UNSW
SYDNEY

Author Identification with Deep Learning

COMP9417 2023 T1 - Group Project: Team Spirit

Duke Nguyen
z5398432

Weixian Qian
z5408671

Lavender Kong
z5271686

Jaeff Hong
z5316653

May 25, 2023

Contents

1 Introduction

'Authorship Identification' is the task of identifying the author of a text, given a set of candidate authors [Mohsen et al., 2016]. It is a multi-class single-label text classification task, where texts or features extracted from texts are features, and the author is the class label [Mohsen et al., 2016]. Authorship identification can be used to identify the works of authors writing under pseudonyms. Historically, non-machine learning methods have been used to identify the authors of the Federalist Papers, for this purpose. It can be used in a variety of fields, including textual criticsm, information security, forensics, etc. [Romanov et al., 2020]. Previous works in this area, for example, include [Mohsen et al., 2016, Romanov et al., 2020, Sarwar et al., 2018]. Until 2016, this problem has usually been tackled using classical supervised learning models. Afterwards, deep learning models have been used. However, implementations of SOTA models like pre-trained BERT or BERT derivatives have been scarce or non-existent. In this projec, we will be using deep learning models on the English SPGC (Standardized Project Gutenberg Corpus)[Gerlach and Font-Clos, 2020] under the GNU General Public License, specifically we will be using SVM with SDAE and BERT.

2 Literature Review

Previous works done in author identification mainly tackle two issues, the extraction of features of texts to represent writing styles of different authors and methods to correctly predict the author, given text.

One such work [Mohsen et al., 2016] that went with a deep learning approach, utilized character n-grams in conjunction with Stacked Denoising AutoEncoders (SDAE) for feature extraction on Reuters Corpus Volume 1 (RCV1). The SDAE was then fed with the produced variable size n-gram feature set and the extracted features were used as input for a support vector machine (SVM) classifier. The result was a model that could outperform previous state-of-the-art author identification (95.12% accuracy).

Another work [Romanov et al., 2020] compared usage of Support Vector Machine (SVM) and modern classification methods based on deep learning Neural Networks (NNs) architectures, namely Long Short-Term Memory (LSTM), Convolutional neural networks (CNN) with attention and transformers. The corpus was collected from the Moshkov library(500 Russian authors). Obtained results from comparing between datasets of 2, 5, 10 and 50 candidate authors showed that the approach based on SVM demonstrated superior accuracy to modern deep NNs architectures, having an accuracy by more than 10% on average.

Bidirectional Encoder Representations from Transformers (BERT) [Devlin et al., 2018] is a pre-trained deep learning model which can be fine-tuned to create state-of-the-art models for a wide range of tasks, such as text generations and text classifications.

3 Methodology

3.1 Dataset

In 2020, the Standardized Project Gutenberg Corpus was released under GNU General Public License, which standardizes the free and open source Gutenberg texts for the first time. Previous works also have

also not extensively used the Gutenberg Corpus. It is for these reasons that we will be using this corpus for our training, [Gutenberg, 2023]. For ease of access, we will be using the readily available SPGC-2018-07-18 version available at [Gerlach and Francesc, 2008] which contains pre-processed data on 55905 books. There are three zipped files respectively to be downloaded, namely: SPGC-counts-2018-07-18.zip (the counts file), SPGC-metadata-2018-07-18.csv (the metadata file), and SPGC-tokens-2018-07-18.zip (the tokens file).

3.2 Exploratory Data Analysis

Once all the downloaded files are unzipped, we have two folders: counts, and tokens, and the csv metadata file. The counts folder contain all the txt counts files which are the token counts in dictionary format of each text. The tokens folder contain all the txt tokens files which are the lower-cased prepossessed text of each document. The csv metadata file contains a table detailing all the information for each document, i.e. id, title, author, year of birth of author (authoryearofbirth), year of death of author (authoryearofdeath), list of languages of the document (language), number of downloads (downloads), list of subjects (subjects), and the data type of the entry (type).

We can see the concise summary in Table 1a, and the first few and the last few entries of the metadata CSV in Table 1b. We can also see that this corpus also includes data types other than 'text' entries, for example, 'sound' also contribute to a minor amount, as shown in Figure 1a. The dataset contains documents in a variety of languages and subsets of languages, which can be observed from Figure 1b. English, by itself, however, is still the overwhelming majority subset of the entries. For the purpose of this project, we are only going to be using documents in English and having the data type as 'text'.

```
Index: 57713 entries, PG0 to PG999999
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   title              57642 non-null  object
 1   author             55451 non-null  object
 2   authoryearofbirth  42946 non-null  float64
 3   authoryearofdeath  41850 non-null  float64
 4   language           57711 non-null  object
 5   downloads          57711 non-null  float64
 6   subjects           57713 non-null  object
 7   type               57713 non-null  object
dtypes: float64(3), object(5)
```

(a) Concise Summary of Metadata CSV

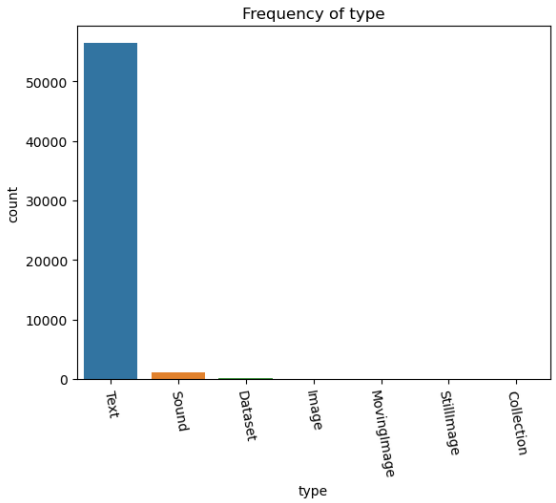
id	title	author	authoryearofbirth	authoryearofdeath	language	downloads	subjects	type
PG0	NaN	NaN	NaN	NaN	NaN	NaN	set()	Text
PG1	The Declaration of Independence of the United ...	Jefferson, Thomas	1743.0	1826.0	['en']	604.0	{'United States -- History -- Revolution, 1775...	Text
PG2	The United States Bill of Rights: The Ten Orig...	United States	NaN	NaN	['en']	158.0	{'Civil rights -- United States -- Sources', '...	Text
PG3	John F. Kennedy's Inaugural Address	Kennedy, John F. (John Fitzgerald)	1917.0	1963.0	['en']	28.0	{'Presidents -- United States -- Inaugural add...	Text
PG4	Lincoln's Gettysburg Address: Given November 1...	Lincoln, Abraham	1809.0	1865.0	['en']	55.0	{'Consecration of cemeteries -- Pennsylvania -...	Text
...	...	...	...	...	...	...	...	...
PG57710	A Son of the State	Ridge, W. Pett (William Pett)	NaN	1930.0	['en']	0.0	set()	Text
PG57711	Hudson Tercentenary: An historical retrospect ...	Chamberlain, Frank	NaN	NaN	['en']	0.0	set()	Text
PG57712	Proses moroses	Gourmont, Remy de	1858.0	1915.0	['fr']	0.0	set()	Text
PG57713	The Animal Parasites of Man	Theobald, F. V.	NaN	NaN	['en']	0.0	set()	Text
PG999999	Piccole anime	NaN	NaN	NaN	NaN	NaN	set()	Text

