EasyChair Preprint

№ 15684

Machine Learning Approach for Suicide and Depression Identification with Corrected Unsupervised Labels

Meenal Badki

January 7, 2025

Machine Learning Approach for suicide and depression identification with corrected unsupervised labels

Abstract:
It can be life-saving to identify suicidal thoughts in depressed people early on so that they can receive the necessary medical care and support. Novel NLP research endeavours to categorise, based on provided text, whether an individual is clinically healthy or suicidal. But there haven't been any significant efforts to distinguish between suicidal ideation and depression, which is a different and significant clinical challenge. Web query data has become a promising alternative because EHR data suicide notes, and other verified sources are hard to come by. Internet sources like Reddit are credible even in a clinical setting because they provide an anonymous environment that encourages candid symptom disclosure. Nevertheless, web-scraped labels resulting from online datasets also contain inherent noise, which makes a noise-removal procedure necessary to enhance performance. Consequently, we propose SDCNL, a deep neural network method for classifying suicide versus depression. Our algorithm is trained using online content, and we suggest a novel unsupervised label correction technique that does not require prior knowledge of the noise distribution to verify and correct noisy labels, in contrast to previous work. Our thorough testing of numerous deep word embedding models and classifiers demonstrates the potent performance of SDCNL as a novel and clinical solution for a difficult issue. Another use cases that we have considered in this research are : financial news dataset, Vietnam news agency, credit default swap. We have derived to final results for all the four use cases as mentioned above, majorly developing from the use case of suicide/depression dataset by building and testing the framework and then testing them on other three datasets.

Keywords: Suicide/Depression, Noisy labels, Deep Learning, Online Content, Natural Language Processing, Unsupervised Learning, credit default, financial news, news agency.

1 Introduction

Depression is still one of the most important problems in the world, and if it is not treated, it frequently leads to suicidal thoughts or attempts. The diagnosis of depression and the determination of the threshold for suicidal ideation are significant issues for both the individual and the general public. Suicide notes, Electronic Health Records (EHRs), and questionnaire data are among the many sources of information used by many current methods for identifying suicidal ideation [10]. However, acquiring data in such formats is challenging and ultimately results in limited datasets, complicating attempts to accurately automate diagnosis.

On the other hand, as the Internet and social media in particular have grown, online forums have become well-liked sources of help and support for those in need. Because these forums are large and open to the public, there is a chance that they will be scraped in order to build databases for automated mental health diagnosis systems. An increasing number of studies are using this data for diagnostic purposes, particularly for neural network based approaches that need large datasets to be trained efficiently, these studies are described in the review paper[10].

Specifically, Reddit has become a significant data source for mental health disorder diagnosis[19]. Reddit is an online social media platform where users create sub-reddits, or communities with specific goals. Sub-reddits like r/depression and r/SuicideWatch openly discuss mental health issues and provide personal accounts of their experiences. In order to maintain privacy and anonymity, Reddit especially permits users to create disposable, alternate accounts. This encourages disclosure and makes it possible for people who don't have many support systems in real life to get help online [5]. The large user base, transparency of these online environments, and controlled moderation of these posts to guarantee authenticity offer a hitherto unseen chance for large-scale computational analysis of mental health problems.

There is still a dearth of research on identifying when people with underlying mental health issues, like depression, are at risk of attempting suicide, despite the substantial research on using text to distinguish between mentally stable and healthy patients. This poses a significant clinical challenge for the development of new treatments for depression as well as for the application of existing treatments [2, 13]. There are currently no effective methods for differentiating between suicidal and healthy behaviour because it is a more complex task to distinguish between suicidal tendency and depression. Because online data is informal and lacks verification, it has historically been challenging to use in such fine-grained situations due to the unreliability of its labels. Specifically, subreddit-based data labelling depends on self-reporting because users select the subreddit that best describes their mental state, which may lead to an over- or underreporting of their diagnosis. Because there is a chance that some labels will be tainted, this idea is known as noisy labels. According to estimates, noisy labels can cause 10% to 40% of datasets to deteriorate [20], which poses significant difficulties for machine learning algorithms.

Three prominent groups of current approaches to the noisy label problem are as follows: data cleaning methods, noise-tolerant methods, and noise-robust methods [20, 7]. While noise-tolerant methods directly model

the noise during training, noise-robust approaches rely on algorithms that are inherently less sensitive to noise (e.g., lower dimensional or regularised algorithms). Despite the fact that both strategies have drawn a lot of interest in the field of image processing [20], NLP algorithms cannot use these techniques. Several noisy label methods have been proposed recently in the NLP domain [8, 12, 22]. The suggested approaches, however, are not ideal for our task. Some approaches, for instance, use a smaller trusted data set to correct a larger noisy data set [8, 22]. However, this is not practical for our application because it is impossible to determine which posts are accurate. Alternative approaches necessitate training a network from scratch using corrupted labels [12], which is less effective at utilising transfer learning from state-of-the-art models that have already been trained [6, 17, 4] and requires a comparatively large amount of data.

Techniques for cleaning data are better suited for the current task. Nevertheless, the majority of label cleaning techniques currently in use either assume or require knowledge of the dataset's noise distribution [11, 9]. In our use-case, an unsupervised technique like clustering is needed since the noise distribution is unknown beforehand. While some approaches for noisy label learning employ unsupervised clustering algorithms [3], none of these methods yield correct labels. Instead, they use instance weighting or exclusion to train a model to be resistant to noise. These approaches would not work well for our task because of the high noise proportion. In particular, deep neural networks require a lot of data, so weighting or deleting a lot of data would negatively impact performance. Thus, the present task requires an unsupervised method for data cleaning that utilises label correction rather than elimination. To the best of our knowledge, there are no current methods which perform label correction using unsupervised clustering methods, and particularly not in the NLP domain.

In this paper, SDCNL is presented to address the unexplored issue of classifying between depression and more severe suicidal tendencies using web-scraped data and neural networks. The primary contributions which can be summarised as follows:

- Deep neural network sentiment analysis applied for depression versus suicidal ideation classification, an important but un-explored clinical and computational challenge.
- A novel, unsupervised label correction process for text-based data and labels which does not require prior noise distribution information, allowing for the use of mass online content
- Extensive experimentation and ablation on multiple datasets, demonstrating the improved performance of all SDCNL components on the challenging proposed task.

2 Methods

Figure 1 shows an outline of the SDCNL method. We start by using word embedding models to process text data that was scraped from Reddit. Since clustering algorithms perform poorly in high-dimensional domains, these embeddings are subsequently processed using an unsupervised dimensionality reduction algorithm [21]. After that, the reduced embeddings are fed into a clustering-based algorithm that predicts new labels noise-free and in an unsupervised fashion. To correct the ground-truth labels, these alternate labels are compared against the ground-truth labels using a confidence-based thresholding procedure. A supervised deep neural network is then trained using the corrected set of labels.

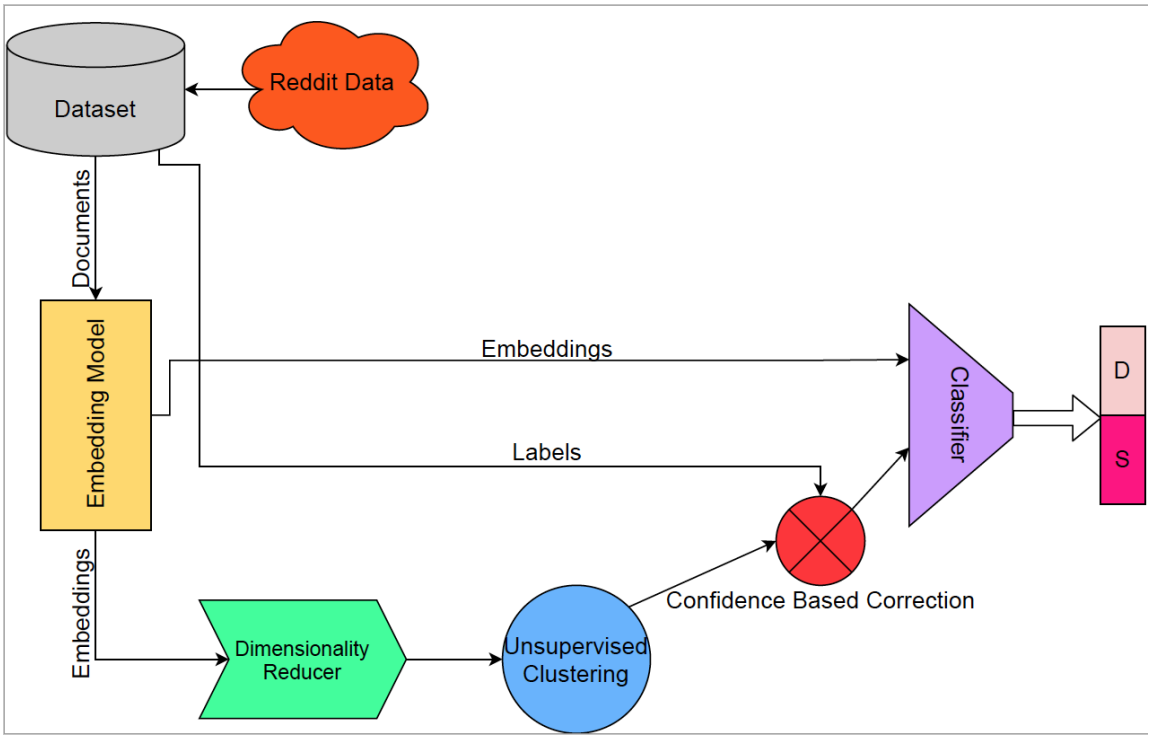


Figure [1]. Schematic of SDCNL pipeline used for classification of suicide vs depression and noisy label correction via unsupervised learning

2.1 Embedding Models

Initially, our framework uses word embedding models to create numerical word embeddings from raw documents, called posts in our case. While our suggested approach can be applied to any embedding model, our task requires greater-than-word text embedding models that are optimised to handle phrases, sentences, and paragraphs. Sentence-BERT [17], Google Universal Sentence Encoder (GUSE) [4], and Bidirectional Encoder Representations from Transformers (BERT) [6] are the three cutting-edge transformers that we experiment with. The latest bidirectionally trained transformer, BERT, produces a 768 × 512 dimensional vector of embeddings and performs well on a range of benchmark NLP tasks. Sentence-BERT is a 768 × 1 dimensional vector that is the result of retraining and optimising the original BERT architecture for longer inputs and better clustering performance. GUSE is a transformer that is also optimised and trained for text longer than a word, but it yields a vector that is 512 by 1 in dimensions instead. Sentence-BERT is a 768 × 1 dimensional vector that is the result of retraining and optimising the original BERT architecture for longer inputs and better clustering performance. GUSE is a transformer that is also optimised and trained for text longer than a word, but it yields a vector that is 512 by 1 in dimensions instead.

