McMaster University

STATS 771

Statistical Research Project

Feature Extraction and Selection Techniques for Handling High-Dimensional Data in NLP

Author:
Tolu Olusooto
(1235801)

Professor:
Dr. Angelo J. Canty

April 25, 2023

Abstract

This paper examines the curse of dimensionality in Natural Language Processing (NLP) and its impact on model overfitting. The focus of the paper is to compare 3 dimension reduction techniques (DRTs) - Latent Semantic Analysis (LSA), Linear Discriminant Analysis (LDA), and Auto-Encoder (AE) - used in text classification tasks. The paper is divided into four main sections - Data, Literature Review, Experiment, and Conclusion. The Data section analyzes the dataset used and describes the preprocessing steps. The Literature Review section showcases prior applications and observations of the DRTs investigated in this paper. The Experiment section outlines the methodology used to compare the DRTs, and the results of the experiment are presented. The Conclusion section highlights the key findings of the paper, with the TF-IDF method being the most efficient and effective reduction technique in terms of classification accuracy and computational time. The Conclusion section highlights the key findings of the paper. The TF-IDF method is found to be the most efficient and effective reduction technique in terms of classification accuracy and computational time.

1 Introduction

Natural Language Processing (NLP) is defined as a set of techniques used by computers to process and/or synthesize human languages (Eisenstein 2019a). In recent years, NLP has become more integrated into our daily lives thanks to the advancements in technology and machine learning (ML). From tools such as Google Translate, email classification, and most recently ChatGPT, NLP-based tasks have made it easier for people to communicate and share information. NLP is an interdisciplinary topic drawing concepts from subjects such as computational linguistics, statistics, and artificial intelligence (Eisenstein 2019a).

Though there are many societal benefits gained from the use of NLP, the computational and statistical challenges from processing such high dimensional data (HDD), a dataset where the number of parameters is large relative to the data sample or larger than data sample, can be quite a strain (Buehlmann and Geer 2013). These limitations can lead to a reduction in processing speed and classification accuracy (Ayesha et al. 2020). When processing HDD in ML, a known statistical problem tends to occur: curse of dimensionality. Curse of dimensionality is for every small increase in dimension or parameter in the dataset, an exponentially larger increase in data sample is needed for an accurate prediction or classification to be made by the model (Ayesha et al. 2020). A symptom of curse of dimensionality is model overfitting. That is increasing the dimensions of the data improves the model performance; however, past a certain dimensionality, this will have the opposite effect (Ayesha et al. 2020). The effects of overfitting a model are a noticeable decline in performance from the training to testing set. With regards to NLP data such as text, also known as corpus, another common symptom of curse of dimensionality is data sparsity (Allison et al. 2006). Data sparsity is when the training set does not contain enough features or words to accurately classify text samples outside the training set, leading the model to overfit (Allison et al. 2006).

To address the curse of dimensionality, there is a class of ML algorithms known as dimensionality reduction techniques (DRT) that can be used. These techniques fall into one of two categories: feature selection and feature extraction (Ayesha et al. 2020). Feature selection involves subsetting the most relevant features in the dataset relative to the given task, while feature extraction transforms the data to a lower dimension while retaining the most important information (Ayesha et al. 2020).

1.1 Motivation

With the increasing amount of data being generated for NLP models such as BERT and ChatGPT to process, the issue of model overfitting becomes a significant concern. While computational advancements such as GPUs and cloud computing have facilitated the development and training of larger models, statistical advancements like DRT are employed to mitigate overfitting issues associated with these models. This paper aims to compare the most common DRTs used in text classification today. To keep the paper concise, only 3 DRTs were examined in detail. These include one supervised, one unsupervised and one semi-supervised method. Latent Semantic Analysis (LSA) was the unsupervised method used, while Linear Discriminant Analysis (LDA) was the supervised method used. Finally, Auto-Encoder (AE) was the semi-supervised method chosen.

The paper is divided into four main sections, namely Data, Literature Review, Experiment, and Conclusion. In the Data section, the dataset used is analyzed and the preprocessing steps are detailed. The Literature Review section showcases some prior applications and observations of the 3 DRTs investigated in this paper by other studies. Additionally, this section provides a theoretical derivation of each DRT. The Experiment sections describes the methodology used to compare the DRTs and the results of the experiment is presented. Lastly, the Conclusion section summarizes the key findings of the paper.

2 Data

The data used in this paper is called 'BBC News Archive' (Greene and Cunningham 2006). The dataset contains 2,225 news articles published by BBC News from 2004-2005 (Greene and Cunningham 2006). These articles are classified into one of five topics: (Business, Entertainment, Politics, Sport, Tech). The data was made available by Derek Greene and Pádraig Cunningham; who used it in their 2006 paper 'Practical Solutions to the Problem of Diagonal Dominance in Kernel Document Clustering'. Both the processed and raw data were made available, and for the purpose of this paper the processed data will be used. Processed data contains 4 columns: the article title, filename, news category and content of the article (Greene and Cunningham 2006).

2.1 Data Preprocessing

All the data wrangling was done using the `python` library `pandas` (McKinney et al. 2010). During the cleaning stage, 98 duplicate articles were removed. The title and content of each article were merged into a single column named "text," forming a comprehensive corpus. Additionally, a encoded representation of the categories was generated, while the filename, content, and title columns were discarded.

After processing the data, exploratory data analysis (EDA) was performed, revealing an imbalance in the data with the majority of the articles being either sports or business-related. The tech articles represented the smallest share. However, the overall difference between the categories is relatively small, with the most significant difference from the expected ratio being 4%. The imbalance in data will not be addressed as it is the reality of real-world datasets and the level of imbalance is minimal. The EDA also revealed that some high-frequencies words were common amongst all 5 categories, which did not assist in distinguishing them from one another. The words `"said"` and `"will"` were the most common words in the dataset. Given that the dataset comprises news articles, it is reasonable to see the heavy representation of these verbs in the dataset.