(b) Sample Entries of Metadata CSV

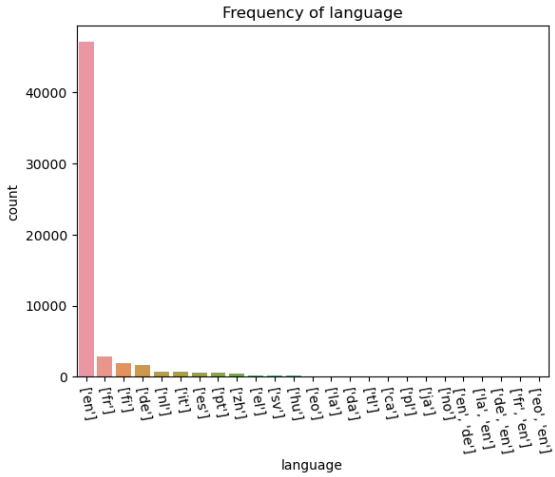
Table 1: Metadata Stats

3.2.1 Limitations

Original vs. Translated Works Upon examining this English corpus, we find a few limitations. First of all, there are no columns which specify whether the work is an original or translated one. And, in case of it being a translated worked, there is no record of the translator, only of the original author. We can see an example of this in Table 2, where Nietzsche is a German author, and 'Thus Spake Zarathustra: A Book for All and None' was originally written in German, however, the language column designates it



(a) Data Type Frequency



(b) Language Type Frequency

Figure 1: Column Frequency

as 'English', and there is no mention of the translator. Different authors will have different translation styles, and different choices of word. Although we find no instance of different translations of the same work, multiple works by the same author can have different translators, leading to different distribution of words, especially if we feed the text to a CountVectorizer. We can't simply remove all the translated works either because there is no way to tell whether a work is original or translated.

id	title	author	authoryearofbirth	authoryearofdeath	language	downloads	subjects	type
PG1998	Thus Spake Zarathustra: A Book for All and None	Nietzsche, Friedrich Wilhelm	1844.0	1900.0	['en']	3742.0	['Philosophy, German', 'Superman (Philosophica...	Text

Table 2: Translated Works

Multiple-Author Works Another serious limitation of the dataset is that it designates a single author to every work, even when a work has multiple authors. We show two examples of this in Table 3. In the first example there are two entries of 'The Federalist Papers' with the author being only 'Jay, John', even though it was authored by two more people which are 'Hamilton, Alexander', and 'Madison, James'. If we access the Gutenberg sites, of both of these works, we can see that all of the respective authors are mentioned [Gutenberg, 2021] [Gutenberg, 2005]. This implies that the pipeline to crawl the Gutenberg works has a bug which only preserves a single author for every single entry. This would force the model to predict against classifying one of the correct authors of a work as its author. This is a serious issue in the dataset, and we have no easy fix at the moment to remedy the problem.

id	title	author	authoryearofbirth	authoryearofdeath	idx	num_tokens	num_unique	text
PG2225	"Captains Courageous": A Story of the Grand Banks	Kipling, Rudyard	1865.0	1936.0	1826	48548	4801	a man of honor by george cary eggleston illust...
PG2186	"Captains Courageous": A Story of the Grand Banks	Kipling, Rudyard	1865.0	1936.0	1817	62578	6705	http from page images generously made availabl...
PG43302	"The Kingdom of God Is Within You": Christiani...	Tolstoy, Leo, graf	1828.0	1910.0	24994	0	0	None
PG4602	"The Kingdom of God Is Within You": Christiani...	Tolstoy, Leo, graf	1828.0	1910.0	3853	15057	2873	transcriber notes every effort has been made t...
PG25493	A Cathedral Courtship	Wiggin, Kate Douglas Smith	1856.0	1923.0	15482	0	0	None
...	...	...	...	...	...	...	...	...
PG2034	Waverley; Or, 'Tis Sixty Years Since	Scott, Walter	1771.0	1832.0	1689	16704	5487	online distributed proofreading team at http e...
PG2782	Wilhelm Tell	Schiller, Friedrich	1759.0	1805.0	2299	25282	8114	scanned images of public domain material from ...
PG6788	Wilhelm Tell	Schiller, Friedrich	1759.0	1805.0	5593	50364	11058	http from page images generously made availabl...
PG19154	With Lee in Virginia: A Story of the American ...	Henty, G. A. (George Alfred)	1832.0	1902.0	12144	0	0	None
PG2805	With Lee in Virginia: A Story of the American ...	Henty, G. A. (George Alfred)	1832.0	1902.0	2319	56507	10250	available by internet archive http note projec...