While some classifiers need embeddings at the word level, others need embeddings at the document level. The multi-dimensional word level embeddings and document level embeddings that BERT produces are supplied by CLS tokens. We adjust the inputted embeddings to meet the requirements of the classifier based on its specifications. Furthermore, as baselines, we conduct experiments on three vectorizers: Count Vectorizer (CVec), Hashing Vectorizer (HVec), and Term Frequency–Inverse Document Frequency (TFIDF).

2.2 Label correction

We suggest an unsupervised label correction technique to deal with the problem of label noise for our task. In order to transform the high-dimensional features produced by the embedding models into lower-dimensional representation, we first feed our word embeddings through a dimensionality reduction algorithm. The "Curse of Dimensionality" refers to the fact that high-dimensional data frequently produces poor performance and poorly separated clusters due to the nature of most clustering algorithms [21]. As such, it is necessary to represent the data in lower dimensions. We test out three different dimensionality reduction techniques: Uniform Manifold Approximation and Projection (UMAP), Deep Neural Auto-encoders, and Principal Component Analysis (PCA) [15]. A popular reduction technique called PCA takes a matrix of numerical data, identifies its key information, and represents it as a set of new orthogonal variables. Auto-encoders enforce effective representation learning by compressing data into a low-dimensional space using an unsupervised neural network and then reconstructing it while preserving as much information as possible.

The reduced embeddings are derived from the encoder portion's output. In order to create a low-dimensional graph that is as close to the input as possible, UMAP creates a graph from high-dimensional data. UMAP works especially well with high-dimensional data because of its faster processing and better global structure preservation.

Our word embeddings are reduced in size, and then we employ clustering algorithms to split them into two different clusters so that we can give each post a new label. We use clustering algorithms because they are non-supervised; this is important because we need a clustering procedure that is independent of the web-scraped labels because we don't know the noise distribution in the labels beforehand. Gaussian Mixture Model (GMM) is the clustering algorithm that we employ. In order to use probabilities to cluster given data, a parametric probability density function known as a GMM is used as a model of the probability distribution of continuous measurements. We begin with K-Means clustering as a baseline. K-means aims to partition n observations into k clusters, with each observation being assigned to the cluster with the closest mean. The clusters minimise the distance within the cluster while optimising the distance between the clusters. We employ subspace clustering via spectral clustering [16], which enables unsupervised clustering of high-dimensional data by locating clusters in various sub-spaces within a dataset, in order to get around the dimensionality reduction requirement.

Two labels are now associated with each word embedding: the ground-truth labels, which were initially based on the sub-reddit, and the new labels that came from unsupervised clustering. We then use a thresholding technique based on confidence to adjust the ground-truth labels. The predicted label replaces the ground-truth label if the clustering algorithm predicts a label with a probability above τ, a tuned threshold; if not, the ground-truth label is assumed. To avoid making false corrections, the tuned threshold makes sure that only highly confidently predicted labels are applied when correcting the ground truth. At last, the appropriate set of labels is matched with the corresponding post. Using the clustering algorithms, we are able to obtain class probabilities. Note that our label correction method can be used in any NLP domain or even in other fields, such as the imaging field.

2.3 Classification

We train our deep neural networks to identify whether the posts show suicidal or depressive sentiment using a corrected label set. Any classifier can be used in place of the ones we tested, much like the embedding process. However, since neural networks enable precise representation learning to distinguish between the similar semantics of our two classes, we seek to demonstrate the efficacy of deep neural classifiers for our suggested task.

A dense neural network, a convolutional neural network (CNN), a bidirectional long short-term memory neural network (BiLSTM), and a gated recurrent unit neural network (GRU) were the four deep learning algorithms we experimented with. Three common machine learning models were assessed as baselines: support vector machines (SVMs), multinomial naive bayes (MNB), and logistic regression (LogReg).

2.4 Datasets

Building on our task of suicide or depression classification, we create a primary dataset. This data is a result of Reddit web scraping. Using the Python Reddit API, we gather our data from sub-reddits. We extract content primarily from two sub-reddits: r/Depression and r/SuicideWatch. In total, 1,895 posts make up the dataset. The original post text serves as our input, and the sub-reddit to which it belongs serves as a label. These are two fields that we use from the scraped data. Suicidal posts are labeled on r/SuicideWatch, and depressed posts are labeled on r/Depression. In our code, we provide both the web-scraping script and this dataset.

Furthermore, we validate our label correction approach using the Reddit Suicide C-SSRS dataset [1]. There are 500 Reddit posts from the r/depression sub-reddit in the C-SSRS dataset. Psychologists have categorised these posts using the Columbia Suicide Severity Rating Scale, which gives increasing labels based on the intensity of depression. Because the labels in this dataset are clinically verified and come from the same Reddit domain, we use it to validate our label correction method. We use the IMDB large movie dataset, a popular NLP benchmark dataset, to further validate the label correction method [14]. There are fifty thousand polar movie reviews in the dataset, which is a binary classification task. For evaluation, we employ a random subset of the samples.

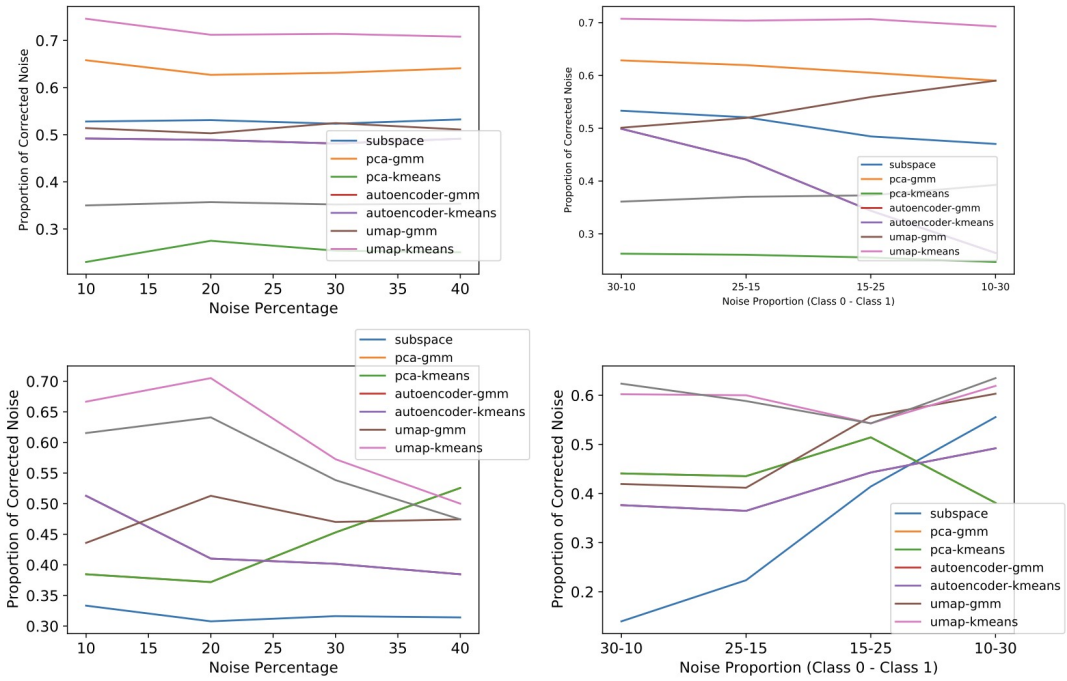
For comparison of the methods (mentioned in this paper) against other related tasks and methods, we build a dataset for binary classification of clinically healthy text vs suicidal text. We utilise the two sub-reddits r/CasualConversation and r/SuicideWatch. r/CasualConversation is a sub-reddit of general conversation, and has generally been used by other methods as data for a clinically healthy class [18].

3 Experimental Results

3.1 Implementation Details

We reserve 20% of each dataset for use as an external validation set. Tensorflow was used to implement the deep learning models, while Sci-Kit Learn was used to implement the remaining models. We utilized a binary cross-entropy loss function and the Adam optimizer to train the deep learning models. We set τ to 0.90 for all experiments based on tuning experiments, where we recorded accuracy at varying values; however, similar values produced similar performance. We employ the following five metrics to measure classification accuracy: area under the curve score (AUC), recall (Rec), precision (Prec), accuracy (Acc), and F1-score (F1). The code includes hyper-parameters specific to the model.

Figure [2]. Correction rates of the label correction algorithms at different noise rates on the IMDB (top) and C-SSRS (bottom) dataset. Left: correction rates with uniform injections of noise. Right: correction rates with class-weighted injections of noise (ratios such as 30%-10% or 25%-15%)



3.2 Label Correction Performance

We report both the accuracy of the clustering algorithm at correcting noisy labels and the classification performance following label correction to assess our clustering performance. As training labels get noisier, it is expected that classification on a clean test set will decrease [7]. Consequently, we argue that the correction method is effective if, following label correction, our algorithm's classification accuracy rises. Importantly, the labels are not clinically verified because our suggested task uses a web-scraped dataset. Unfortunately, this means that since we don't have the true labels, we can't evaluate the correction rate of noisy labels. Hence, we perform label correction evaluation on the benchmark IMDB dataset to demonstrate the value of our method in a general setting, as well as on the C-SSRS dataset to demonstrate effectiveness in our specific domain.

Clustering performance

We introduce noise into the label set at varying rates in order to assess the clustering's effectiveness. Since real-world datasets are estimated to have a label noise rate of 8% to 38.5%, we corrupt 10-40% of the dataset at both uniform and imbalanced rates. Other noisy label papers also have standard noise levels [20]. Next, we assessed how well the clustering algorithms performed in terms of cleaning up the noisy labels.

As demonstrated in Figure 2, our noise correction technique can reliably eliminate more than 50% of injected noise while maintaining a false-correction rate below 10% on both datasets. Moreover, performance does not significantly deteriorate at higher noise percentages, which can be difficult to attain [7]. The SDCNL label correction works well on the C-SSRS dataset, demonstrating the method's applicability in our particular domain, as well as the IMDB dataset, demonstrating the method's generalisability. We use umap-kmeans and umap-gmm

as the optimal combinations of reduction and clustering algorithms in our proposed method. To the best of our knowledge, almost all noisy label correction methods don't evaluate performance on classification accuracy after correction, as it allows for comparison to other noisy label methods that don't use label correction. However, because most recent noisy label methods are in the image domain, drawing comparisons to related work is unfeasible.