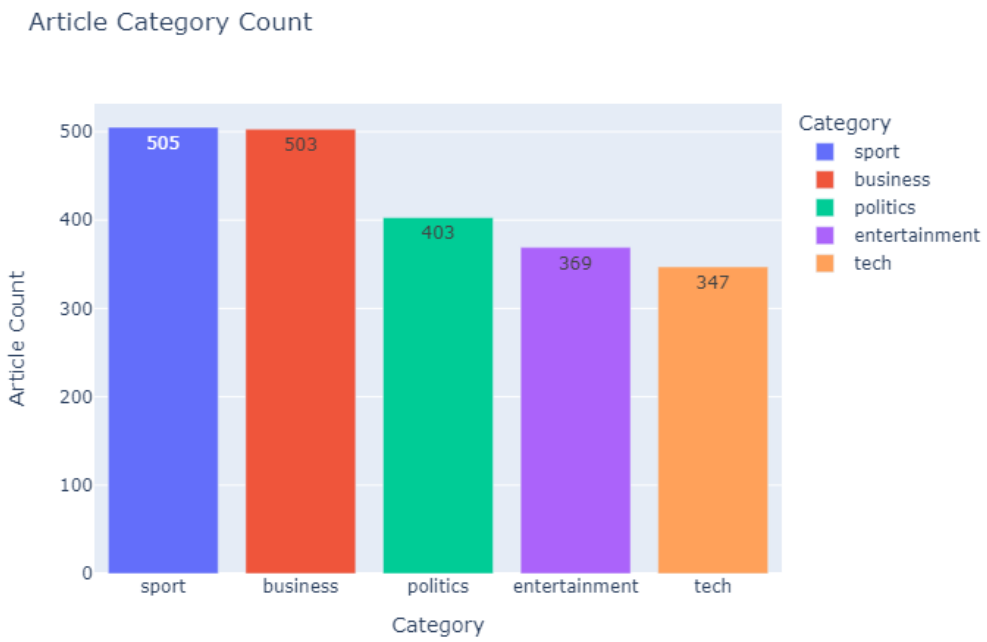


Figure 1: The article count for each of the 5 categories.

Category	Expected Ratio	Actual Ratio	Difference
sport	20%	23.74%	3.74%
business	20%	23.65%	3.65%
politics	20%	18.95%	1.05%
entertainment	20%	17.35%	2.65%
tech	20%	16.31%	3.69%

Table 1: The deviation from the expected distribution for each of the 5 categories.

The processed data was further cleaned by removing numbers and symbols and converting all the words to lowercase in each text. Furthermore, stopwords were dropped from the text. Stopwords are words in text that add little to no contribution to the overall message or topic of the text (Eisenstein 2019b). Some examples of stopwords are `"the,"` `"a,"` `"said,"` `"will,"` and `"for."` Stopwords removal is good practice in NLP since they offer negligible discriminatory power when assigning a text to one of the five categories.

Their inclusion may introduce noise to the model, which could ultimately decrease its accuracy (Manning et al. 2009c). The removal of stopwords is a feature selection technique as it reduces the number of words that the model has to process. A comprehensive list of stopwords was referenced during the stopwords removal process, the list is a combination of stopwords from the `nltk` library and the `stopwordsiso` library (Bird et al. 2009a); (Diaz and Suriyawongkul 2020).

Continuing with NLP best practices, the dataset was normalized through lemmatization after removing stopwords (Bird et al. 2009b). Lemmatization is the reduction of a word to its root form or lemma, for example, converting `"running"` or `"ran"` to `"run"` (Manning et al. 2009b). This process takes into account a word's morphology and matches the resulting lemma against a vocabulary set, ensuring it is a valid word (Manning et al. 2009b). The `nltk` library's `WordNetLemmatizer` module was used to lemmatize the dataset (Bird et al. 2009a). However, further analysis of the corpus found the module did not accurately interpret certain words, specifically acronyms. Words such as `"US"`, short for United States, or `"OS"`, short for operating systems, were treated as plurals of the letters `"U"` and `"O"`, respectively. To solve this issue, a conditional statement was added to exclude these acronyms from lemmatization. After applying the aforementioned processes, the dataset went from 65,553 potential features to 24,134, which is more than a 63% reduction in potential features.

Figure 2: The wordclouds are created from the cleaned dataset, showing the most common words within each of the 5 categories.

The wordclouds shown above were generated using the `wordcloud` library in Python (Oesper et al. 2011). Based on these wordclouds, it is apparent that certain highly distinctive topical words appear quite frequently within each category. However, it is worth noting that generic words such as `"people"` and `"time"` are also amongst some of the most common words within each category.

The bag-of-words (BoW) module from Python's `sklearn` library was used to transform the dataset into vectors (Pedregosa et al. 2011). The BoW method represents each unique word as a column, and each article in the dataset as a row (Manning et al. 2009a). A frequency count is given to each word depending on how often it appears within an article, regardless of the word's context (Manning et al. 2009a). The transformed dataset is now a document-term matrix with 2,127 rows representing the articles and 24,134 columns representing the count of unique words. Most of the values in this matrix will be 0, making it a sparse matrix.

The Term Frequency-Inverse Document Frequency (TF-IDF) formula was applied to the BoW vectors, resulting in a numerical representation of the relevance of each word within a particular article, with respect to the other articles (Manning et al. 2009a). TF-IDF is the product of the term frequency (TF) and inverse document frequency (IDF) (Manning et al. 2009a). TF is the number of times a word appears within an article, while IDF measures the significance of that word in determining the category of the article (Manning et al. 2009a). The TF-IDF formula is as follows (Manning et al. 2009a):

$$IDF_t = \log(\frac{N}{df_t}) \tag{1}$$

$$TF\text{-}IDF_{t,d} = TF_{t,d} \times IDF_t \tag{2}$$

where,
t = a given word;
N = number of articles;
df = number of articles a given word appears in;
d = a given article