Table 4: Duplicated Works

id	title	author	authoryearofbirth	authoryearofdeath	language	downloads	subjects	type
PG18	The Federalist Papers	Jay, John	1745.0	1829.0	['en']	461.0	{'Constitutional history -- United States -- S...	Text
PG61	The Communist Manifesto	Engels, Friedrich	1820.0	1895.0	['en']	2608.0	{'Socialism', 'Communism'}	Text
PG1404	The Federalist Papers	Jay, John	1745.0	1829.0	['en']	2670.0	{'Constitutional history -- United States -- S...	Text

Table 3: Multiple Author Works

Work with Unavailable Text In addition, there are also inconsistencies between the metadata CSV and the tokens and the counts files. Specifically, there are some entries in the metadata CSV that are not present in the tokens, and counts file, or those files are empty space. We simply need to do pre-processing here and remove all invalid entries. An example of a few invalid entries are as follow:

Error at PG2357: list index out of range
Error at PG5192: list index out of range
Error at PG8700: list index out of range

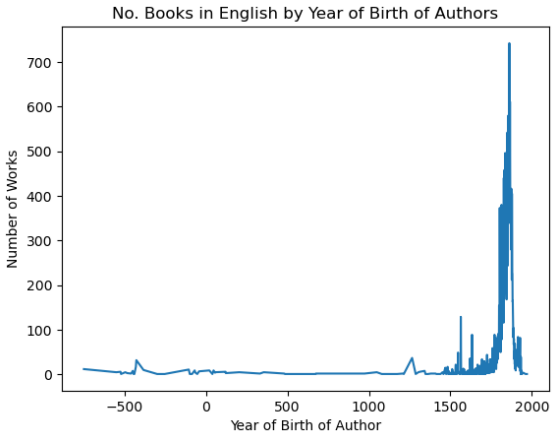
During this process, we also load all the text and their respective number of tokens (num_tokens), and number of unique tokens (num_unique) as calculated from their counts file as additional columns into the Dataframe for ease of analysis afterwards. We also remove works whose authors do not have a valid year of birth or year of death.

Duplicated Works Furthermore, we also find duplicated works (see Table 4, where the same work, with the same time, author, authoryearofbirth, authoryearofdeath, but with different pg_id and text. These works could be sourced from different distributors, therefore having additional preface section, and so on. There are also duplicated works with no corresponding text due to being invalid entries. We simply keep the one with the longest text (highest number of num_tokens), and remove the others.

After all these preprocessing, we end up with 31031 valid entries. The code in this section which corresponds to '1. Preprocessing' in EDA.py, can be used with newer version of SPGC, and a pre-processed version of the SPGC will be produced.

3.2.2 Distribution of Works and Authors

When we analyse the year of birth of all authors, we find that they center at 1822 (See Table 2b, and Figure 4a). If we analyse them by decade, we have the distribution as shown in Table 4b. We find that the majority of the author is born between 1822 and 1866. This means that most works are skewed towards having 19th century vocabulary and expression. This is reflected in Figure 2a. When analysing number of works per author, we find that most authors write under 50 works (see 5a). We also find that most documents are below 10,000 tokens, and the number of unique tokens are below 5,000 (see 6. The most frequent words can be displayed on the Word Cloud on Figure 5b.



(a) Works by Author's Year of Birth

	authoryearofbirth	authoryearofdeath
count	31031.000000	31031.000000
mean	1822.548226	1892.970610
std	177.131728	178.230488
min	-750.000000	-650.000000
25%	1822.000000	1893.000000
50%	1850.000000	1918.000000
75%	1866.000000	1940.000000
max	1970.000000	2015.000000

(b) Author's Year of Birth and Year of Death Statistics

Figure 2: Authors Stats

3.3 Data Pre-Processing

The corpus used in training the models is a subset of the original data. More specifically, only documents that were in English and in addition, the documents that were classified as 'text' were selected. From this reduced corpus, only 30 authors had over 50 documents, therefore, these 30 authors were chosen with each having 50 documents, for a total of 1500 texts.

From this, three dataset were created, the dataset that the SVM used was randomly divided into the following 3 sets:

1. Training Set: 60% of dataset
2. Test Set: 30% of dataset
3. Validation Set: 10% of dataset

Each set contains the 30 authors, with the training set containing 30 documents each for a total of 900 docs, the test set containing 15 documents each for a total of 450 docs and the validation set containing 5 documents each for a total of 150 docs.

The second dataset was used to train both the BERT and SVM model. This dataset had to split the documents into 2000 x 512 token chunks per author due to the BERT model only being able to process 512 tokens at a time. In essence, instead of 50 documents per author, the original dataset had to be converted into 2000 x 512 tokens per author and this dataset was randomly divided into the following 3 sets:

1. Training Set: 80% of dataset
2. Test Set: 10% of dataset
3. Validation Set: 10% of dataset

We also used the same split on a smaller dataset(1000×512 tokens), trained on BERT and made a report.

3.4 Model

In this section, we elaborate two deep learning models in our experiment: the article level NN (SDAE SVM), and the sentence level NN (BERT [Devlin et al., 2018]).

3.4.1 SDAE SVM Model

Since SVM is a popular approach to tackle author identification tasks, we follow Mohsen et al. [2016] which utilizes character n-grams as features and uses a Stacked Denoising Autoencoder to further enrich these features with an SVM classifier to make final predictions.

The tokenized texts are first converted into a variable length character n-gram feature set of length 1-5 inclusive, where the count of each character n-gram is counted to form as initial features. Feature Selection is used, namely frequency-based feature selection where the top 10000 occurring n-grams are selected and the rest are removed. The sets are then normalized using a min-max normalization and the normalized features are then passed into a stacked denoising autoencoder (SDAE) for further enrichment of features.

A DAE injects artificial noise in the input and attempts to output the original input by recognizing noisy data. The training of a DAE is as follows:

1. Bionomial noise is injected into data x, so that it gets transformed into $\bar{x}$. This means a random selection of elements in the feature set are set to 0.
2. $\bar{x}$ is encoded into the hidden representation y using a non-linear transformation function σ.

$$y = \sigma(W\bar{x} + b) \tag{1}$$

   where W is the weight matrix of the encoding layer, b is the bias and σ is the sigmoid function,

3. The hidden representation y is decoded into the reconstructed output z.

$$z = \sigma(W'y + b') \tag{2}$$

   where W' is the weight matrix of the decoding layer layer. Tied weights are used, therefore $W' = W^T$. The sigmoid function is used again.

4. The output z must reconstruct the original input x without noise. The reconstruction error is the cost function where binary cross-entropy was used and further, the adam optimizer was used.