Classification Performance after Label Correction

Lastly, to demonstrate the effectiveness of the label correction method, we train a classifier on noisy C-SSRS data and validate on a separate C-SSRS test which has no noise. Finally, we train a classifier on noisy C-SSRS data and validate on a separate noise-free C-SSRS test set to show the efficacy of the label correction method. We then train the model with the corrected labels, validate on the same unaltered test set, and apply our label correction method to the same set of noisy labels.

Model	Accuracies per Task (%)	
	Noisy	**Corrected**
guse-dense	57.97	70.63
bert-dense	48.86	70.13
bert-bilstm	56.71	68.35
bert-cnn	55.70	70.13

Table [1]: Classifier accuracy comparison after injecting randomized noise (20%) into C-SSRS labels (left) against using the label correction method (UMAP + GMM) to remove the artificial noise and subsequently training classifier (right).

We demonstrate that applying our label correction method results in a minimum 11% increase in accuracy (Table 1). Our label correction process is useful for cleaning noisy labels in NLP and for our task because it operates on a dataset within the same domain. Furthermore, employing a probability threshold affects performance, as demonstrated in Figure 3, demonstrating the significance of this step in guaranteeing the accuracy of the corrected labels. As a result, we decide on the thresholding strategy for our finished model.

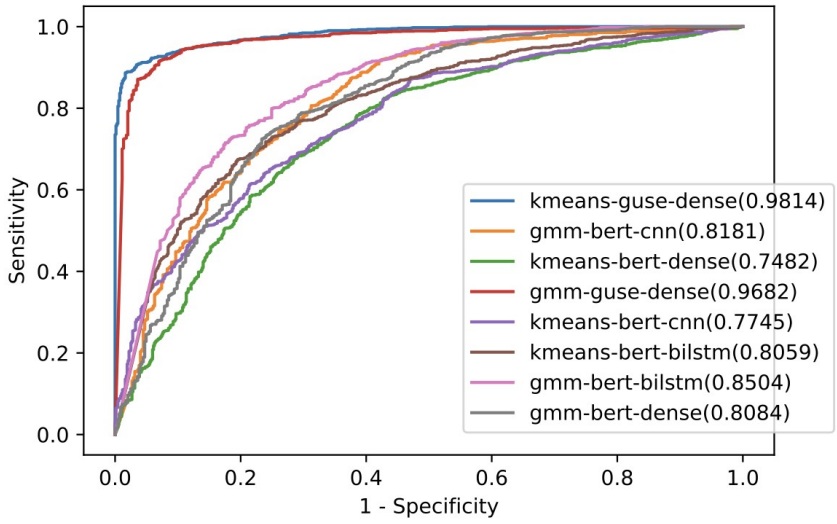


Figure [3]. ROC curves of model performance after using label correction. The 4 best combination of models with the two final label correction methods are shown (GMM vs K-means). UMAP used to reduce the dimensions of the embeddings.

3.3 Classification Performance

Metrics (%)	Model Combinations			
	guse-dense	bert-dense	bert-lstm	bert-cnn
Acc	72.24	70.50	71.50	72.14

Rec	76.37	71.92	67.77	73.99
Prec	71.38	70.77	74.28	72.18
F1	73.61	71.25	70.70	72.92
AUC	77.76	75.43	77.11	76.35

Table [2]. Performance of the four best combinations of embedding models and classifiers

Deep Neural Network Performance

We conducted all of the experiments and identified the four most effective combinations to run the remaining tests. The main dataset comparing depression and suicide is used to train these combinations, and the labels are left uncorrected. Appendix A in the supplemental contains the full results. Table 2 displays the performance of the four most powerful models. These are the combinations: bert-dense, bert-dense with a fully-dense neural network, bert-bilstm with a Bi-LSTM neural network, and bert-dense with a fully-dense network for GUSE. All DNNs outperform the baselines, demonstrating the significance of our contribution. We use the four above models for all upcoming experiments.

Comparison to other tasks

Although NLP text-based methods for suicide detection have been the subject of much research, none has been specifically focused on our task of identifying low-risk depression from suicidal ideation. We put our suggested model to the test once more by dividing the population into clinically healthy and suicidal categories.

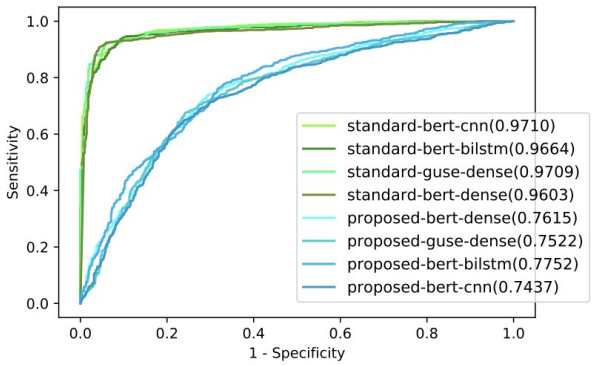


Figure [4]. ROC curves of model performance from four best models on our task (proposed) against the conventional suicide vs healthy task (standard)

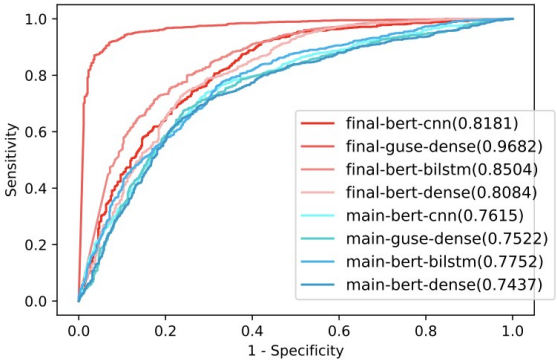


Figure [5]. ROC curves of performance of top 4 models with label correction (red) against the same models without using label correction (blue)

Figure 4 displays that on identical models, the commonly-researched task achieved much stronger baseline performance compared to our task; therefore, our baseline evaluations should correspondingly be lower. This task difficulty also demonstrates the value of automated methods such as ours in clinical settings.

Final Evaluation

We assessed SDCNL's classification efficacy using the Reddit dataset. Using the suggested label correction technique, we produced ground-truth labels for this dataset. It would be ideal to use labels supplied by mental health professionals to give a comprehensive test of our model, but there isn't one available for our task.

We argue that the use of the proposed label correction method on the Reddit dataset is justified, since we have shown it to be effective on the C-SSRS and IMDB datasets.

Metrics (%)	UMAP-KMeans				UMAP-GMM			
	guse-dense	bert-dense	bert-bilstm	bert-cnn	guse-dense	bert-dense	bert-bilstm	bert-cnn
Acc	**92.61**	73.56	75.15	74.72	**93.08**	83.74	84.16	84.59
Prec	**93.61**	83.18	84.02	87.38	94.76	**95.51**	93.10	95.38
Rec	**94.85**	77.66	79.00	76.65	**96.16**	85.08	87.09	86.05
F1	**94.22**	80.25	81.19	81.64	**95.44**	89.99	89.99	90.45
AUC	**98.18**	76.83	80.93	78.69	**96.88**	81.97	85.08	82.91

Table [3]. Final classification performance after using the two label correction methods, with and without a thresholding scheme. Best performances for each noise removal method are bolded. Best overall model is bolded

and italicised.

The final results of the models with threshold label correction (GMM) and without (K-Means) are shown in Table 3. We confirm that the thresholding component is crucial because all metrics significantly outperform the K-Means baseline. Furthermore, our label correction method produces a much higher AUC value and enhances the ROC curve, as shown in Figure 5. We obtain significantly better results by correcting the labels in the test set with high expected noise. In our final combination, we use a fully-dense network for the classifier, GUSE as the embedding model, UMAP for dimensionality reduction, and a GMM for the clustering algorithm to correct labels. Because GUSE embeddings produce fewer embeddings than the other transformers and preserve information, they probably produce the best results.

Pre-processing of the dataset
We refer to Reddit data for this paper as a part of suicide/depression dataset
We download nltk libraries consisting of word-net and stopwords
We create a user agent
We build a scraper function to create scraped posts outputted as lists, the output we get is the number of posts downloaded and the number of unique posts
We remove any repeat posts thereafter
Then we perform scraping of suicide watch json data from Reddit
Then we perform scraping of depression json data from Reddit
Then we convert the data to csv (after this above json data has been interpreted in pandas data-frame), and we save as combined dataset (consisting of test and

Embedding models (transformers)
This part is built which contains the implementation of BERT and Google Universal Sentence Encoder (GUSE) transformer
First part is GUSE:
We load GUSE from tensor-flow hub development platform
Then we transform the text features to word embeddings using GUSE
Then we import bert-base, tokenisers and models from libraries

Dimensionality reduction
This part is built which contains the clustering algorithm with proper hyper-parameters set
Algorithms that we are using are PCA, Deep Auto-encoder and UMAP
We initialise proper hyper-parameters for PCA, Deep Auto-encoder and UMAP accordingly
We get low dimension features from these aforementioned algorithms

Clustering methods
Then we include clustering algorithms like GMM, K-means and Subspace clustering
We get predictions and probability values from the aforementioned algorithms
We refer to already implemented subspace clustering algorithms to get the predictions which we refer to in the next steps

Threshold-based correction
We use a threshold-based correction to determine whether to use the ground truth label or unsupervised clustering label
We refer to train and test features/labels to do threshold based correction
The condition for correcting threshold based probability values is tau = 0.9
We compare the predicted train/test probabilities with original train/test probabilities
If predicted train/test probabilities > tau or predicted train/test probabilities < (1-tau), we get final train/test labels else we retain original train/test labels

Classification
This part which contains the implementation of our deep neural classifiers with correct hyperparameters
We load the features after creating them
We train hyperparameters
The algorithms that we have referred to are CNN, fully dense network, Bi-LSTM
Henceforth, we get the accuracy/precision/recall/g-mean/AUROC values

Use case for social media data analysis:

Blogs, online forums, comment sections on media sites and social networking sites such as Facebook and twitter all can be considered as social media. These social media can capture millions of peoples' views or word of mouth. Communication and the availability of these real time opinions from people around the world make a revolution in computational linguistics and social network analysis. Social media is becoming an increasingly more important source of information for an enterprise. On the other hand people are more willing and happy to share the facts about their lives, knowledge, experiences and thoughts with the entire world through social media more than ever before. They actively participate in events by expressing their opinions and stating their comments that take place in society. This way of sharing their knowledge and emotions with society and social media drives the businesses to collect more information about their companies, products and to know how reputed they are among the people and thereby take decisions to go on with their businesses effectively. Therefore it is clear that sentiment analysis is a key component of leading innovative Customer Experience Management and Customer Relationship Marketing focused enterprises. Moreover for businesses looking to market their products, identify new opportunities and manage their reputation. As businesses look to automate the process of filtering out the noise, understanding the conversations, identifying the relevant content and take appropriate action upon it. Many are now looking to the field of sentiment analysis. In the era which we live today, sometimes known as information age, knowledge society; having access to large quantities of information is no longer an issue looking at the tons of new information produced everyday on the web. In this era, information has become the main trading object for many enterprises. If we can create and employ mechanisms to search and retrieve relevant data and information and mine them to transfer it to knowledge with accuracy and timeliness, that is where we get the exact usage of this large volume of information available to us.