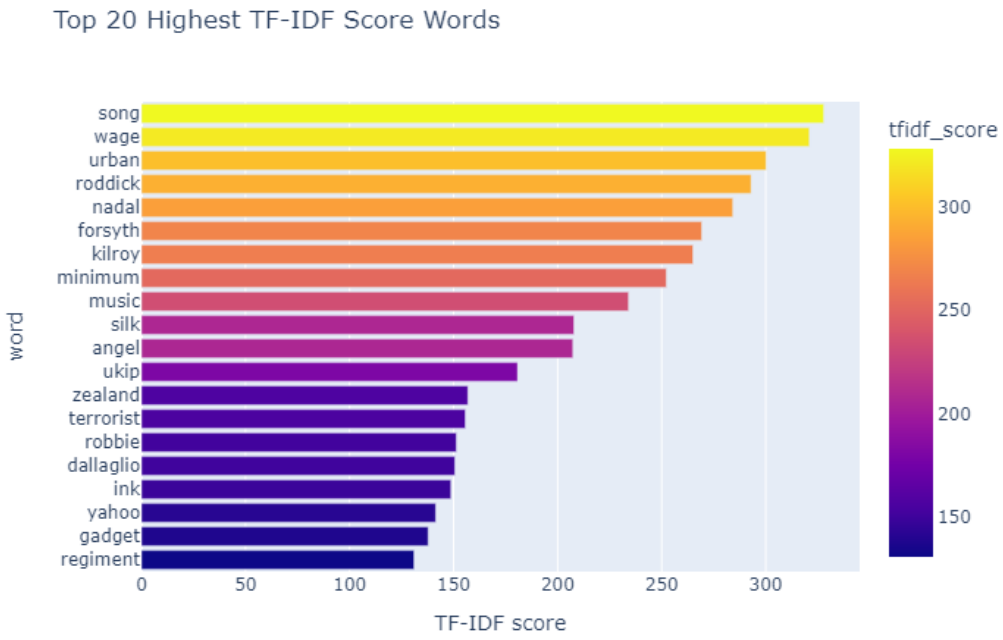


Figure 3: The top 20 most important words in the BBC News Archive dataset, based of the TF-IDF formula.

A high TF-IDF score implies that the word is relevant to the article and important in categorizing it; the opposite is true if the score is low (Manning et al. 2009a). Based on the plot above, it can be inferred that words such as `"song"` and `"music"` will useful in properly categorizing entertainment articles, while `"link"` and `"gadget"` will be effective in categorizing tech-related articles. Similarly, words like `"urban"` and `"terrorist"` can be used to correctly classify political articles. There are 3 obvious categories with topical words that have high TF-IDF scores, however, the 2 remaining categories may present a challenge for a classifier to differentiate from the other categories. TF-IDF can serve as a feature selection technique, allowing the user to control the number of terms in the dataset. This can be achieved by either specifying a maximum number of terms n, which selects the top n terms with the highest TF-IDF scores, or by setting a threshold m for the minimum number of times a term must appear in a document to be considered a feature (Manning et al. 2009a). The TF-IDF matrix was used as the

baseline, to serve as a point of comparison against the other datasets where DRT was implemented.

3 Literature Review

Several journal articles were reviewed with the aim of better understanding the effectiveness of the 3 DRTs being investigated. In the study, "Overview and Comparative Study of Dimensionality Reduction Techniques for High-Dimensional Data" conducted by Ayesha et al. (2020), it was shown that LSA was most effective on high-dimensional (HD) textual data, with the ability to understand the context of a word in relation to the document. However, the study found that the LSA method was both computationally and temporally expensive and had a tendency to overfit the model (Ayesha et al. 2020). In their paper, "How Dimensionality Reduction Affects Sentiment Analysis NLP Tasks: An Experimental Study", Akritidis and Bozanis (2022) observed that reducing the dimensionality of an HD textual dataset using LSA resulted in an increase in training time as the number of dimensions increased. However, the untreated dataset with a dimension space of 10^4 needed less training time, and the model accuracy was better compared to the reduced dataset with a dimensionality of 10^3 (Akritidis and Bozanis 2022). A moderate reduction in dimensionality was more time-efficient and had a small to moderate impact on model accuracy (Akritidis and Bozanis 2022).

In their paper "Linear Discriminant Analysis in Document Classification", Torkkola (2001) found that using LDA treated data was more effective at minimizing the misclassification error rate than the original untreated dataset. The original dataset, with 5,718 dimensions, resulted in an error rate of 11.2%, while the LDA reduced dataset, with 513 dimensions, had an error rate of 11.4% (Torkkola 2001). However, the LDA-treated dataset, which had a dimension of 64, resulted in an error rate of 7.8% (Torkkola 2001). Similar to the paper mentioned above, other studies have also highlighted the effectiveness of LDA in selecting the most discriminative words from HDDs and noted the added benefit of reduced train and test time. For example, in the paper "Employing Fisher Discriminant Analysis for Arabic Text Classification" by AbuZeina and Al-Anzi (2018), it was found that the use of semantic removed LDA transformed features performed similarly to the semantic-rich LSA extracted features when classifying Arabic text.

In their paper "Deep Variational Auto-Encoder for Text Classification", Xie et al. (2019) found that AE was more efficient than classical DRTs like PCA in dealing with sparse and discrete data, with an accuracy of 96% as opposed to PCA's accuracy of 89%. AE was more efficient on smaller datasets as its performance began to drop on larger datasets (Xie et al. 2019). According to the 2019 research article titled "Empirical Comparison between Autoencoders and Traditional Dimensionality Reduction Methods" by Fournier and Aloise, AE outperformed classical DRTs, such as PCA, at lower dimensions due to its ability to learn the highly nonlinear nature of the data. However, as the data dimensions increased, PCA's projection improved to retain enough information, reducing the accuracy gap between the two methods (Fournier and Aloise 2019). Fournier and Aloise (2019) observed that the computational time needed to train an AE was on average 2-4 times higher than that of PCA. They advised that PCA be used on larger data as its accuracy is comparable to AE (Fournier and Aloise 2019).

3.1 Latent Semantic Analysis (LSA)

LSA is a linear DRT which uses linear functions to transform HDD to lower dimensions (Ayesha et al. 2020). As an unsupervised technique, LSA does not require labelled data to transform HDD to lower dimensions. LSA utilizes singular value decomposition (SVD), a linear algebra technique, to reduce the dimensions while retaining the similarity structure of the documents or articles (Dumais 2005). The articles are typically compared using cosine similarity, where articles with high similarity have a value close to 1 and dissimilar articles have a value close to 0 (Dumais 2005). However, other methods such as Euclidean distance or Jaccard similarity can also be used for comparison purposes (Dumais 2005). It should be noted that in LSA, if the requested dimension level is larger than the number of samples the LSA will reduce the dimension to the size of the samples available (Dumais 2005).