The training of a SDAE consists of 2 procedures, unsupervised pre-training and then supervised fine-tuning. In pre-training, the steps are as follows:

1. For the k^{th} DAE, the above training steps are following to obtain the encoder function f_θ^k.
2. The encoder function f_θ^k is applied on the clean input x^k which is fed into the next DAE as input where

$$x^{k+1} = f_\theta^k(x^k). \tag{3}$$

3. The above 2 steps are repeated for each DAE, for $k \in [0, L]$ in the stack where, x^1 is the original input and the final x^L is the encoded features. We pre-train the SDAE for 20 epochs with batch size of 1 as this gave the best performance.

An illustration of this pre-training can be found in the appendix 10.

Next, the DAE's weights are all fine-tuned. The fine-tuning steps are as follows:

1. The encoders from each DAE are stacked and a logistic regression layer is added on top as shown in 11.
2. The encoders have already been pre-trained following the above steps and the learnt weights are used to initialize this network.
3. The fine-tuning is supervised, therefore labelled data is used to train the network and finetune the weights and parameters using categorical cross entropy as the cost function to minimize using back-propagation. Further, the logistic layer uses the softmax activation due to the classification nature of this problem and the adam optimizer is used. We fine-tune the network for 30 epochs using batch size of 1 as this gave the best performance.

The DAE is initialized with 1000 units of hidden representation, of which the final more richer encoding function is obtained after pretraining and fine-tuning. This can now be fed to a linear SVM classifier with an internal 5-fold cross validation using default parameters.

3.4.2 BERT Model

Bidirectional Encoder Representations from Transformers (BERT) [Devlin et al., 2018] is a masked language model. We will use the HuggingFace pre-trained BERT and finetune it to predict the author on several paragraphs, which are chopped from an article.

The basic structure of BERT is as follows:

$$\begin{aligned} H_i^0 &= W_w \cdot X_i + W_p \cdot X_p \\ H_i^{'} &= T(H_i) \\ T(X) &= Norm \circ Gelu \circ QKV(X) \end{aligned} \tag{4}$$

$$Norm(x) = \frac{x - \mu}{\sqrt{\sigma}} \tag{5}$$

Where X_i denotes encoded input words, where words are transformed into vectors using BertTokenizer, X_p is words positions, W_w, W_p represent the word embedding and position embedding. H_i is an encoder layer in BERT, $T()$ represents a transformer [Vaswani et al., 2017], where $Norm$ is used to normalize inputs in EQ.(5), and $Gelu$ [Hendrycks and Gimpel, 2016] is an activation function, which can be thought of as a smoother ReLU. $QKV()$ is represented as QKV attention mechanism.

$$QKV(X) = softmax(\frac{QX \cdot KX^T}{\sqrt{d_k}})VX \tag{6}$$

where Q, K, V are all linear layers to X, and $\sqrt{d_k}$ is represented as the degree of K dimension to control the scale of dot product. The model structure is shown in Figure 3.

To finetune BERT, we have four different set ups. The first three set-ups add last four encoder layers: Max-Pooling, Mean-Pooling, and Concat-Pooling. We also have Default-Pooling which only use last encoder layer. In Max-Pooling, we maximize each element in all four encoder layers. And in Mean-Pooling, we calculate the mean of them. In Concat-pooling, we simply concat the last four layers. A linear layer is used to fit the dimension of the layer from each of the strategy above to the number of label. Similar to SDAE SVM, we use cross entroy with softmax input to calculate loss. Hyperparameters are specified in [Popel and Bojar, 2018]'s work, using 24 batch size (for both training and validatoin) and $2e-05$ learning rate. The token is BERT's maximum token length which is 512. We also train the models within 10 epochs, since we find that that's when they converge to a point that is almost minimum as shown on Figure 9a.

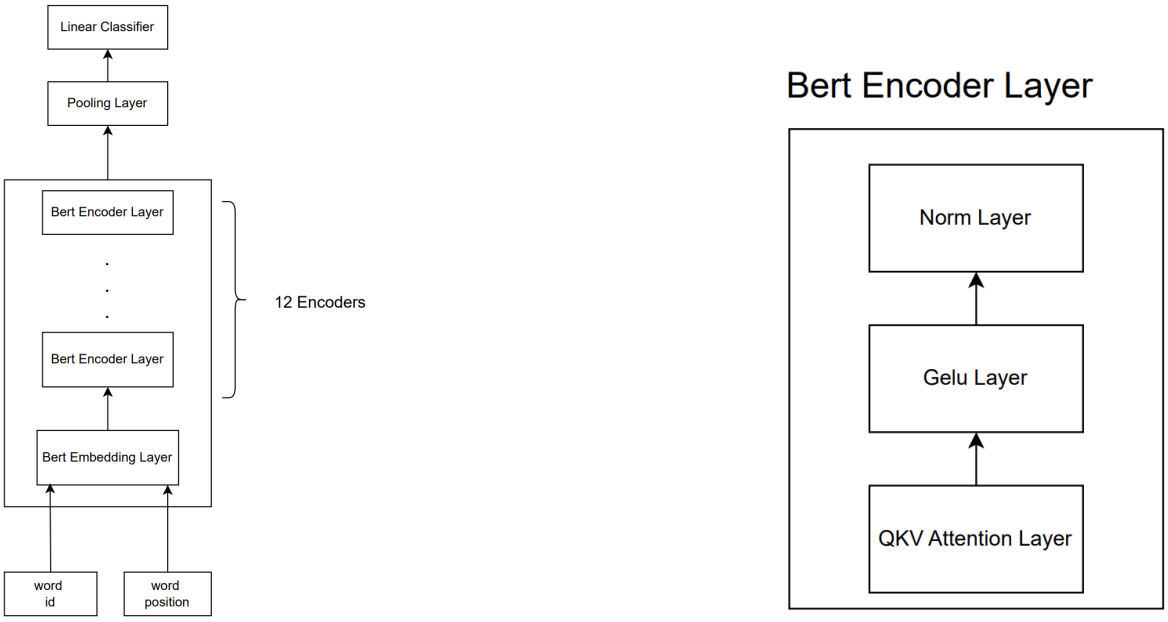
3.5 Metrics

We will be using the standard multi-class classification metrics for the comparison of our models: Cohen Kappa, Matthews Correlation Coefficient (MCC), micro-accuracy, macro-average precision, maco-average recall, macro-average F1-score. In addition, we will also display weighted average precision, weighted average recall, and weighted average F1-score for those who are interested. Cohen Kappa (a number between -1 and 1) measures inter-rate reliability, where values above 0.8 represents good agreement, and numbers approaching -1 representing no agreement. MCC (a number between somewhere between [-1, 0] and +1) takes into account true and false positives and negatives to measure classification, with a value closer to +1 being a perfect prediction. The rest are common classification metrics. We will also use Log Loss to compare between BERT models since they don't output a prediction but a probability value for each entry.

