMART information retrieval system [7], as shown in Table 1. The predefined notations specify unique characters for the three main factors: term frequency, document frequency, and normalization. The notation *ltc* means that we used logarithmic term frequency as specified by the first character *l*, *idf* document frequency as denoted by the middle

Table 1. Notations of SMART information retrieval system

Term Frequency		Document Frequency		Normalization	
n (natural):	$tf_{t,d}$	n (no):	1	n (none): 1	
l (logarithm):	$1 + \log\{tf_{t,d}\}$	t (idf):	$\log \frac{N}{df_t}$	c (cosine):	$\frac{1}{\sqrt{w_1^2+\ldots+w_M^2}}$
a (augmented):	$0.5 + \frac{0.5+tf_{t,d}}{max\{tf_{t,d}\}}$	p (prob idf):	$max(0, \log \frac{N-df_t}{df_t})$		
b (boolean):	1 if $tf_{t,d} > 0$; 0 otherwise				
L (log average):	$\frac{1+\log(tf_{t,d})}{1+\log\{avg_{t\in d}\}}$				

character t, and cosine normalization as indicated by the last character c. In our experiments, we used the combinations of n, l, and b term frequencies, n and t document frequencies, and n and c normalization. As a result, we have a total of 12 unique term weightings.

3.3. Representation Learning

In the third layer, after applying various term weighting approaches, we apply selected representation learning methods. Specifically, from single layer models, we applied SVD, PCA, ICA [4], and LDA [8], and we used deep auto-encoders [5] from multi-layer models. In addition, we used feature selection using DF as a baseline method.

3.4. Classification

In order to evaluate the new representations, we apply binary document classification on the reduced sets and compare the results. Better classification performance is assumed to indicate better representation. In this study, we used Support Vector Machines (SVM) as the main classification method. For simplicity, we only tested using the sigmoid kernel, as our objective is to evaluate various document representation methods rather than to find the best classification method for documents.

3.5. Evaluation

To avoid any ambiguities of the results, we use 10-fold cross validation and report the average and the standard deviation of accuracy. We also test the quality of different document representations given different training sizes. That is, we examine the effect of different representation methods on small training sets, starting from as small as 1% of the data and gradually increasing to up to 35% of the data. In order to compare the performance of different measures, given different training portions, we used ROC curves and the Area Under the Curve (AUC).

4. Experimental Evaluation

4.1. Data

Reuters [2] is a news agency with a focus on business and financial news. The preprocessed version of the *Reuters* benchmark dataset has a collection of 8293 articles that appeared in Reuters newswire in 1987. Each document is described by a vector space model of 18933 unique terms. The dataset is labeled manually by agents from Reuters Ltd., where there is a total of 65 categories. The categories are not equally distributed, which means that some categories have few samples while others have larger number of samples.

In the adopted version of the data, stop words and punctuations are eliminated, terms are not trimmed, numbers are not eliminated, and there exist misspelled terms in the dictionary. The values in the dataset represent word count in each document. Preliminary investigation of the data confirmed the sparsity of the feature vector; 75% of terms in the dataset occurred less than 13 times, and 50% of terms occurred less than 5 times. These values are coherent with the fact that document-by-term matrices are very sparse, and there exist many infrequent terms.