In LSA, the number of features (i.e., columns) in the reduced data matrix is typically chosen to be less than or equal to the number of rows (i.e., samples) in the original data matrix. This is because LSA works by decomposing the original document-term matrix into a smaller set of "latent" topics or concepts, which are represented by the columns of the reduced matrix.

The SVD is used to decompose document-term matrix $A_{m,n}$ into 3 matrices: $U_{m,k}, \Sigma_{k,k}, V^T_{k,n}$ (Singla 2018). Both U and V^T are orthonormal matrices, meaning their column vectors have a length of 1 and the dot product

of any 2 column vectors equal 0 (Singla 2018). The document matrix U has documents as rows and latent topics as columns, capturing the relationship between the documents and latent topics (Singla 2018). The term matrix V^T has termed as columns and latent topics as rows, capturing the relationship between the terms and the latent topics (Singla 2018). The ordered diagonal matrix Σ provides the importance value for each latent topic in accurately approximating the original document-term matrix, the values ordered by importance (Singla 2018). The rank of matrix A is denoted by k, which represents the number of latent topics or dimensions a dataset can be reduced to (Singla 2018). Determining the value of k is the user's responsibility, and the aim of LSA is to find the optimal dimension to reduce to such that $k < n$ while minimizing the error between the original $A_{m,n}$ and the new approximation $A_{m,k}$ (Singla 2018). After dimension reduction, the document-term matrix $A_{m,k}$ has terms or words represented by latent topics, with a topic possibly having multiple associated terms. This transformed dataset is a significant drawback of LSA, as it can be hard to interpret.

SVD is derived as follows (Singla 2018):

$$
\begin{aligned}
A_{m,n} &= U\Sigma V^T T \\
&\quad V \text{ is the eigenvectors of } A^T A\text{:} \\
A^T A &= V\Sigma^2 V^T \\
&\quad U \text{ is the eigenvectors of } AA^T\text{:} \\
AA^T &= U\Sigma^2 U^T \\
\Sigma &= \sqrt{\Sigma^2} \\
A_{m,k} &\approx A_{m,n} \\
&\approx U_{m,k}\Sigma_{k,k}V^T_{k,n} \\
&\approx \sum_{i=1}^{k} u_i \sigma_i v_i \\
&\quad A_{m,k} \text{ is a good approximation of } A_{m,n} \text{ when:} \\
&\quad \min \|A_{m,n} - A_{m,k}\|
\end{aligned}
\tag{3}
$$

where,
$A_{m,n}$ = document-term matrix;
U = document matrix;
Σ = latent topics importance value;

V^T = term matrix;
m = number of terms;
n = number of documents;
k = number of latent topics

3.2 Linear Discriminant Analysis (LDA)

Like LSA, LDA is a linear DRT that uses linear transformations to reduce the dimensions of an HDD (Ayesha et al. 2020). However, unlike LSA, LDA is a supervised method that requires labelled data to accurately find linear boundaries that separate different categories (Ayesha et al. 2020). The goal of LDA is to project the HDD onto a lower-dimensional space with well-defined category boundaries in order to avoid overfitting (Raschka 2014). However, LDA may not perform well on non-Gaussian datasets since it assumes that the data follows a Gaussian distribution and that the covariance matrix is the same for all categories (Raschka 2014). Based of these assumptions, LDA computes the mean and covariance matrix for each category and seeks linear combinations of the features such that they maximize the equation: $S = \frac{\sigma^2_{between}}{\sigma^2_{within}}$ concerning the categories (Raschka 2014). These linear combinations also referred to as the discriminant functions, are used to transform the HDD to a lower dimension while preserving the class boundaries (Raschka 2014). Finally, the discriminant functions can be used to categorize new data points based on which category has the highest discriminant function values (Raschka 2014).

The discriminant function is derived as follows (Chen 2020):

$$
\begin{aligned}
a_i &= v^T x_i \\
\mu_j &= \frac{1}{n_j} \sum_{x_i \in C_j} a_i \\
&= \frac{1}{n_j} \sum_{x_i \in C_j} v^T x_i \\
\mu_j &= v^T m_j \\
\mu &= \frac{1}{n} \sum_{j=1}^{c} n_j \mu_j \\
&= v^T \frac{1}{n} \sum_{j=1}^{c} n_j m_j \\
\mu &= v^T m \qquad (4) \\
\sigma^2_{between} &= \sum_{j=1}^{c} n_j \left(\mu_j - \mu\right)^2 \\
\sigma^2_{between} &= v^T S_B v \\
\sigma^2_{within} &= \sum_{j=1}^{c} \sum_{x_i \in C_j} \left(a_i - \mu_j\right)^2 \\
&= v^T \sum_{j=1}^{c} S_j v \\
\sigma^2_{within} &= v^T S_W v \\
&\therefore \max_{v=k} \frac{\sigma^2_{between}}{\sigma^2_{within}}
\end{aligned}
$$

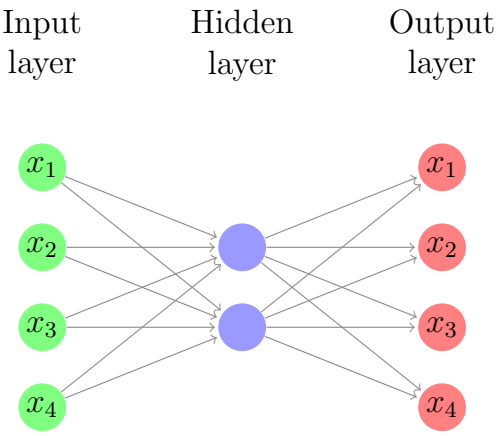
where,
a_i = discriminant function;
v = matrix with column vectors k;
x_i = a given data point;
k = number of vector columns in v;
n_j = number of data points from a given category j;
n = total number of data points;
c = total number of categories;
C_j = a given category j;

μ_j = mean of a given category;
μ = global mean;
$\sigma^2_{between}$ = distance category means are from the global mean;
σ^2_{with} = spread within each category;

The number of categories in the HDD determines the value of k in LDA (Hastie et al. 2017a). In the context of this paper, applying LDA to the dataset would result in reducing the original dimensionality from 24,134 to 4; $k = c - 1$. Each dimension (or category) is associated with a set of terms that are most closely related to that topic (Hastie et al. 2017a).