4 Results

When using the whole document dataset, SVM performs the best with the 1-layer SDAE SVM (or 1 DAE), across all measured metrics. This is in contrast to findings in Mohsen et al. [2016] which stated that SDAE SVM performs best with 3 DAEs. The metrics decreased across the board when the number of DAEs are increased from 1 to 2, and 2 to 3. All SDAE SVM models, however, still outperform the raw SVM.

Interestingly, when using the dataset containing 2000 x 512 token chunks per author, the 2-layer SDAE SVM outperforms the others across the board. Further, between the highest f1-score achieved from using



(a) Bert Layer Structure

(b) Bert Encoder Layer Structure

Figure 3: Bert Structure

the plain character-ngram feature set (Raw SVM column) and encoded character-ngram feature set (x-layer SDAE SVM column) from the table, we find the the 2000 x 512 token dataset only increases by 0.5% whilst the whole document increases by 14%. We can also see that the 2000 x 512 token dataset achieves a relatively high f1-score, despite only using plain character-ngrams with a 92.8% accuracy.

	cohen_kappa	matthews_corrcoef	micro-accuracy	macro avg precision	macro avg recall	macro avg f1-score	weighted avg precision	weighted avg recall	weighted avg f1-score
Raw SVM	0.819071	0.821772	0.825112	0.881030	0.824921	0.837734	0.880422	0.825112	0.837551
1-layer SDAE SVM	0.962888	0.962963	0.964126	0.966423	0.964127	0.964198	0.966271	0.964126	0.964120
2-layers SDAE SVM	0.939691	0.939941	0.941704	0.948037	0.941429	0.941132	0.947720	0.941704	0.941138
3-layers SDAE SVM	0.935053	0.935272	0.937220	0.943550	0.936825	0.936844	0.943503	0.937220	0.937043

Table 5: Results of SVM on whole document dataset

	cohen_kappa	matthews_corrcoef	micro-accuracy	macro avg precision	macro avg recall	macro avg f1-score	weighted avg precision	weighted avg recall	weighted avg f1-score
Raw SVM	0.926552	0.926564	0.929000	0.929046	0.929000	0.928849	0.929046	0.929000	0.928849
1-layer SDAE SVM	0.920172	0.920192	0.922833	0.922889	0.922833	0.922583	0.922889	0.922833	0.922583
2-layers SDAE SVM	0.931207	0.931230	0.933500	0.933783	0.933500	0.933304	0.933783	0.933500	0.933304
3-layers SDAE SVM	0.926552	0.926573	0.929000	0.929107	0.929000	0.928746	0.929107	0.929000	0.928746

Table 6: Results of SVM on 2000 x 512 token dataset

We find small dataset like 1000 words per paragraph, performs poorly on BERT [Devlin et al., 2018] in

Table 7. Inspired by [Sun et al., 2019], we validated that fine tuning last four encoder layers has a consistent performance compared with fine tuning all twelve layers. We applied different pooling strategies on BERT and concluded that BERT-Max pooling has the best metrics compared with other pooling strategies. However, we also noted that BERT-Max pooling had a high log loss and had a lower micro-accuracy than BERT-Default pooling in Table 8.

Dataset scale	Test accuracy
1000 paragraphs per author, 30000 in total	61%
2000 paragraphs per author, 60000 in total	90%

Table 7: Results of BERT on different sizes of token dataset

model	cohen_kappa	matthews_corrcoef	log_loss	micro-accuracy	macro avg precision	macro avg recall	macro avg f1-score	weighted avg precision	weighted avg recall	weighted avg f1-score
Max-Pooling BERT	0.903966	0.904197	2.698977	0.907167	0.913532	0.907167	0.906943	0.913532	0.907167	0.906943
Mean-Pooling BERT	0.896552	0.896801	2.319134	0.900000	0.906794	0.900000	0.899752	0.906794	0.900000	0.899752
Default-Pooling BERT	0.898103	2.424897	0.901731	0.909204	0.901500	0.898372	0.901500	0.901731	0.909204	0.901500
Concat-Pooling BERT	0.876724	2.636909	0.880180	0.891401	0.880833	0.877122	0.880833	0.880180	0.891401	0.880833

Table 8: Results of BERT on different patterns

5 Discussion

We find that among the SDAE SVM models, 1-layer SDAE SVM performs the best when being trained with the whole document dataset. On the 2000 paragraphs per author dataset, 2-layer SDAE SVM performs the best among all the models (SVM and BERT), although among BERT models, Max-Pooling has the best overall metrics.

5.1 Limitation

Bert model can only process 512 tokens per input at a time. This limitation forces us to split documents into 512 length tokens which corrupts the coherence of the book and hinders the model from capturing the overall writing style. Another limitation for BERT is that because the input data from SPGC is preprocessed, we do not have access to the original cases of the words, and punctuations, which could improve BERT's performance.

For the SDAE SVM model, we expect more number of layers of DAE to increase the SVM performance. However, when we increase the number of DAE to more than 1, the loss is significantly higher compared to SDAE with 1 layer. As the number of DAE increases to 3, the loss remains at a high value even after 50 epochs. This means that as the number of autoencoders increase, the SDAE fails to learn higher-order features from the dataset. This can be due to a variety of reasons, including a suboptimal batch size or learning rate. It could also be due to the fact that the dataset has limitations stated in section 3.2.1.

5.2 Future Work

We need to fix the pipeline of author extraction for SPGC, as well as adding additional crawling capability to differentiate between original and translated work, as well as recording the name of the translated author in order to provide an accurate and detailed dataset.

In terms of BERT, there are several ways to develop our own BERT variant. We can simply enlarge the token size per input of BERT, also to maintain the same number of parameters, we can prune the number of encoder layers, we can evaluate the performance in different settings. Another potential idea is adding a paragraph position embedding to BERT, so BERT can capture their order in separate paragraphs. We also want to exploit hierarchical attention [Yang et al., 2016] which can capture sentence level attention in a document on Bert.