Approach:

To analyse sentiments and then come to a conclusion through them, we need to have enough sentiments in the correct format. There are thousands and millions of sentiment data in the web, especially in social media sites that can be used to get valuable conclusions. But they are not in a correct format or not in a structured way to get maximum usage out of them. We need to convert them to a correct format and use them as we want. It is the first part in our approach, which is developing a crawler to crawl data from Twitter social media. The Crawler should be able to crawl user sentiments from twitter and at the same time get user details in order to do product profiling for customers as the later part of the whole approach. After having access to large sources of data which is in a structured manner through the crawler and using a database, the next step is analysing sentiments. Sentiments can be in different languages; in our project we cover the English language. Analysing sentiments is a way of processing natural languages, therefore this part is about natural language processing. For this we use Natural Language Toolkit, also known as NLTK which is a leading platform for building Python programs to work with human language data. There are different ways that we can use to analyse sentiment data using this toolkit, but none of them gives hundred percent accuracy, because natural languages are used in many different ways by people. In our approach that is presented through this paper, we have implemented two supervised learning techniques named Naïve Bayes Classification and Maximum Entropy Classification to classify unknown sentiments, which later gives the probability of how much if a sentiment is positive and negative. As another method it uses SentiWordNet which is a lexical database that assigns scores to words and thereby finds a sentiment score to an entire sentence. It chose the first method which is Naïve Bayes Classification as the best method out of these three techniques after evaluating all the three methods. Using the selected sentiment analysing method in this approach, it gives not only positive, negative and neutral sentiment score to the user sentiments, but it can solve the issues of using short words, different jargon words and smileys in social media. To differentiate the senses of ambiguous words such as "apple", it used word sense disambiguation technique and could be able to differentiate different senses for ambiguous words. Then as the third part of the project, we created a dashboard to show the results from the crawler and sentiment analysis using python. Here it will display how the sentiment polarity differs for a selected item with the time using a graph. Using this we can visualise how it is changing the user sentiment polarity of a specified brand or product with time. As the final part, the output of the sentiment analysis module will be used as the input for data mining module. It used the sentiment scores of a particular product or service with the user information such as age, profession, area and gender to profile products, analyse the trend for that particular product or service and forecasting. The result from data mining can be applicable in a most profitable way such as analysing how people sentiments changes and will change for products and services with their age, location, profession and gender. Anyone who is interested in searching what people say about a particular product or service can use this system. This is especially very useful for organisations producing products and services, to know what people are talking about their products and

services and were they positive or negative, who are the competitors they have and what they can do to improve the reputation of their products and services among customers and what features should they include in order to attract positive remarks about their products.

Objectives (of this use-case and points that would be covered in this paper):
- Understand how the portfolio holdings of CDS were reported on Form N-CSR (S) and Form N-Q.
- Identify the proportion of mutual funds using CDS.
- Among the fillings with CDS information, identify:
the proportion of filings that are easy to extract (e.g., well tabulated with pure numbers), the proportion of fillings that could be extracted using NLP (e.g., holdings are described in words), the proportion of filings that cannot be effectively extracted.
- Extract the characteristics of the CDS holdings:
Identifier of the mutual funds (eg., CIK, name, etc), Metadata (eg., reporting period, etc), the particulars of the CDS holdings (Reference entity, the direction of trade like buy/sell, notional amount, currency, contract premium, counterparty)
- After extracting the CDS information, some downstream analysis should be explored: time-series analysis of the mutual funds utilization of CDS during 2004 to 2016. (normal times > pre-crisis > crisis > post-crisis > regulated > call for reviving), explore the pattern of using index CDS, sovereign CDS, and single-name corporate CDS, explore whether the funds using more complex sentences describing their CDS positions make more money than those using concise tabulated numbers.

Another use case on financial news sentiment analysis:
Literature Review/Background:
The term “Sentiment Analysis” was first defined in 2003 by Nasukawa and Yi [1] as “determining the subjectivity polarity (positive or negative)” and polarity strength (strongly positive, mildly positive, weakly positive etc.) of a given review text; in other words – determining the opinion of the writer.” Turney’s pioneering work on Sentiment Analysis [] applied an unsupervised approach to classify review data into positive class and negative class. This paper focuses on finding sentiments in another type of dataset: financial news, and demonstrating correlation between the sentiments of financial news and stock market variation. Classical finance theory assumes investors are rational, non-emotional entities and prices represent the equilibrium of expected returns and possible risks. This leaves no scope for investor sentiment to play a role in price determinations. Modern behavioural finance, however, recognises both sentimental investors and rational investors. [3] states that “Now, the question is no longer, as it was a few decades ago, whether investor sentiment affects stock prices, but rather how to measure investor sentiment and quantify its effect.” Every day, a lot of company news is published that directly affects the investors’ behaviour. Manually reading this news and labelling it as positive or negative is a very difficult task due to the sheer volume of news generated which is increasing rapidly. Moreover manual evaluation of news may not be completely objective due to factors like reader bias, emotion and fatigue. Automatic sentiment analysis can avoid these pitfalls. The sum total of information received by the investors is reflected through the stock price of the firms. Through this process, information is transformed from a textual form to a numerical form. This process of conversion is very useful, because it allows information to be easily summarised and enables us to compare the sentiments of news with the market returns. There may be variations about the exact meaning of a piece of news, but there cannot be any variation about market returns. The financial news that makes a positive impact on the stock market returns is good and the one that makes a negative impact on stock market returns is bad [4]. In comparison to the work done in sentiment classification applied to the review domain or product reviews, very little work has been done in the field of application of these techniques in the financial domain using unsupervised approach. This paper tries to address this research gap. The overall purpose of the study is to propose a semantic orientation based unsupervised approach for finding sentiments strength of financial text.
Turney [2] in a seminal 2002 paper gave a novel unsupervised approach for classification of reviews based upon sentiment indicators. Sentiment indicators are part of speech (POS) phrases extracted from the document, usually combination of adjectives. Semantic orientation (SO) is calculated as the PMI-IR (Point wise Mutual Information-Information Retrieval) between the phrase in question and the word “excellent” minus the PMI-IR between the phrase in question and the word “poor” of all the extracted phrases of a document. Then the review text is classified on the basis of the average SO of the phrases.Pui et al. [5] gave a framework for concurrent mining of multiple time series and using the textual content as the primary source of prediction. Schumaker and Chen [6] used the following textual representations - bag of words,

noun phrases and named entities. This paper analyzed whether individual stock prices could be predicted with a twenty minute lag from release of an article. Cheng [7] transformed unstructured information into structured data and then extracted core phrases with the help of rough sets theories. Kaya and Karsligil [4] suggested a model in which a noun and a verb are taken as a word couple instead of using a single word. Feature selection techniques are applied in order to reduce the size of features. Deng et al. [8] gave a stock market prediction model based on SentiwordNet (SWN) to give scores of the sentiment indicators and finally derive the overall news sentiment. Hagenau et al. [9] represented text using highly expressive features and also incorporated exogenous market feedback for the selection of words. Instead of using only BOW (Bag of Words) approach, different types of features such as noun-verb combinations, 2-gram, 2-word combinations were used and the market feedback was utilized to label the news articles. Siering [10] trained a machine learning classifier using a lexicon based approach. The results were analysed on the basis of two categories i.e. domain dependent and domain independent. Rout et al.[11] have compared the two approaches for identification of sentiment from tweets and found the proposed approach to be more accurate than lexicon based approach. Zheng et al. [12] have done sentiment analysis of Chinese online reviews using N-char-grams and N-POS-grams and found that 4-POS-grams give best results. Naik et al. [13] have extracted tri-co-occurrences of sentiment words from publicly available movie review and university selection datasets and determined the semantic orientation of phrases. Rani et al. [14] have proposed a semi-supervised method for POS tagging that uses association rules to build a classifier model from a combination of a POS tagged corpus and untagged text data. The literature survey suggests that there is a scope to apply Turney's method in financial markets.

Background:

Credit derivatives have a wide range of products and we would be analysing a class of credit derivatives called Credit Default Swap (CDS). Credit Default Swap have a reference entity linked to them which are generally governments or corporations. The buyer has a credit asset with the reference entity and buys a CDS from the seller to insure himself against a default in the payment by the reference entity. It is thus used as a hedging tool to produce the risk associated with a credit asset. The buyer makes periodic payments to the seller till the date of the maturity of the contract and this constitutes the spread of the CDS. In the event of a credit default, the seller has to pay the buyer of the CDS the face value of the credit asset and all the interest payments that the buyer would have earned between that time till the date of the maturity of the asset. Credit default swaps are traded over the counter and hence there isn't much information available on it. The forms filed by the Mutual Funds regarding their CDS activities were in an unorganised manner before SEC has requested for more frequent and detailed fund holdings at the end of 2016.