3.3 Auto-Encoder (AE)

The AE is an unsupervised neural network that can encode datasets effectively, thereby reducing their dimensions (Xie et al. 2019). It is comprised of three primary layers: an input layer, where the input data is received, a hidden layer, which compresses the dimensions of the original data to a specific dimension (i.e., encoding), and an output layer, where the original dataset is reconstructed from the encoding layer (i.e., decoding) (Xie et al. 2019). Unlike the other DRTs mentioned above, AE is able to learn non-linear relationship present in the data (Xie et al. 2019). Both the encoder and decoder layers are activated by an activation function that determines which nodes in the layers should be on or off, based on the input layer (Hastie et al. 2017b). Through the use of backpropagation AE learns to accurately map the input data to the output data, adjusting the weights and biases in the network to improve the mapping accuracy (Hastie et al. 2017b). Backpropagation computes the error between the predicted and actual output and uses this error to retrain and adjust the weights and biases in order to minimize the error (Hastie et al. 2017b). The neural network can learn complex relationships from data through this technique, however, it may consume a significant amount of computational resources and time (Hastie et al. 2017b).



The above diagram is of a typical auto-encoder architecture consisting of two parts: an encoder (input layer and hidden layer) and a decoder (hidden layer and output layer) (Xie et al. 2019). The encoder and decoder are derived as follows (Sigal 2017); (Stewart 2023):

$$\begin{aligned} e, d &= \min_{e,d} \|x - (e \circ d)x\|^2 \\ e &= \sigma(Wx + b) \\ d &= \hat{\sigma}(\hat{W}e + \hat{b}) \\ \hat{W} &= W^T \\ l(x, \hat{x}) &= \|x - \hat{x}\|^2 \\ &= \left\|x - \hat{\sigma}(\hat{W}(\sigma(Wx + b)) + \hat{b})\right\|^2 \\ \mathcal{L} &= (1/N) * sum(l(x_i, \hat{x}_i)) \\ &\min \mathcal{L} \end{aligned} \tag{5}$$

where,
e = encoder network;
d = decoder network;
W = weight matrix;
σ = an activation function;
$\hat{\sigma}$ = an activation function;
x = input data;
$\hat{x}$ = approximation of the input data;
l = loss function;

b = b is the bias vector;
$\mathcal{L}$ = total number of data points;

The weight matrix W captures the relevant patterns and relationships in the input data, according to Hastie et al. (2017). The number of rows in the matrix corresponds to the number of dimensions in the decoded output data, while the number of columns corresponds to the number of dimensions in the original input data (Hastie et al. 2017b). The bias vector allows each neuron to make predictions by adjusting the activation function left or right (Hastie et al. 2017b). AE utilizes backpropagation to iteratively minimize the average loss function $\mathcal{L}$ across the entire training set (Hastie et al. 2017b).

4 Experiment

Four metrics were used to compare the 3 DRTs: cosine similarity, which measures the degree to which the DRTs preserve the semantic structure of the original data; reconstruction error, which quantifies the difference between the original data and the reduced data via the mean squared error; classification accuracy, which evaluates the accuracy of a classifier trained on the reduced dataset generated by each DRT; and computational time, which assesses how long it takes for each DRT to generate the reduced dataset.

4.1 Design

Two commonly used text classifiers, Random Forest (RF) and Support Vector Machine (SVM), were employed during this experiment. As the focus of the paper was on DRTs, the classifiers' hyperparameters were not fine-tuned; they were left at their default settings as determined by the `sklearn` library. An important advantage of doing this was that the same classifier would be used for all the reduced datasets. To ensure reproducibility of the experiment, a random seed was set. A randomized stratified training and test set were used for each DRT trial run, with a 70/30 data split. As the categories were fairly balanced, stratified sampling was selected to ensure that the training and testing datasets reflect this balance. Baseline results were obtained using the untreated TF-IDF matrix. The DRTs were used to reduce the dimensions of the original data by 2%, 1%, and 0.1%, except for LDA which does not allow users to set the level of dimension reduction.

To account for this, an additional reduction dataset was created for each of the other DRTs with the same dimensions as the LDA reduced dataset. For consistency, the untreated TF-IDF was also partitioned in the same way as the reduced dataset generated by DRTs. The classifiers were then applied to these resulting datasets, and the four metrics mentioned earlier were used to evaluate the performance of the DRTs. The experiment was conducted on a PC with the following specifications: Intel(R) Core(TM) i7-7700 CPU at 3.60GHz processor, 16 GB of RAM, and Windows 10 Pro operating system. All experimental and EDA work was completed using `python` version 3.7.10 (Van Rossum and Drake 2009).

4.2 Results

TF-IDF, LSA, and LDA dimension reduction were performed using the `sklearn` library in `python`, while AE dimension reduction was implemented using the `keras` library (Pedregosa et al. 2011); (Chollet et al. 2015). The activation function for the AE was selected following best practices outlined by Goodfellow et al. (2017) in their book "Deep Learning." Specifically, the relu activation function was used for the encoder and sigmoid for the decoder. The batch size of 64 and epoch of 10 was selected based on the recommendations in the same book (Goodfellow et al. 2017).