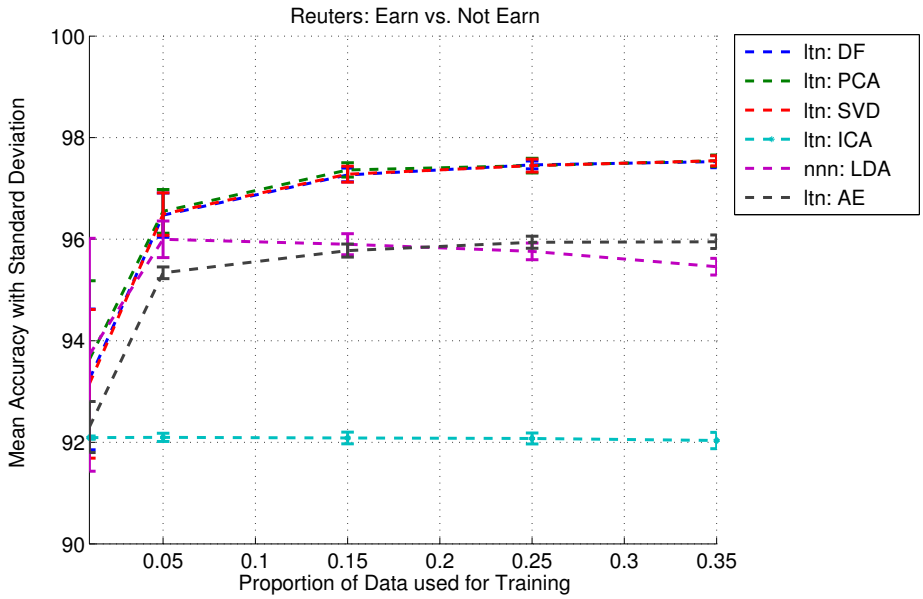

FIGURE 1. Reuters - Earn vs. Not Earn: mean accuracy with standard deviation when the proportion of training data varies. Each representation learning method is applied with its best term weighting, LDA with *nnn* (only the term frequency) and other methods with *ltn* (log of term frequency and *idf*).

4.2. Results

Experiments reported here are conducted using a 1-vs-all approach, where the selected labels are earn vs. not-earn. The class proportions for the selected labels are 45% and 55% respectively. After pre-processing the data to remove infrequent terms, we applied 12 different term weighting combinations, followed by 6 representation learning methods. This resulted in 72 different variation of the dataset. SVM, with 10-fold cross validation, was applied to each version of the set where the number of training samples varied from 1% to 35%. In order to report all the results, we compute the Area Under the Curve (AUC) for each data version, given the different training sizes. The AUC results of classification are reported in Table 2. Fig. 1 shows the best results for each representation learning approach.

TABLE 2. Reuters - Earn vs. Not Earn: AUC for average accuracy

Term	Representation Learning					
Weighting	DF	PCA	SVD	ICA	LDA	AE
nnn	0.901	0.906	0.901	0.809	0.930	0.901
lnn	0.903	0.909	0.903	0.454	0.926	0.903
bnn	0.881	0.886	0.881	0.817	0.916	0.881
ntn	0.926	0.925	0.927	0.824	0.918	0.927
ltn	0.942	0.943	0.942	0.894	0.912	0.942
btn	0.939	0.940	0.939	0.414	0.908	0.939
nnc	0.649	0.666	0.649	0.478	0.537	0.650
lnc	0.633	0.659	0.633	0.414	0.537	0.633
bnc	0.566	0.610	0.566	0.776	0.537	0.566
ntc	0.564	0.608	0.564	0.785	0.537	0.564
ltc	0.545	0.567	0.545	0.851	0.537	0.545
btc	0.537	0.539	0.537	0.862	0.537	0.536