Counterparty	Reference Entity/Obligation	Buy/Sell Protection	(Pay)/Receive Fixed Rate	Expiration Date		Notional Amount		Unrealized Appreciation/ (Depreciation)
Citibank N.A.	Georgia Pacific 8.125% due 5/15/2011	Buy	(3.55)%	12/20/10	$	1,030,000	$	(42,489)
UBS AG	CDX.NA.IG.6	Buy	(0.40)%	6/20/11	$	55,217,800		(74,344)
Morgan Stanley	CDX.NA.HY.6	Buy	(3.45)%	6/20/11	$	5,501,000		11,923
Morgan Stanley	CDX.NA.HY.6	Buy	(3.45)%	6/20/11	$	5,501,000		(25,759)
UBS AG	CDX.NA.HY.6	Buy	(3.45)%	6/20/11	$	5,501,000		(97,046)
Morgan Stanley	CDX.NA.HY.6	Buy	(3.45)%	6/20/11	$	5,501,000		(21,794)
Citibank N.A.	Windstream 8.125% due 8/1/2013	Buy	(1.60)%	9/20/11	$	2,885,000		(15,843)
UBS AG	Windstream 8.125% due 8/1/2013	Buy	(1.63)%	9/20/11	$	2,750,500		(16,547)
							$	(281,899)

Figure[]: Credit Default Swap reporting in a N-CSR report

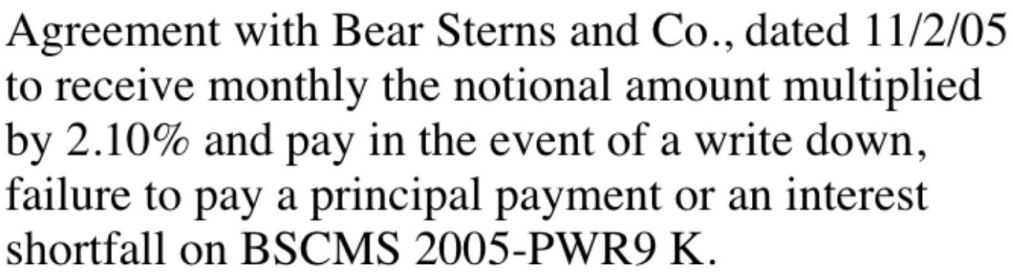
Credit Default Swaps on Corporate Issues—Buy Protection

Reference Entity	Counterparty	Implied Credit Spread at December 31, 2016[k]	Industry	Fixed Deal Pay Rate	Maturity	Notional[l]	Fair Value[d]	Unamortized Premiums Paid	Unrealized Appreciation (Depreciation)
Hovnanian Enterprises, Inc.	JPMorgan Chase Bank, N.A.	13.2%	Consumer Durables & Apparel	5.0%	9/20/19	$(5,000)	$859	$796	$63
Hovnanian Enterprises, Inc.	JPMorgan Chase Bank, N.A.	13.7%	Consumer Durables & Apparel	5.0%	12/20/19	(2,000)	388	371	17

Figure[]: Credit Default Swap reporting in a N-Q report

The length of each report could span hundreds of pages making it difficult and tedious to employ humans to gather information. This resulted in it being extremely difficult to get relevant information from these reports to carry out further analysis. Thus, information regarding CDS is extremely valuable as it would provide transparency and can be used to set appropriate capital requirements for financial institutions trading CDS. There have been a few previous studies exploring the usage of CDS by Mutual Funds [23], [24] but these reports examined only by a small number of the institutions over a short period of time. Hence, we choose to comprehensively examine all the reports from 2004-2017 and this makes the results of our project extremely

valuable for further research. In the area of Named Entity Recognition, the sequence labeling of sentences can be conducted using two methods, namely Bidirectional Long Short Term Memory - Conditional Random Field or a simple Conditional Random Field.



Figure[]: Unstructured sentence reporting of CDS in a report

However, when we successfully extracted raw (data before extracting the information) structured and unstructured CDS information, we could not find enough unstructured reporting of CDS to implement BiLSTM-CRF with high precision and recall. Therefore, this paper will focus on the implementation of a CRF model to extract information from unstructured sentences.

Use of CDS by corporate bond funds:
CDS has been used in multiple research papers to understand how multiple factors affect the maximum revenue that can be generated from investment in CDS. One such interesting paper discusses how complex strategies which involve a team of CDS team-managed funds outperforms a solo-managed team due to the abundance of diverse skill and opinions in the team. But at the same time they might perform poorly if the market conditions are extremely dynamic and quick decisions are required (Adam Gutter, 2015).

Mutual fund holdings of CDS:
A similar research to ours was conducted to understand the effects of CDS on pre and post financial crisis capital markets by using mutual funds holdings of CDS contracts between the period of 2007-2011 (Wei Zhu, 2016). The research concluded that reference entities which had the highest grossing interest revenue were actually the ones which deemed as too big to fail. The metadata present in the CDS reports enables research studies to come up with interesting and meaningful interpretations which in turn allow major investors in CDS to make much more informed decisions. To mitigate this, the study has described a three-layered deep learning architecture which consists of a single layer of character–level word representation using bidirectional LSTM(Long Short Term Memory, another layer of bi-directional LSTM for extracting the contextual meaning and final layer of CRF(Conditional Random Field) for predicting the label for each member in the sentence.

Bidirectional LSTM - CRF for Adverse Drug Event Tagging of Electronic Health Records
Sequence Tagging refers to the pattern-recognition problem which involves labelling each member of the sentence to a pre-assigned label. The labelling task itself can be treated as an independent classification problem for one member per task. Electronic Health Records are a great example of containing both structured and unstructured data which can be difficult to extract with just the use of rule-bases extraction tools like Regular Expressions and hence a need for a deep learning framework is required to learn from the patterns observed in the unstructured data.
To mitigate this, the study has described a three-layered deep learning architecture which consists of a single layer of character–level word representation using bidirectional LSTM(Long Short Term Memory, another layer of bidirectional LSTM for extracting the contextual meaning and final layer of CRF(Conditional Random Field) for predicting the label for each member in the sentence.

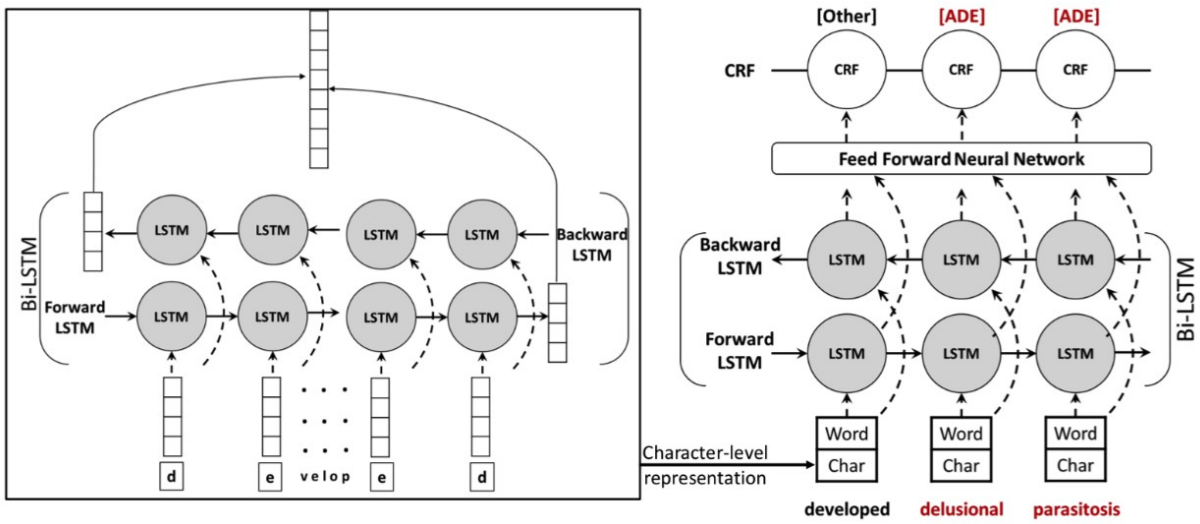


Figure []: Deep learning architecture used in the study

Similar to our methodology, the study has made use of various datasets to verify the implementation of their algorithm and then use their manually tagged 1,084 notes to train the model. Unlike our use case, the study was able to find pre-annotated datasets and hence required little to no data pre- processing. The evaluation set used by them consisted of 213 EHR(Electronic Health Records) notes and yielded them a F1 score of 0.8373, 0.8454, and 0.841 of phrase level execution. This goes to show that in order to extract unstructured data from such a large corpus would require a deep learning framework to yield accurate result of extraction. However, given the flexible and ever updating nature of our data, it is imperative for us to preprocess and weed out any data that is beyond the scope of the project which includes reporting of any asset class other than Credit Default Swap.

Time-series analysis:

The volume of data that is being structured in this process makes it a perfect use case of time series analysis. Time Series Analysis helps us find complex pattern in data which has time as one of its categorical features. Given the nature of data that we are analysing, we are trying to observe the data on a quarterly basis and therefore we can implement Long Short Term Memory (LSTM) on top of RNN to learn the long term dependencies of a specific categorical feature (Olah, 2015). This further enhances the memory state of the already existing LSTM and makes it great for language learning, handwriting recognition tasks which requires identifying key patterns in the reference object. After NLP, auto-encoders would allow us to derive key insights on the structured data and help us make useful predictions based on the patterns detected in the data. (Bansal, 2018)

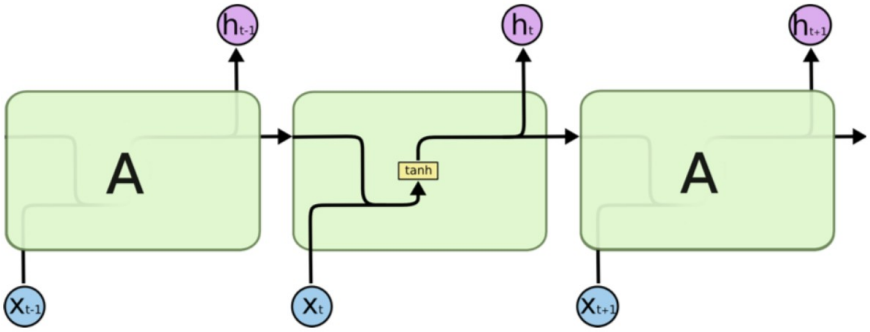
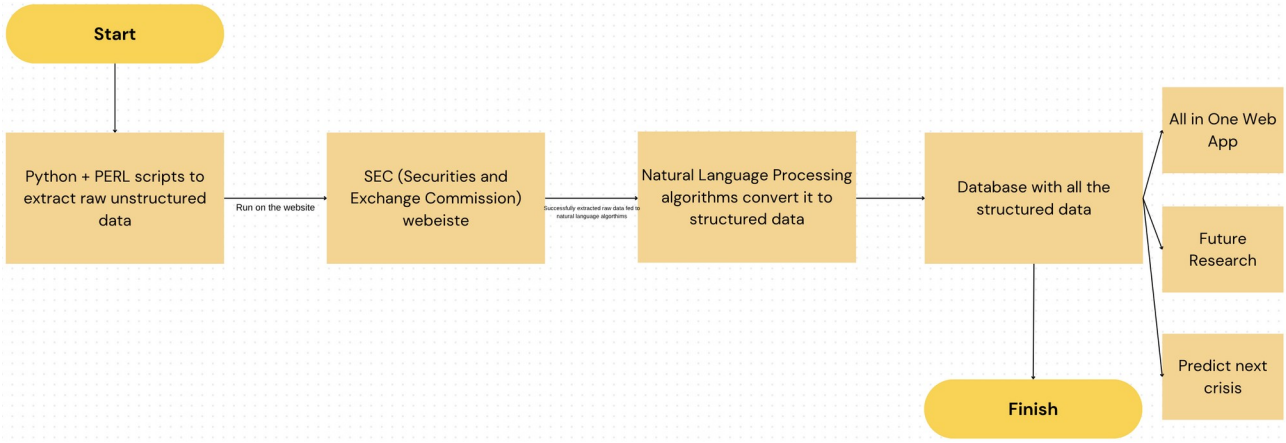


Figure []: Flow of information of a LSTM module

LSTM or Long Short Term Memory are a special type of Recurrent Neural Network which are capable of learning long-term dependencies. Traditionally, neural networks are not capable of remembering conclusions that they might have drawn from previous analyses (Colah, 2016). Each time you throw a problem at them, they start working on it from scratch. Thats where LSTMs play a vital role. It allows neural networks to remember the conclusions from previous analyses and use it in future downstream analysis.