4.2.1 Computational Time

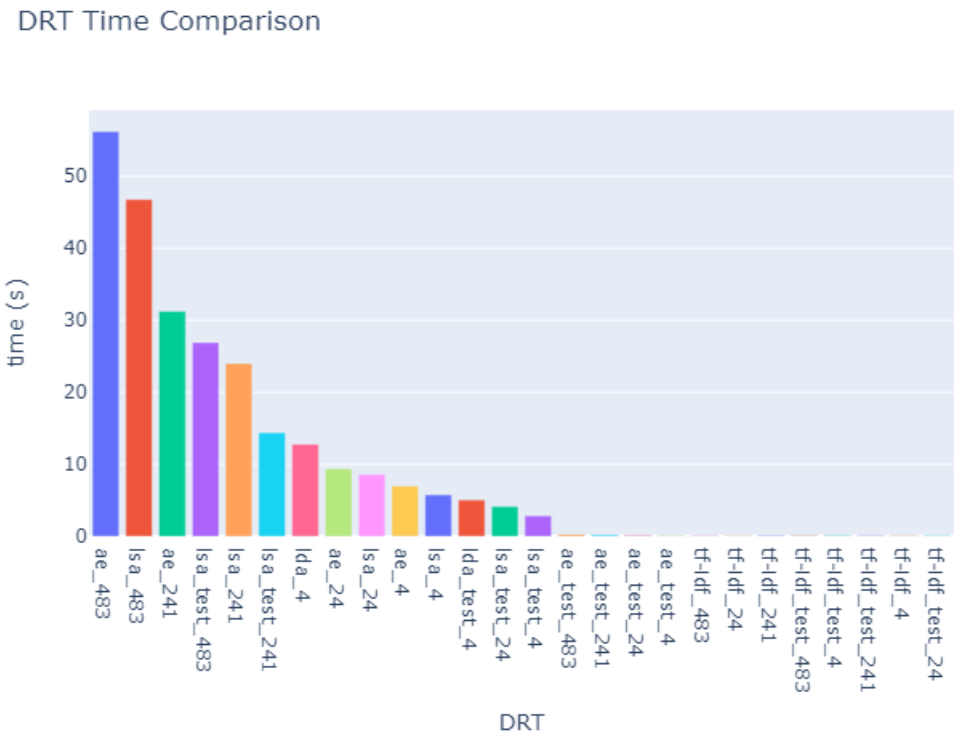


Figure 4: The computational time it took for each DRT to transform the train and test datasets.

The TF-IDF DRT method was very efficient, taking less than 1 second, approximately 0.1 second, to compute regardless of the dimension reduction size. In contrast, both AE and LSA were far more computationally demanding, with AE being more computationally expensive than LSA. The AE method took over 50 seconds to train and compress the original data by 2%, but once trained, it was the second fastest DRT to transform the test data after TF-IDF. On the other hand, LSA, on average, took the longest time to transform the train and test data, regardless of the dimension reduction level. Based solely on computation time, LDA was the most efficient DRT method, taking only 13 seconds to transform the train set and 5 seconds to transform the test set.

4.2.2 Cosine Similarity and Reconstruction Error

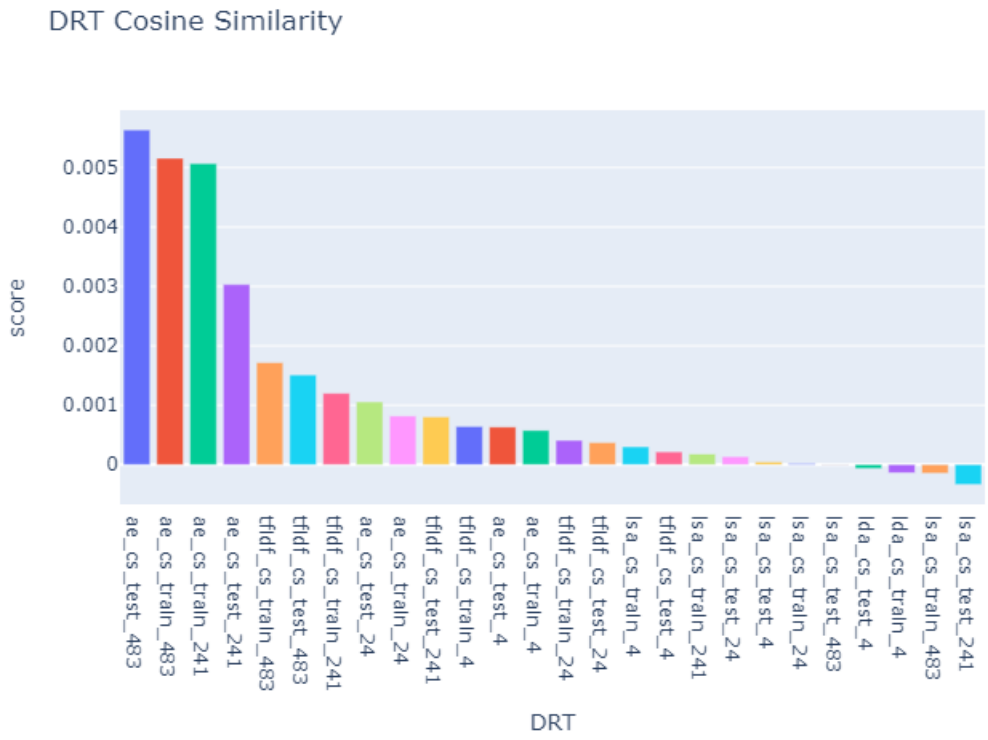


Figure 5: The cosine similarity between the train and test sets each for DRT.

Out of the 3 DRTs that were tested, the AE had the highest cosine similarity value when comparing the reduced training and test sets. The larger the dimension of the reduced datasets, the higher the cosine similarity. However, given the low magnitude of the cosine similarity values, it can be inferred that the reduced datasets have very little in common with the original data, which might affect their effectiveness during the classification task.