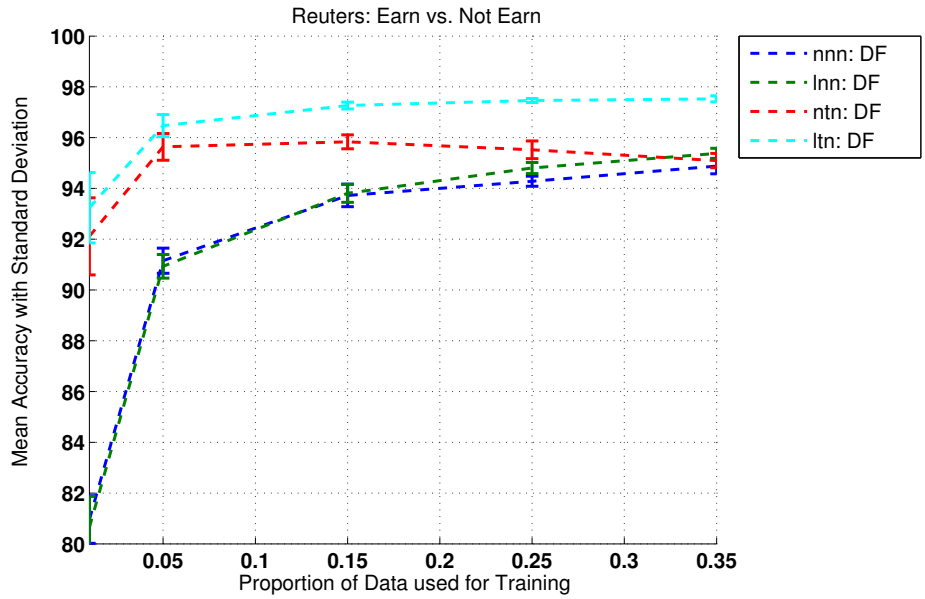

FIGURE 2. Reuters - Earn vs. Not Earn: mean accuracy with standard deviation when the proportion of training data varies. This figure shows the impact that *idf* has on representation learning (and eventually on classification results). A significant improvement is achieved when *idf* is applied, especially with smaller training sets.

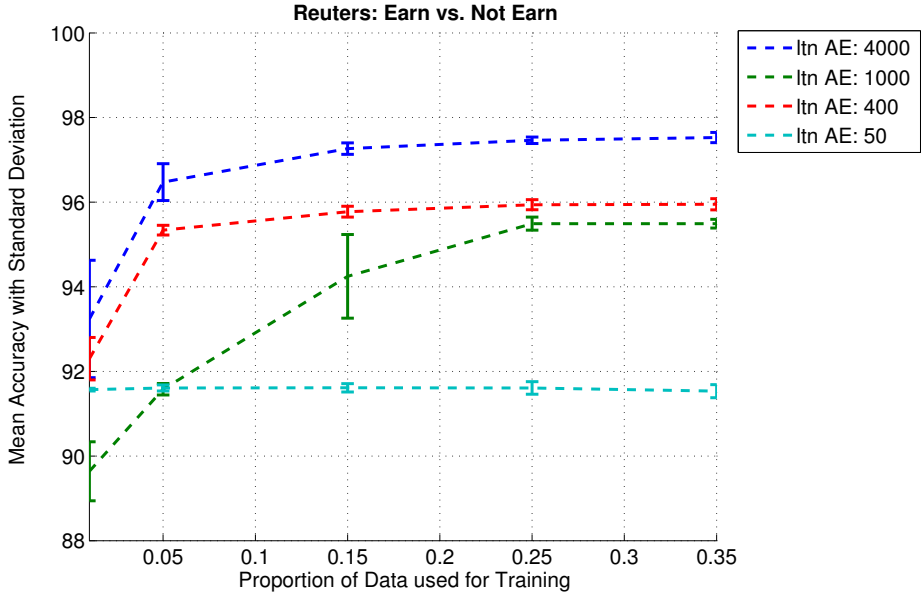

FIGURE 3. Reuters - Earn vs. Not Earn: mean accuracy with standard deviation when the proportion of training data varies. This figure shows the impact of using features learned from each layer's output in a deep auto-encoder. We used a total of 4 layers where the new dimensions are 4000, 1000, 400, and 50, respectively.

From the achieved results, we can conclude that for both PCA and SVD, cosine normalization has inferior effect on the classification results. The same conclusion applies to DF, LDA, and AE. That is, in all methods except ICA, cosine normalization decreases the performance of classification. In addition to that, document frequency $-t-$ from the SMART notation (a.k.a., *idf*) has the opposite impact. We observe that when we apply *idf* to almost any representation learning method, we gain increase in the reported accuracy. In fact, *idf* improves classification results especially with lower training sizes. When the portion of training size increases, *idf* has lower impact on the results, see Fig. 2. For all methods except LDA and ICA, we can conclude that *ltn* has the best performance, followed by *btn*, then *ntn*.