Methodology:

The workflow for the entire project can be divided into three steps namely the data preprocessing, implementing Natural Language Algorithms and Downstream Analysis on extracted data. Figure [] gives an overview of all steps have been implemented. In nutshell, the web crawling is done using Python and PERL scripts to collect gigabytes worth of financial reports from the United States Securities and Exchange Commission website will include our first step namely data collection process. Then we will move on to our Natural Language Processing step which will help us convert the unstructured data collected into structured database. Furthermore, with the help of this database, we will conduct a time series analysis which will allow us to analyse the structured data and make useful predictions like predicting the next financial crisis.



This makes them a great tool for our use case as their implementation would allow us to remember the meanings of the words learnt from the word vectors that we figured in our language algorithms. Figure 2 goes through a simple repeating module of LSTM that shows us how the information in the LSTM module persists from one module to another.

Credit Default Swap Reporting Dataset: A 16813 rows dataset containing CDS reporting from 2004-2017 with categorical variables like Reference Entity, Reference Obligation, Reference Entity, Notional Amount, Expiration Date etc.

	CIK	Reporting Type	Reporting Year	Counterparty	Notional Amount	Reference Entity/Obligation	Expiration Date	Appreciation/Depreciation	Upfront Payments Paid/Received	Buy/Sell Protection	Description
0	0000315774	N-CSR	17	Morgan Stanley	5,000,000	Gatx Corp., 6.00%, 02/15/18	12/20/21	NaN	161,498	NaN	NaN
1	0000315774	N-CSR	17	Morgan Stanley	5,000,000	International Paper Co, 7.50%, 08/15/21	12/20/21	NaN	(44,764	NaN	NaN
2	0000883939	N-Q	09	Morgan Stanley	NaN	NaN	06/20/14	NaN	NaN	Buy	NaN
3	0001317146	N-CSRS	12	Morgan Stanley Capital Services, Inc.**	9000	NaN	6/20/16	16,340	-	Buy	NaN
4	0000837529	N-Q	07	Morgan Stanley International Limited	500000	Goldman Sachs International Hartford Financial Services Group Inc.	December 20, 2011	NaN	NaN	NaN	NaN
5	0001320615	N-CSRS	10	Morgan Stanley Capital	361,080	NR	12/20/10	NaN	NaN	NaN	NaN
6	0001320615	N-CSRS	10	Morgan Stanley Capital Services Inc	164,700	NaN	12/20/19	NaN	NaN	NaN	NaN
7	0001320615	N-CSRS	10	Morgan Stanley Capital Services Inc	$2,350,000	NaN	03/20/16	NaN	NaN	NaN	NaN
8	0001320615	N-CSRS	10	Morgan Stanley	5,000,000	NaN	12/20/15	NaN	NaN	NaN	NaN
9	0000883939	N-CSRS	10	Morgan Stanley	90,000	NaN	06/20/15	(8	(61	NaN	NaN
10	0000883939	N-CSRS	10	Morgan Stanley	NaN	NaN	12/20/15	(5	(120	NaN	NaN
11	0000315554	N-CSRS	09	Merrill Lynch International	500	Morgan Stanley /Abitibi-Consolidated, Inc.	Sep 2010	(1,980,450)	NaN	NaN	NaN
12	0000315554	N-CSRS	09	Goldman Sachs & Co.	$ 3,930	Morgan Stanley /American Airlines, Inc.	Sep 2012	(1,790,881)	NaN	NaN	NaN
13	0000315554	N-CSRS	09	Bank of America	3570	Morgan Stanley / AMR Corp.	Jun 2013	(1,943,439)	NaN	Sell	NaN
14	0000315554	N-CSRS	09	Barclays	500	Morgan Stanley /AMR Corp.	Jun 2013	(941,724)	NaN	Sell	NaN
15	0000315554	N-CSRS	09	Barclays BankAlcoa, Inc.	1500	Morgan Stanley / BoWater, Inc.			NaN	NaN	NaN
16	0001508782	N-CSR	13	JPMorgan ChaseCDX.NA.IG.20	200	Morgan Stanley /Delta Airlines, Inc.	4/24/2014	4147	4161	NaN	Put Option - OTC - Morgan Stanley Capital Services Inc., USD vs JPY
17	0001508782	N-CSR	13	JPMorgan ChaseCDX.NA.IG.20	200	Morgan Stanley /Delta Airlines, Inc.	4/24/2014	4147	4161	NaN	Put Option - OTC - Morgan Stanley Capital Services Inc., USD vs

Figure[]: Credit Default Swap Dataset with all the categorical variables

Restructuring the Data:
In order to build a corpus for our NLP tasks, it was important for us to restructure the file structure such that it is easy to navigate between different kinds of reports. We had around 146 GB of data in one SEC folder which contained all the Funds' N-CSR, N-CSRS and N-Q reports along with their CIK number. The initial folder structure was that of the main SEC-Edgar-Data folder containing all the folders with the Mutual Funds names which in turn had a folder named with the Funds' CIK number which then had the folders N-CSR, N-CSRS, N-Q in them which respectively contained the Funds financial reports from 2004-2017. After going through the data, it was found that few of the Mutual Funds had changed the Fund names through the time period and this resulted in there being duplicate files under different Fund names but with the same CIK number. We first wrote a script to restructure the data such that we could split it into 3 parts from which we could extract the data sequentially.

Algorithm 1: Folder Restructure Pseudocode

```
Result: Restructured Folder
initialization;
for all the directories in SEC-Edgar-Data folder do
    for all the directories in the Fund Name folder do
        if the directory is N-CSR then
            Move the entire sub-directory to the N-CSR Folder created
             outside SEC-Edgar-Data;
        end
        if the directory is N-CSRS then
            Move the entire sub-directory to the N-CSRS Folder created
             outside SEC-Edgar-Data;
        end
        if the directory is N-Q then
            Move the entire sub-directory to the N-Q Folder created
             outside SEC-Edgar-Data;
        end
    end
end
```

The above algorithm goes over the pseudocode of the script which was run on the original folder and it segregated the original folder into three folders namely N-CSR, N-CSRS and N-Q. Each of these folders contained a folder with the CIK number and inside that folder was the folder with the Mutual Funds name containing the respective reports. In the event that a Mutual Fund had changed its name during the time period of 2004-2017, then this restructuring made it easier for us to identify the Fund by its unique CIK number. After restructuring , the folder structure was as follows(similar structuring for N-CSRS and N-Q):

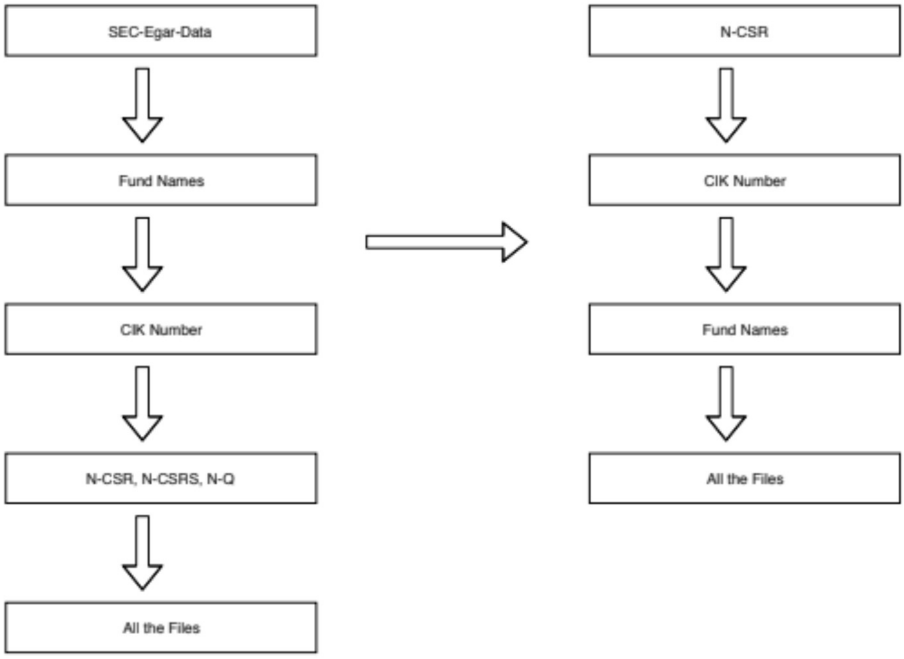


Figure 6: Updated File Structure

Figure []: Updated File Structure

The second part of the restructuring process was figuring out the reports which had unstructured data and the reports which had structured data which could be extracted. We could extract the structured data by running a Perl script which would extract the relevant CDS information from the reports. We wrote another script which would run the Perl script on all the files that we had.

It would then redirect the output to a temporary text file. We then checked the contents of the text file to determine if there was extracted information in the temporary text file. If there was information in it, we moved it to a new folder called Structured. If it was unstructured and the temporary text file didn't have relevant information in it, we moved those reports to the unstructured folder. This way, we had a smaller set of folders on which to run our NLP model.