Using one of the large language models would be interesting to see its performance on author identification. Namely, Facebooks LLaMA model [Touvron et al., 2023] recently released this year in 4 different parameter sizes of 7, 13, 33, 65 billion parameters. Its 13 billion parameter model outperforms GPT-3 on several benchmarks, despite its significantly lower parameters where GPT-3 contains 175 billion parameters. 7 billion parameters is still a large amount, therefore the LoRA [Hu et al., 2021] method can be used which can reduce the number of trainable parameters by 10,000 times and the GPU memory requirement by 3 times, to fine-tune LLaMA to the task of authorship identification.

The preprocessed data only contained 30 authors with over 50 documents. In future works, a larger corpora of data could potentially increase accuracy and more authors could be incorporated into predictions.

6 Conclusion

In this project, we compared different SVM and BERT models on the task of authorship identification. We demonstrate that deep learning methods based on SDAE and BERT can be successfully applied in author identification. Further, we show that despite SVM being an old supervised learning model, it still outperforms BERT on the task of author identification.

References

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

M. Gerlach and F. Font-Clos. A standardized project gutenberg corpus for statistical analysis of natural language and quantitative linguistics. *Entropy*, 22(1):126, 2020.

M. Gerlach and F.-C. Francesc. Standardized Project Gutenberg Corpus. `https://zenodo.org/record/2422561#.ZDjHg3ZByUk`, 2008. [Online; accessed 7-April-2023].

P. Gutenberg. The Communist Manifesto Entry. `https://www.gutenberg.org/ebooks/61`, 2005. [Online; accessed 7-April-2023].

P. Gutenberg. The Federalist Papers Entry. https://www.gutenberg.org/ebooks/1404, 2021. [Online; accessed 7-April-2023].

P. Gutenberg. Gutenberg, 2023. URL https://www.gutenberg.org/.

D. Hendrycks and K. Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

A. M. Mohsen, N. M. El-Makky, and N. Ghanem. Author identification using deep learning. In *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 898–903. IEEE, 2016.

M. Popel and O. Bojar. Training tips for the transformer model. *arXiv preprint arXiv:1804.00247*, 2018.

A. Romanov, A. Kurtukova, A. Shelupanov, A. Fedotova, and V. Goncharov. Authorship identification of a russian-language text using support vector machine and deep neural networks. *Future Internet*, 13(1):3, 2020.

R. Sarwar, Q. Li, T. Rakthanmanon, and S. Nutanong. A scalable framework for cross-lingual authorship identification. *Information Sciences*, 465:323–339, 2018.

C. Sun, X. Qiu, Y. Xu, and X. Huang. How to fine-tune bert for text classification? In *Chinese Computational Linguistics: 18th China National Conference, CCL 2019, Kunming, China, October 18–20, 2019, Proceedings 18*, pages 194–206. Springer, 2019.

H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

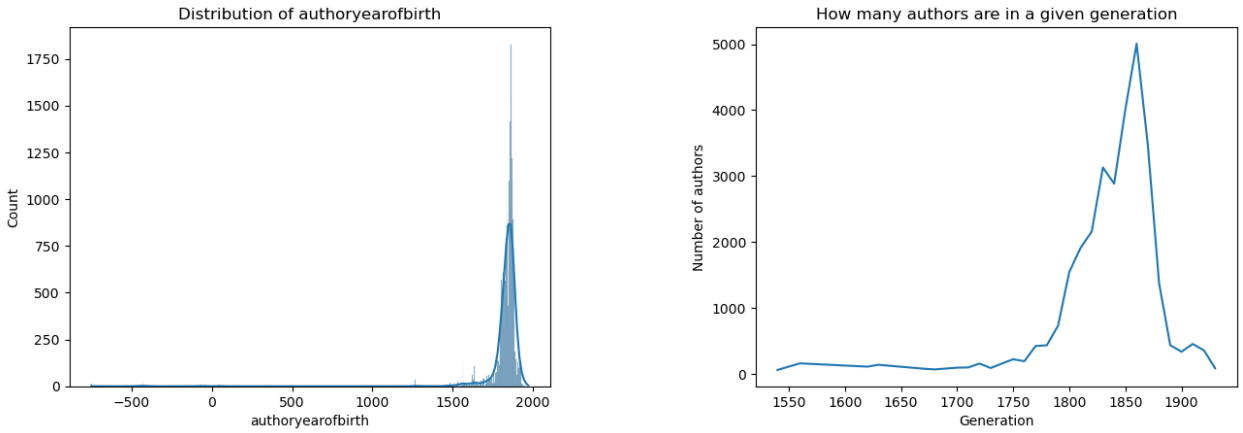
P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, and L. Bottou. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(12), 2010.

C. Xing, L. Ma, and X. Yang. Stacked denoise autoencoder based feature extraction and classification for hyperspectral images. *Journal of Sensors*, 2016, 2016.

Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 1480–1489, 2016.