For all variations of ICA, we observe that the reported accuracies are almost consistent when the training portions are different. That is, whether we train the classifier with only 1% of the data or 35%, we achieve very similar results. Similarly to previous methods, cosine normalization reduces classification performance. However, this cannot be generalized as in the $-tc$ case, significant improvements are achieved when cosine normalization is applied. To conclude for ICA, the best representations are *ltn*, *btc*, and *ltc* respectively. As for LDA, the best term weighting schemes are: *nnn*, *lnn*, and *bnn*. Both document frequency and cosine normalization have negative impact on the performance.

The performance of deep auto-encoders did not exceed other methods. In fact, the impact of different term weights was almost similar to that of DF, PCA, and SVD. To assess the performance of each level in deep auto-encoders, we apply SVM to the output of each layer. The layer outputs had the following number of dimensions in order: 4000, 1000, 400, and 50. Fig. 3, shows the results achieved, with varying training set size. In general, the largest dimension size (4000) had the best performance. The output of the second layer, 1000 dimension, had lower performance when compared to the previous one. Interestingly, the output of the third layer showed an improvement over the second layer, though the dimension continued to decrease, i.e., only 400 in this layer. This implies that higher layers in auto-encoders, which reduces the feature space, attempt to learn better and more compact representations.

Reflecting upon the results achieved, we observe that no single method has significantly outperformed the rest of the selected models. DF, the simplest and most computationally inexpensive method, achieved comparable results to time-consuming methods such as LDA and deep auto-encoders. However, experiments show that proper term weighting has significant impact on the results achieved. Specifically, *ltn* outperformed other term weighting methods for all selected representation learning, except LDA, in which experiments verified that LDA works best with word count term frequency.

5. Conclusion

In this work, we present a systematic review of representation learning methods with an application to text mining. Our main objective is to compare existing models with new emerging models (i.e., deep learning). We perform document classification to evaluate the validity of each selected method. We demonstrate that well-established single layer models such as PCA outperforms more complex methods: LDA and deep Auto-encoders. Following this work, we will investigate the validity of other subcategories of deep learning including supervised and semi-supervised methods.

References

[1] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[2] D. Cai, X. Wang, and X. He. Probabilistic dyadic data analysis with local and global consistency. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML'09)*, pages 105–112, 2009.

[3] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[4] T. Kolenda, L. K. Hansen, J. Larsen, and O. Winther. Independent component analysis for understanding multimedia content. In *Neural Networks for Signal Processing, 2002. Proceedings of the 2002 12th IEEE Workshop on*, pages 757–766. IEEE, 2002.

[5] R. B. Palm. Prediction as a candidate for learning deep hierarchical models of data. *Technical University of Denmark, Palm*, 2012.

[6] M. Ranzato and M. Szummer. Semi-supervised learning of compact document representations with deep networks. In *Proceedings of the 25th international conference on Machine learning*, pages 792–799. ACM, 2008.

[7] G. Salton. The smart retrieval system experiments in automatic document processing. 1971.

[8] H. Shan and A. Banerjee. Mixed-membership naive bayes models. *Data Mining and Knowledge Discovery*, 23(1):1–62, 2011.

[9] L. Von Melchner, S. L. Pallas, and M. Sur. Visual behaviour mediated by retinal projections directed to the auditory pathway. *Nature*, 404(6780):871–876, 2000.

[10] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *ICML*, volume 97, pages 412–420, 1997.

Basma Alharbi
King Abdullah University of Science and Technology (KAUST)
P.O. Box 1639
Thuwal 23955-6900,
Kingdom of Saudi Arabia
e-mail: `basma.harbi@kaust.edu.sa`

Xiangliang Zhang
King Abdullah University of Science and Technology (KAUST)
P.O. Box 2925
Thuwal 23955-6900,
Kingdom of Saudi Arabia
e-mail: `Xiangliang.Zhang@kaust.edu.sa`