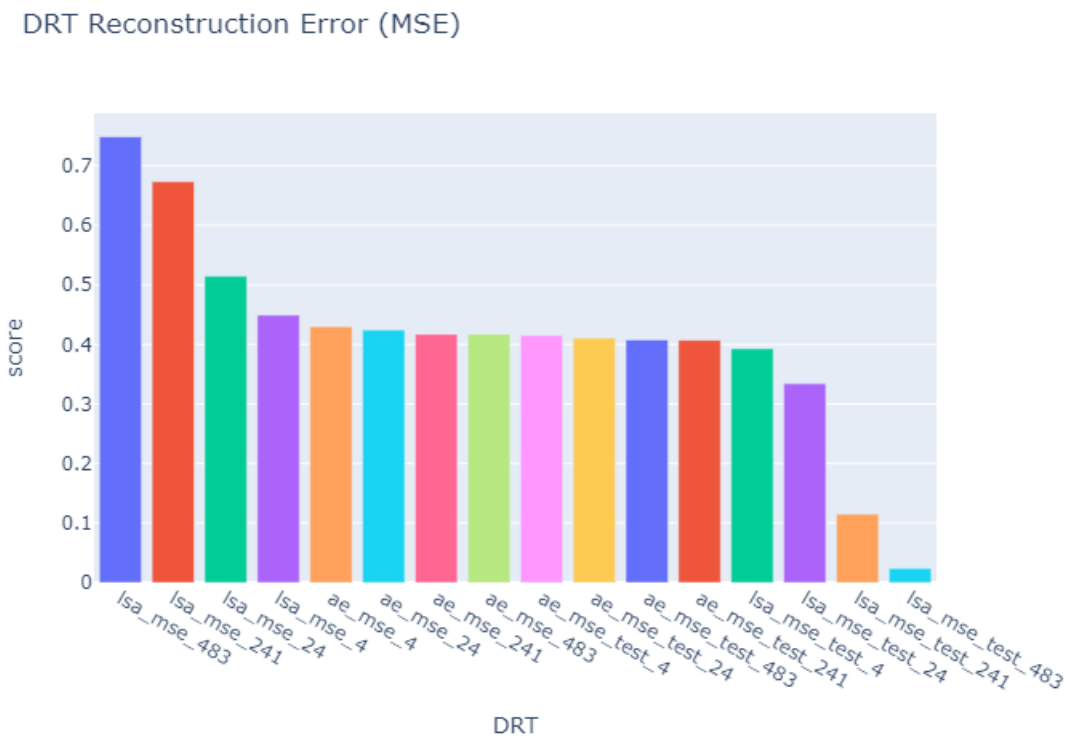


Figure 6: The reconstruction error for the train and test sets each for DRT.

LSA had the smallest and largest reconstruction error scores, depending on the dimension size and dataset used, while the AE's scores were consistent regardless of the dimension size. As the dimension size of LSA reduced, the training set error rate also decreased, but as the dimension size of LSA reduced, the test set error rate increased. This could be because reconstructing higher-dimensional data involves more relationships among the data points, resulting in more room for error. However, with fewer points present in the test data, there might be fewer informative relationships, and an increase in dimensions allows more informative relationships to be mapped. LDA along with TF-IDF could not have their reconstruction error calculated due to the way they transform the data.

4.2.3 Classification accuracy

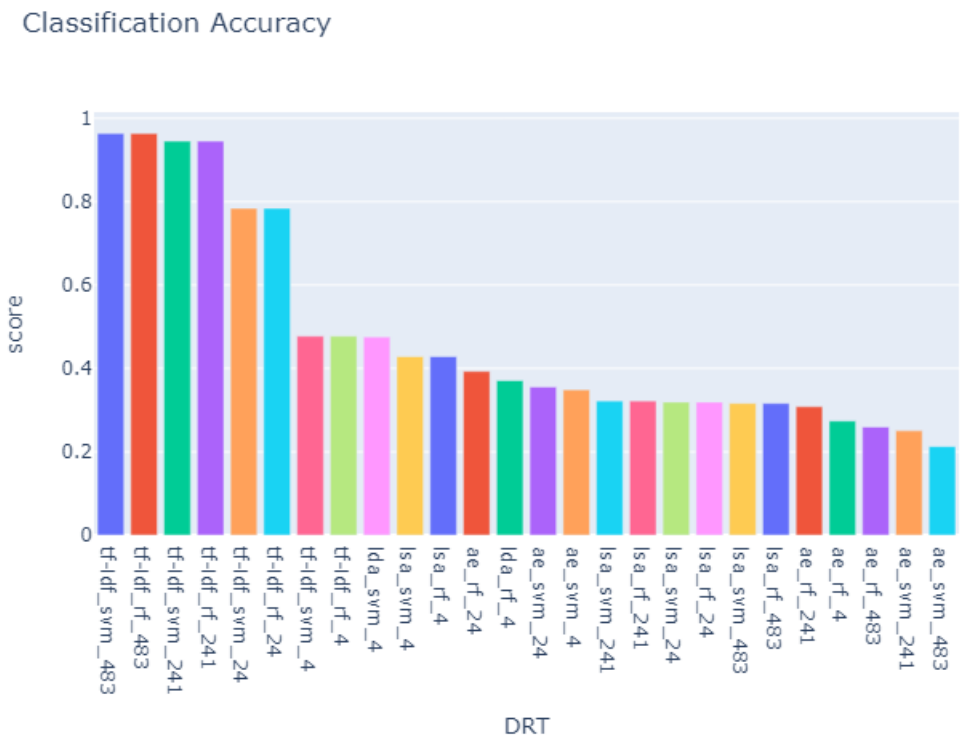


Figure 7: The classification accuracy on SVM and RF for each for DRT.

The feature selection method, TF-IDF, was the best reduction method based on classification accuracy. Nearly all the TF-IDF reduced data produced an accuracy above 75%, except for the 4-dimensional reduced dataset. The results of the TF-IDF were expected, as classification accuracy generally improves as dimensions increase. However, the drop in accuracy of those TF-IDF reduced datasets from 24 dimensions to 4 dimensions was significant. The next best DRT was LDA, with the 4-dimensional reduced dataset paired with an SVM classifier producing almost 50% accuracy. All the LSA-treated data produced an overall accuracy range of 30% to 40%, while AE-treated data yielded an overall accuracy range of 20% to 40%. The performance of all the DRTs tested, except for TF-IDF, was poor, likely due to the small size of the reduced datasets.

5 Conclusion

To summarize, the TF-IDF method proved to be the most efficient and effective reduction technique in terms of classification accuracy and computational time. The study explored three DRTs - LSA, LDA, and AE - and found that LSA was the best at compressing information, but its stability was a concern, while AE was the most stable DRT, producing consistent results in both classification accuracy and reconstruction error. Given the increasing advancements in NLP, these 3 DRTs showcase classical ways of handling overfitting during a text classification task. Although the above results may appear poor, it is worth noting that given the significant reduction in dimension, the DRTs worked fairly well in capturing enough information from the original dataset to yield an accuracy range of 20-50%. The range of the reduction explored was based solely on the limitation of the PC used. The DRTs cannot replace proper data processing for the task at hand, but they can be useful in reducing noise and computation time, as demonstrated by the performance of TF-IDF treated models.