7 Appendix

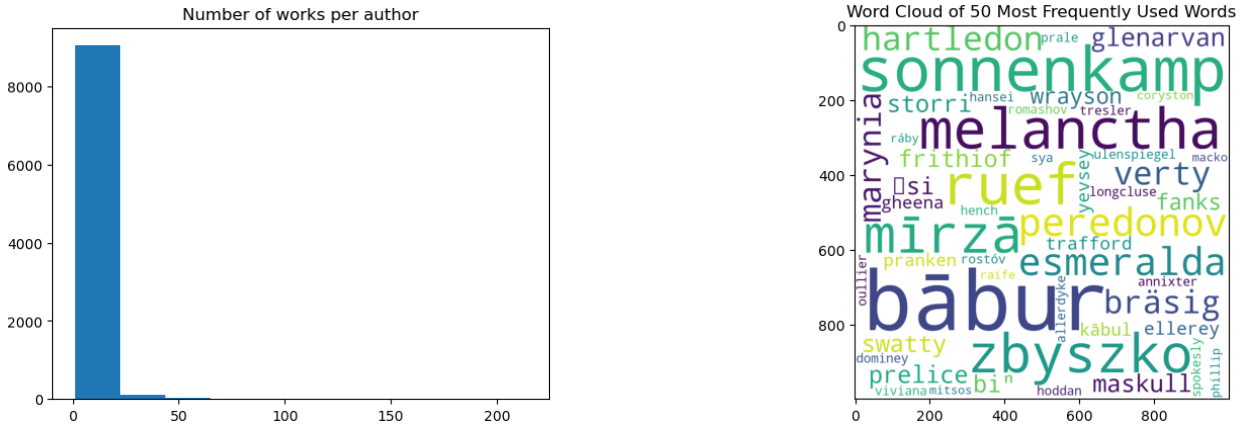
7.1 Data Analysis Plots



(a) Distribution of Author's Year of Birth

(b) Distribution of Author's Generation

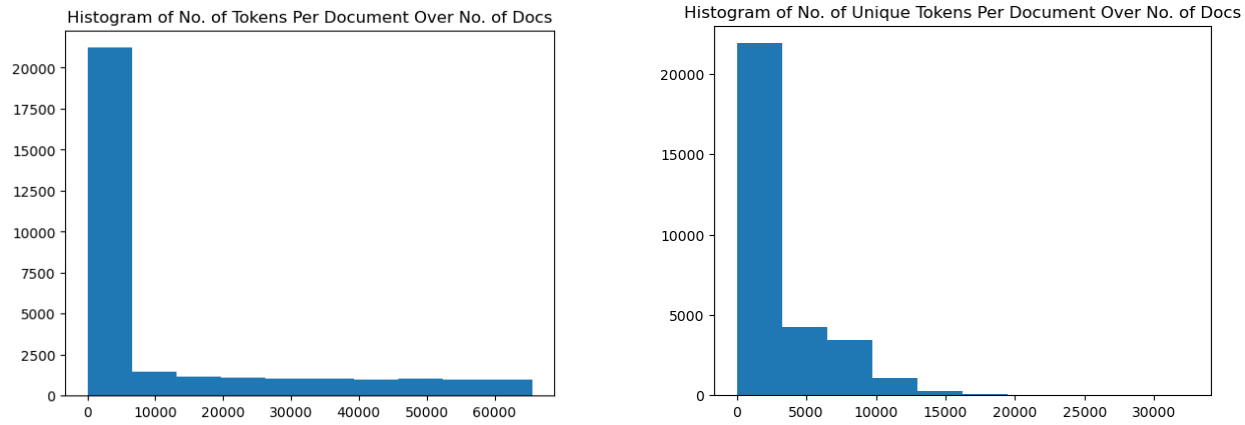
Figure 4: Distribution of Author's Lifespan