Algorithm 2: Data Categorization

```
Result: Data categorized into structured and unstructured format
initialization;
for all the directories in N-CSR folder do
    for all the directories in the CIK Number do
        for each file under Fund Name do
            Run the perl script on the file and redirect output to a
             temp.txt file;
            if there is required output in the temp.txt file then
                Move the original file to Structured folder & remove
                 temp.txt file;
            else
                Move the original file to unstructured folder & remove
                 temp.txt file
            end
        end
    end
end
```

We then unzipped all the files to their original text files. After going through the data, we found out that along with files in there which had CDS information, there was also files which didn't have any mentions of Credit Default Swap. We hence decided to run another script to remove these files and cut down the size of our dataset to only the useful reports to speed up the processing time. This script checked all the text files for mentions of the word Credit Default Swap and other relevant words and when it found it, moved the respective text file to a new folder called Corpus N-CSR(for N-CSR folder) where it also appended the CIK

number along with N-CSR to the file name to make it easier for us to identify the files during future processes. This was similarly done for the N-CSRS and N-Q folders. The folder structure was as follows:

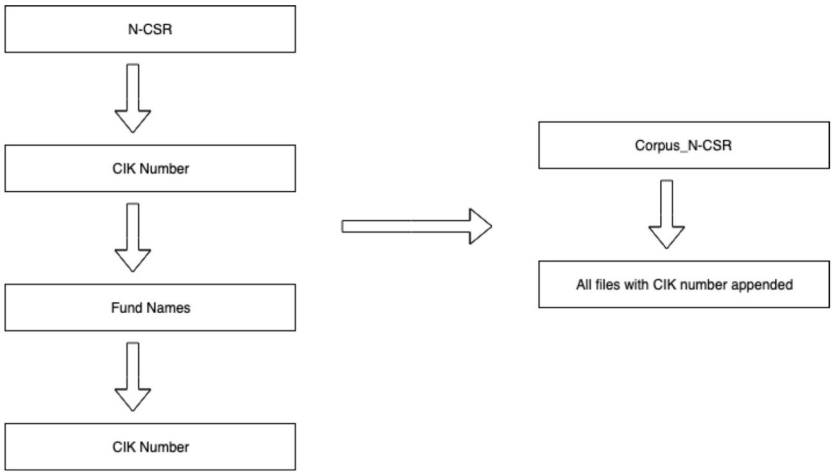
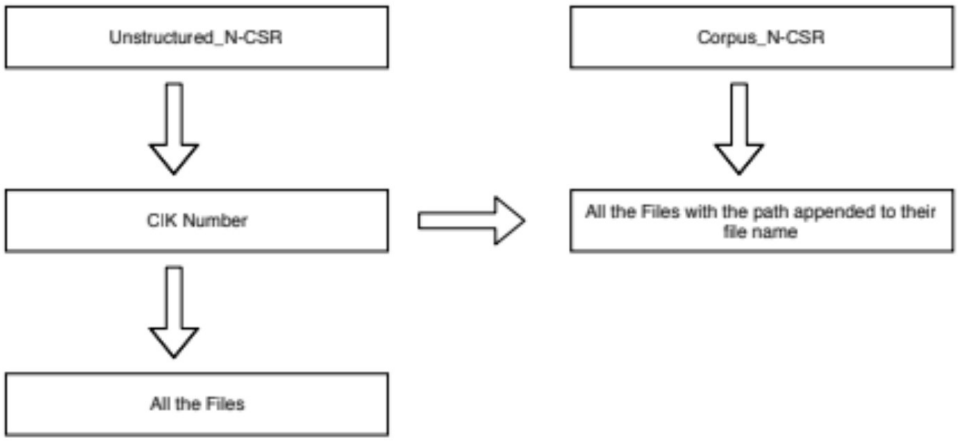


Figure []: Corpus folder structure

Folder Structure for the Corpus Folder which contains all the files that we are going to use: The above scripts were applied to N-CSR, N-CSRS, N-Q.



Algorithm 3: Corpus Generation

```
Result: Final Corpus Created
initialization;
for all the directories in unstructured N-CSR folder do
    for each file under CIK Number do
        Run the perl script on the file and redirect output to a temp.txt
         file;
        if there are mentions of Credit Default Swaps or relevant
         information then
            Move the original file to Corpus N-CSR & rename file to
             include the path in the name;
        else
            Move the original file to No_CDS folder;
        end
    end
end
```

We hence had 3 separate CDS folders at the end containing the useful reports of N-CSR, N-CSRS and N-Q. Once we had these folders, we had to start tagging these reports manually to be able to run our NLP model

on them. To make the tagging process easier, we built a Text Annotation tool using Django with Python which would allow us to import our text files, tag the files according to the labels that we created which would be the Reference Entity, Counterparty, Notional Amount etc. We could then export the tagged data into a csv file.

Table Extraction:

Once we got the folders containing the N-CSR, N-CSRS, N-Q reports, we found out that most of the reports had the CDS information in the form of tables. Hence it was necessary for us to figure out a way to extract these tables out of the reports. The reports had these tables enclosed within HTML tags such as the and <table>tags. We decided to use beautiful soup with python and extract these tables out from the reports using the HTML tags. We wrote a python script to help us do this.

```
def append_classID(filepath):
    """
    Append the classID to the p or div(or any tag found in future inspection)
    tags which contain the different ways of calling CDS and returning the respective
    tag counters for further processing
    """
    # Reading Files
    f = open(filepath, 'r')
    data = f.read()
    f.close()

    # Making soup
    soup = bs(data, "lxml")
    soup.prettify()
    # Adding multiple reporting styles used in reoprts to mention CDS information
    searchtext = ["Credit Default", "CDS Contract",
                  "Default Swap", "Default Contract", "Default Protection", "Credit Derivative",
                  "credit default swap", "credit default"]

    searchtext_pageTable = ["NOTIONAL",
                            "REFERENCE ENTITY", "COUNTERPARTY", "EXPIRATION"]

    p_counter = 0
    div_counter = 0
    table_counter = 0

    # Find the first <p> tag with the search text
    all_p_tags = soup.find_all("p")
    all_div_tags = soup.find_all("div")
    all_caption_tags = soup.findAll("caption")

    # Renname all <page> tags to <div> since there is no such thing as a <page>

    plengthFoundText = len(all_p_tags)
    divlengthFoundText = len(all_div_tags)
    captionlengthFoundText = len(all_caption_tags)
```

Figure []: Table Extractor – 1

The above figure shows the first part of the first method that is called in the script. Here, we take the Financial Report with the HTML tags in it and then use beautiful soup to first make soup out of all the HTML tags. We then specify the words that the tables should contain in order for us to extract them. Furthermore, we find all the p, div, caption and store it in an array which we will later iterate through to find the respective tables within these tags. We have counters for the different types of tags as a way to count the number of tables being extracted from each type of tag.

```
    if captionlengthFoundText > 0:
        print("Length of captionLengthFoundtext is: ", captionlengthFoundText)
        for b in range(captionlengthFoundText):
            word_counter = 0
            for a in range(len(searchtext_pageTable)):
                if searchtext_pageTable[a] in all_caption_tags[b].text:
                    print("Word Found in Caption Table: ",
                          searchtext_pageTable[a])
                    word_counter += 1

            if word_counter > 2:
                print(all_caption_tags[b].text)
                table_counter += 1
                all_caption_tags[b]['class'] = table_counter

    if plengthFoundText > 0:
        print("Length of pLengthFoundtext is: ", plengthFoundText)
        for i in range(plengthFoundText):
            for k in range(len(searchtext)):
                if searchtext[k] in all_p_tags[i].text:
                    p_counter += 1
                    all_p_tags[i]['class'] = p_counter
                    break

    if divlengthFoundText > 0:
        print("Length of divLengthFoundtext is: ", divlengthFoundText)
        for j in range(divlengthFoundText):
            for l in range(len(searchtext)):
                if searchtext[l] in all_div_tags[j].text:
                    div_counter += 1
                    all_div_tags[j]['class'] = div_counter
                    break

    print("The value of p_counter is: ",  p_counter)
    print("The value of div_counter is: ", div_counter)
    print("The value of table_counter is: ", table_counter)
    return soup, p_counter, div_counter, table_counter
```

Figure []: Table Extractor -2

In the above picture, is the second half of the first method that is called where we first check if there exists each of p, div, caption tags. We iterate through the array containing the text inside each of these tags and also iterate through the keywords that we want our tables to contain and compare them. If we find that, for example, in a certain p tag, we find a mention of one of our keywords, we then increase the value of the p counter and also append the value of the p counter as classID. We similarly do this for the other two tags. This method then returns all these values which will be used as the input arguments for the next method.

```
def get_tables(soup, p_counter, div_counter, table_counter):
    """
    Extracts each table on the page and places it in a dictionary.Converts each dictionary to a Table object. Returns a list of
    pointers to the respective Table object(s).
    """
    table_list = []
    space = re.compile(r"\s+")  # used for RegEx Fixed Length to CS purposes
    counter = 0
    # Extracting tables after a certain p tag
    for iterator in range(1, p_counter+1):
        # Find the first <p> tag with the search text
        table_tag = soup.find("p", {"class": str(iterator)})
        # Find the first <table> tag that follows it
        table = table_tag.findNext("table")
        # empty dictionary each time represents our table
        table_dict = {}
        rows = table.findAll("tr")
        # count will be the key for each list of values
        count = 0
        for row in rows:
            value_list = []
            entries = row.findAll("td")
            for entry in entries:
                # fix the encoding issues with utf-8
                if entry.find("p"):
                    entry = entry.find(
                        "p").text.encode("utf-8", "ignore")
                    strip_unicode = re.compile(
                        "([^-_a-zA-Z0-9!@#%&=,/'\";:~`\$\^\*\(\)\+\[\]\.\{\}\|\?\<\>\\]+|[^\s]+)")
                    entry = entry.decode("utf-8")
                    entry = strip_unicode.sub(" ", entry)
                    value_list.append(entry)
                else:
                    entry = entry.text.encode("utf-8", "ignore")
                    strip_unicode = re.compile(
                        "([^-_a-zA-Z0-9!@#%&=,/'\";:~`\$\^\*\(\)\+\[\]\.\{\}\|\?\<\>\\]+|[^\s]+)")
                    entry = entry.decode("utf-8")
                    entry = strip_unicode.sub(" ", entry)
                    value_list.append(entry)
            # we don't want empty data packages
            if len(value_list) > 0:
                table_dict[count] = value_list
                count += 1
        table_obj = Tables(table_dict)
        table_list.append(table_obj)
        print("Number of p_tables done: ", iterator)
```

Figure []: Extracting tables based on search text

The above method is called next wherein we iterate through the list of p tags which contain the words and find the table tag and then go through the rows of the table and extract all the contents. We then return the list of tables at the end. Once we get the list of tables, we save the tables under the given file name in csv format. It is similarly done for the other tags.

Formatting of the data:

checkTable.sh - We then wrote a bash script to run this on all the reports in the three folders. The folders from which we could extract tables were put into a separate structured folder and the rest of the files were put in an unstructured folder. This resulted in there being 3 structured folders each containing the tables from each report.

Format.py - Once we had the all the useful tables in different folders, we started working on formatting the tables. We found after going through the data that most of the tables had a different format for the data. This meant that we had to go through the csv files manually, try and identify a pattern which was followed by the majority of the tables and then try and write a python script to format those kinds of scripts.