Future research could investigate the optimal dimension levels for each DRT using GPUs or cloud services and explore the relationship between DRT and reconstruction error. In general, there are several intersections between NLP and DRT that warrant further investigation.

References

AbuZeina, Dia, and Fawaz S Al-Anzi. 2018. "Employing fisher discriminant analysis for Arabic text classification." *Computers & Electrical Engineering* 66:474–486.

Akritidis, Leonidas, and Panayiotis Bozanis. 2022. "How Dimensionality Reduction Affects Sentiment Analysis NLP Tasks: An Experimental Study." In *Artificial Intelligence Applications and Innovations,* edited by Ilias Maglogiannis, Lazaros Iliadis, John Macintyre, and Paulo Cortez, 301–312. Cham: Springer International Publishing. ISBN: 978-3-031-08337-2.

Allison, Ben, David Guthrie, and Louise Guthrie. 2006. "Another Look at the Data Sparsity Problem." In *Text, Speech and Dialogue,* edited by Petr Sojka, Ivan Kopeček, and Karel Pala, 327–334. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN: 978-3-540-39091-6.

Ayesha, Shaeela, Muhammad K. Hanif, and Ramzan Talib. 2020. "Overview and comparative study of dimensionality reduction techniques for high dimensional data." *Information Fusion* 59 (July): 44–58. https://doi.org/10.1016/j.inffus.2020.01.005.

Bird, Steven, Ewan Klein, and Edward Loper. 2009a. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc."

———. 2009b. *Normalizing Text,* 107–108. " O'Reilly Media, Inc."

Buehlmann, Peter, and Sara van de Geer. 2013. "Preface." In *Statistics for high-dimensional data: Methods, theory and applications,* 8. Springer.

Chen, Guangliang. 2020. *Linear Discriminant Analysis (LDA).* https://www.sjsu.edu/faculty/guangliang.chen/Math253S20/lec11lda.pdf. PowerPoint presentation.

Chollet, Francois, et al. 2015. "Keras." https://github.com/fchollet/keras.

Diaz, Gene, and Arthit Suriyawongkul. 2020. *stopwordsiso.* https://pypi.org/project/stopwordsiso/.

Dumais, Susan T. 2005. "Latent semantic analysis." *Annual Review of Information Science and Technology* 38 (1): 188–230. https://doi.org/10.1002/aris.1440380105.

Eisenstein, Jacob. 2019a. "Introduction." In *Introduction to natural language processing,* 1–12. The MIT Press.

———. 2019b. "Linguistic applications of classification." In *Introduction to natural language processing,* 79–80. The MIT Press.

Fournier, Quentin, and Daniel Aloise. 2019. "Empirical Comparison between Autoencoders and Traditional Dimensionality Reduction Methods." In *2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE),* 211–214. https://doi.org/10.1109/AIKE.2019.00044.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. 2017. "Learning XOR." In *Deep learning,* 172–174. MIT Press.

Greene, Derek, and Pádraig Cunningham. 2006. "Practical Solutions to the Problem of Diagonal Dominance in Kernel Document Clustering." In *Proc. 23rd International Conference on Machine learning (ICML'06),* 377–384. ACM Press.

Hastie, Trevor, Jerome Friedman, and Robert Tisbshirani. 2017a. "Linear Discriminant Analysis." In *The elements of Statistical Learning: Data Mining, Inference, and prediction,* 106–119. Springer.

———. 2017b. "Neural Networks." In *The elements of Statistical Learning: Data Mining, Inference, and prediction,* 392–397. Springer.

Manning, Christopher D., Prabhakar Raghavan, and Hinrich Schütze. 2009a. "Scoring, term weighting and the vector space model." In *Introduction to information retrieval,* 117–119. Cambridge University Press.

———. 2009b. "Stemming and lemmatization." In *Introduction to information retrieval,* 32–35. Cambridge University Press.

———. 2009c. "The term vocabulary and postings lists." In *Introduction to information retrieval,* 27–29. Cambridge University Press.

McKinney, Wes, et al. 2010. "Data structures for statistical computing in python." In *Proceedings of the 9th Python in Science Conference,* 445:51–56. Austin, TX.

Oesper, Layla, Daniele Merico, Ruth Isserlin, and Gary D Bader. 2011. "WordCloud: a Cytoscape plugin to create a visual semantic summary of networks." *Source code for biology and medicine* 6 (1): 7.

Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. "Scikit-learn: Machine learning in Python." *Journal of machine learning research* 12 (Oct): 2825–2830.

Raschka, Sebastian. 2014. *Linear discriminant analysis,* August. https://sebastianraschka.com/Articles/2014_python_lda.html.

Sigal, Leonid. 2017. *Topics in AI (CPSC 532L): Multimodal Learning with Vision, Language and Sound.* https://www.cs.ubc.ca/~lsigal/532L/Lecture9.pdf. PowerPoint presentation.

Singla, Sumedha. 2018. *SVD, SVD applications to LSA, non-negative matrix factorizations.* https://people.cs.pitt.edu/~milos/courses/cs3750/lectures/class10.pdf. PowerPoint presentation.

Stewart, Matthew. 2023. *Comprehensive introduction to autoencoders,* February. https://towardsdatascience.com/generating-images-with-autoencoders-77fd3a8dd368.

Torkkola, Kari. 2001. "Linear discriminant analysis in document classification." In *IEEE ICDM workshop on text mining,* vol. 29.

Van Rossum, Guido, and Fred L. Drake. 2009. *Python 3 Reference Manual.* Scotts Valley, CA: CreateSpace. ISBN: 1441412697.

Xie, Lin, Genggeng Liu, and Hongfei Lian. 2019. "Deep variational autoencoder for text classification." In *2019 IEEE International Conference on Industrial Cyber Physical Systems (ICPS),* 737–742. https://doi.org/10.1109/ICPHYS.2019.8780129.