(a) Words Per Author

(b) Word Cloud by Frequency

Figure 5: Words Stats



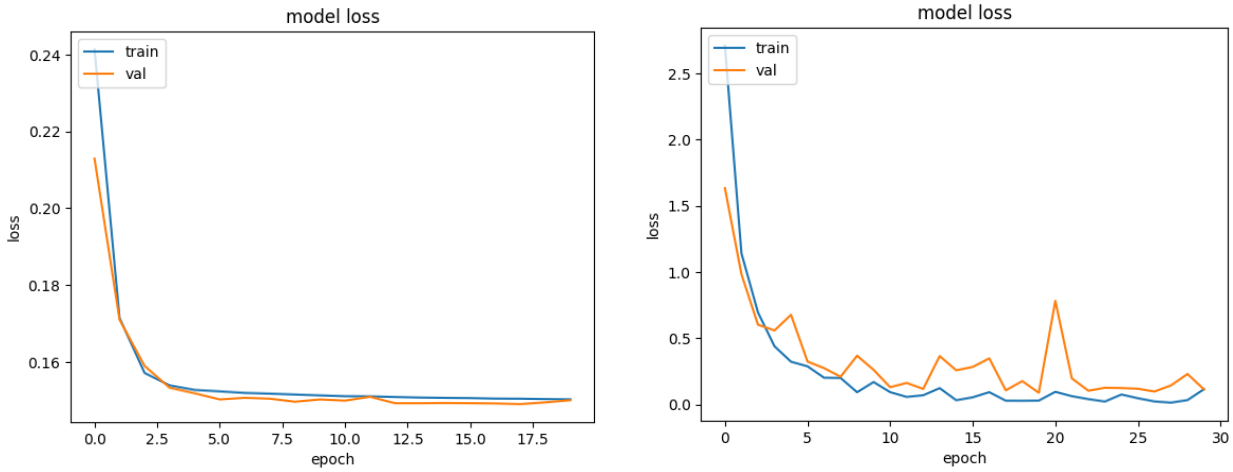
(a) Histogram of Tokens

(b) Histogram of Unique Tokens

Figure 6: Token Histogram

7.2 Pre-training and fine-tuning of best SVM model

The plots for the whole document dataset

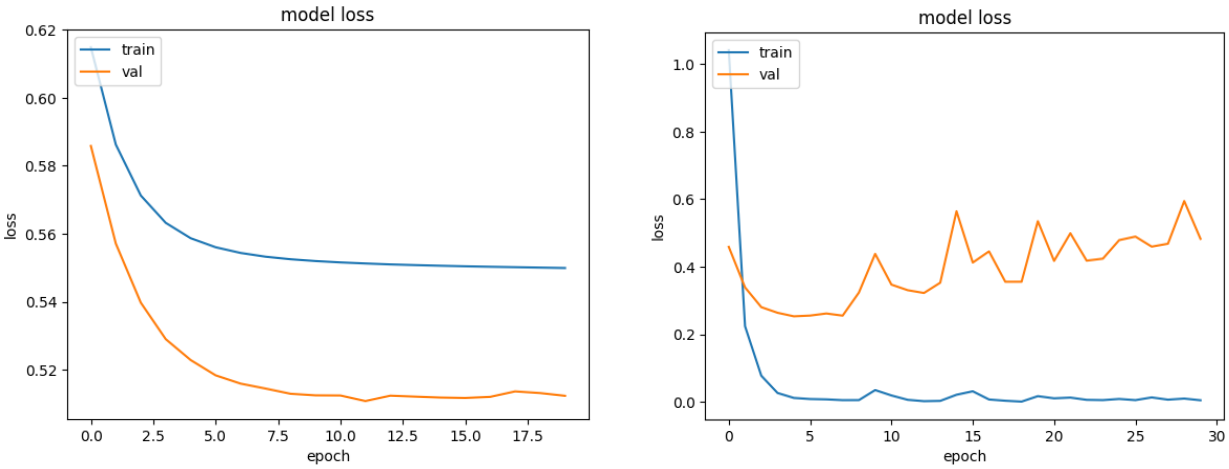


(a) Pre-training of SDAE

(b) Fine-tuning of SDAE

Figure 7: 1-layer SDAE trained on Whole Document

The plots for the 2000 x 512 tokens dataset



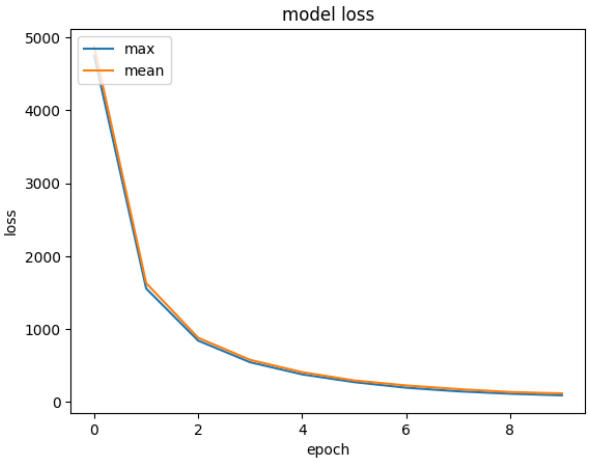
(a) Pre-training of SDAE

(b) Fine-tuning of SDAE

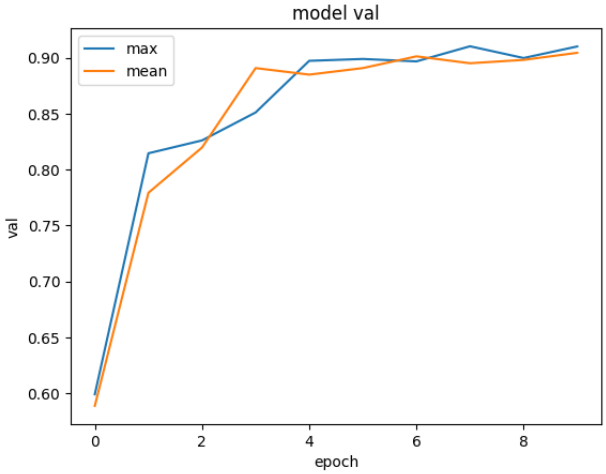
Figure 8: 2-layer SDAE trained on 2000 x 512 token dataset

7.3 Training of BERT plots

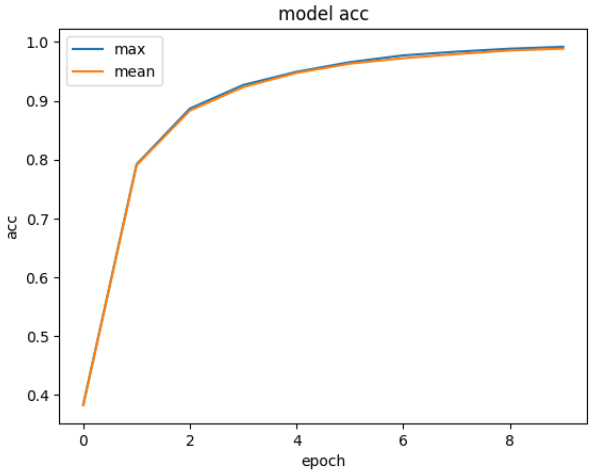
Plot of training loss of BERT



(a) Training loss of BERT



(b) Validation accuracy of BERT



(c) Test accuracy of BERT

Figure 9: BERT Max-Pooling & Mean-Pooling 2000 x 512 token dataset

7.4 Diagrams for the pre-training and fine-tuning of SDAE

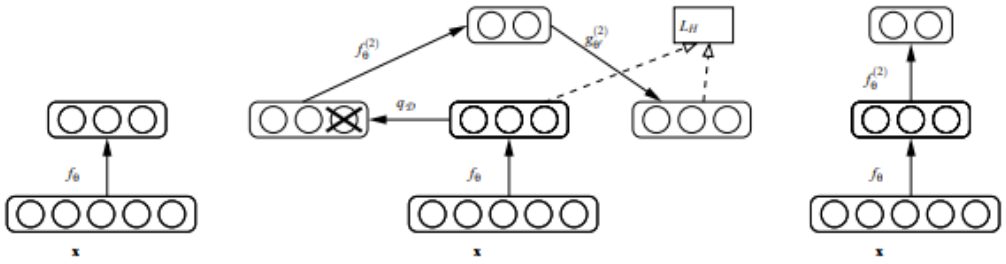


Figure 3: Stacking denoising autoencoders. After training a first level denoising autoencoder (see Figure 1) its learnt encoding function f_θ is used on clean input (left). The resulting representation is used to train a second level denoising autoencoder (middle) to learn a second level encoding function $f_\theta^{(2)}$. From there, the procedure can be repeated (right).

Figure 10: Illustration of SDAE pre-training [Vincent et al., 2010]

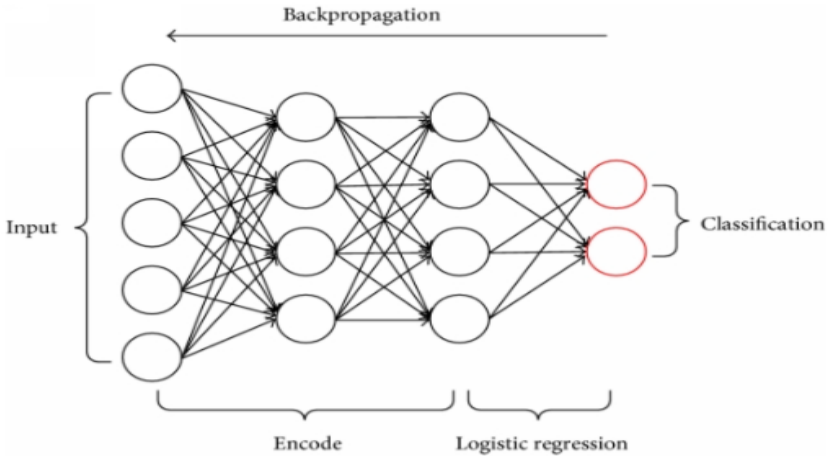


Figure 11: Illustration of SDAE fine-tuning [Xing et al., 2016]