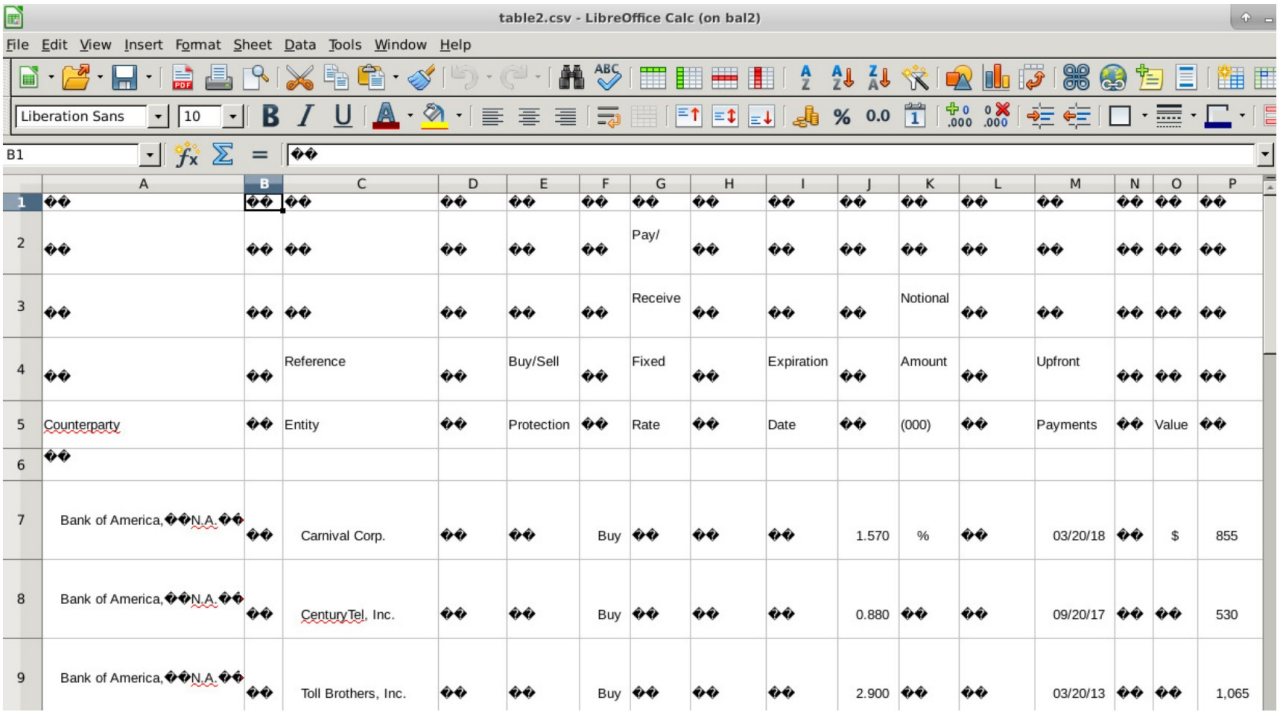


Figure [] : Example 1 of an unformatted table

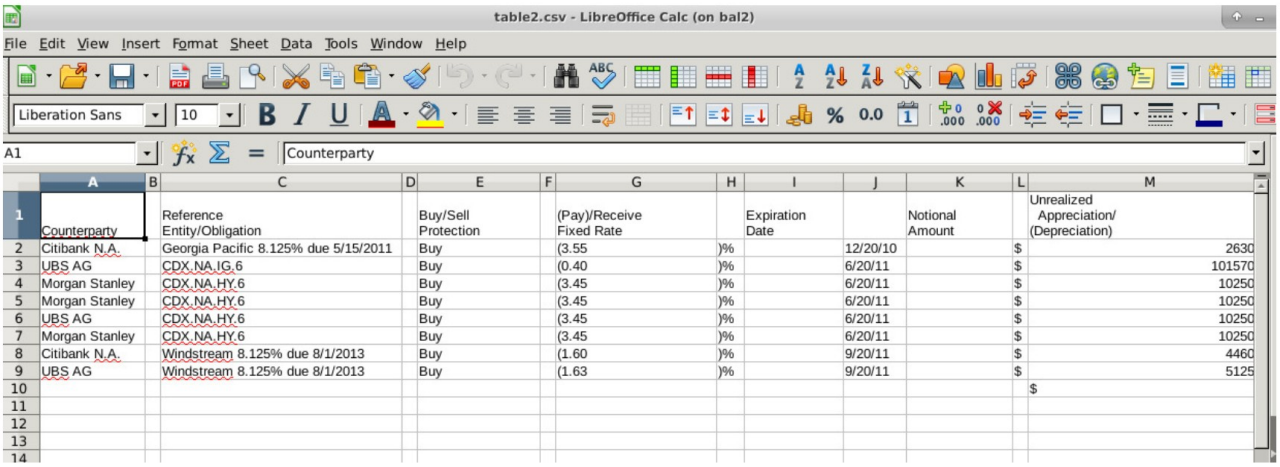


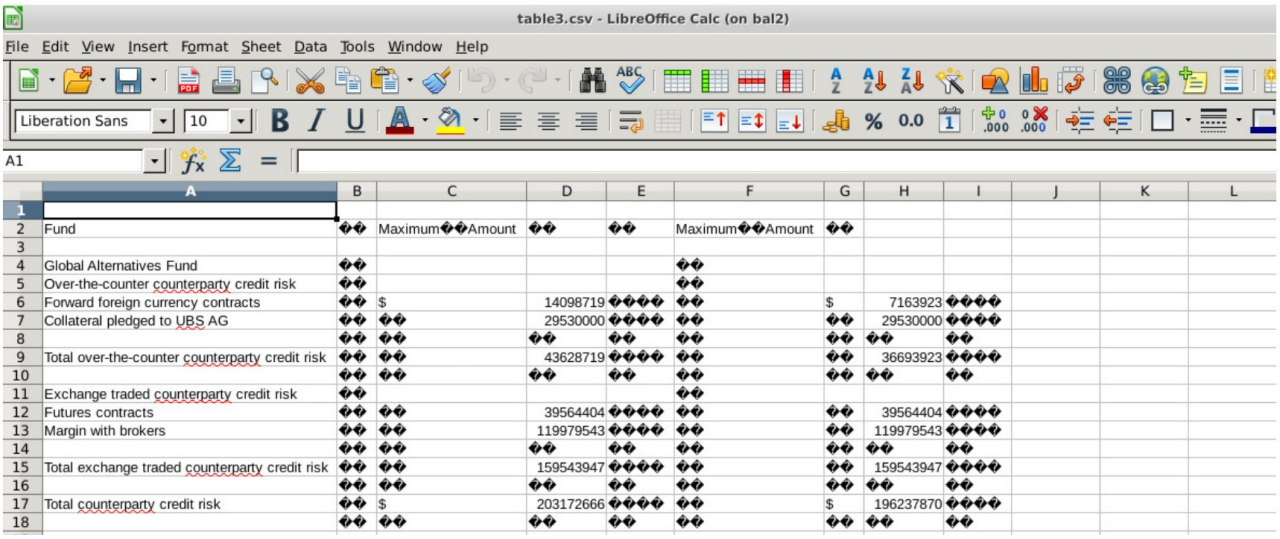
Figure []: Example 2 of an unformatted table

Above two figures show two examples of the types of format that the tables initially have. The other files may have a completely different format. We identified in majority of the files, there were a lot of random characters and spaces present in the cells in the csv files between the cells which actually had the information. We wrote a python script to eliminate all of these random characters and shift the cells with the actual information.

```
import csv
import sys

in_file = sys.argv[1]
out_file= sys.argv[2]

row_reader = csv.reader(open(in_file, "rt", encoding="utf-8"    ))
row_writer = csv.writer(open(out_file, "wt", encoding="utf-8"))

for row in row_reader:
    i=0;
    new_row=[]
    for val in row:
        if not val or val == '\xa0' or val == '$' or val == ')%' or val ==')' or val == '\xa0\xa0'
        or val == '\n\xa0\n' or val == '\xa0\n' or val == '\n\xa0' or val == '\n$' or val == '\n) $'
        or val == '\n)' or val == '\n)%' or val == ')\xa0' or val == 'USD' or val == '\n    0\n'
        or val == '\n    $\n'or val == '\n    )\n' or val == '\n    %\n' or val == '\n\xa0\t'
        or val == '\n\n' or val == '\xa0\n\t' or val == '%' or val == ' ' or val == '\nUSD\n'
        or val == '1' or val == '2' or val == '3' or val == '4' or val == '5' or val == '6' or val == '7'
        or val == '8' or val == '\nUSD\n' or val == '\xa0USD' or val == '\n\xa0USD\n' or val == '\n$\n'
        or val == '\n)\n' or val == '\n%' or val == '\nUSD ' or val == '\n$'
        or val == '\xa0\xa0\xa0\xa0\xa0\xa0$' or val == '\xa0\xa0\xa0\xa0\xa0\xa0\xa0\xa0$'
        or val == '\xa0$' or val == ' \xa0' or val == '\nUSD ' or val == '\n%'
        or val == '\nJPY ' or val == '\n ' or val == '\xa0 '
        or val == '$\xa0\xa0\xa0\xa0\xa0\xa0\xa0\xa0\xa0\xa0\xa0\xa0\xa0\xa0\xa0\xa0 '
        or val == '\xa0 ' or val == '\nRUB ' or val == '\nCHF ' or val == '\nCOP ' or val == '\nMXN '
        or val == '\nUSD  ' or val == '%\xa0' or val == '\nJPY' or val == '\nGBP' or val == '\nCHF'
        or val == '\nUSD' or val == '\n%+'
        or val == '\xa0\n      ' or val == '\n%)' or val == '\xa0\xa0\xa0' or val == '\xa0\xa0\xa0\xa0'
        or val == '$ '
        or val == '\xa0\xa0\xa0\n' or val == ')%\xa0' or val == '\n   ' or val == 'Currency' or val == '\n'
        or val == '\xa0\n\n\t' or val == '%)' or val == '(13)' or val == '(11)' or val == '($'
        or val == '\n\xa0\n      \n' or val == '\n%\n'
        or val == '\n)\xa0\xa0\n' or val == 'USD\xa0' or val == ') $' or val == '\xa0\xa0\xa0\xa0\xa0'
        or val == '\xa0\xa0\xa0\xa0\xa0\xa0\xa0' or val == '\n\xa0 ' or val == '\n) ' or val == '\n$ '
        or val == '\n% ' or val == '\n\n\xa0  ' or val == '\n\nUSD  ' or val == ' \n\xa0\n\n'
        or val == '\n\n                          \xa0\n'
        or val == '\n\n\nUSD\n\n' or val == '\n\n\n                                \xa0\n\n'
        or val == '\n)%\n' or val == '\xa0\xa0\xa0\xa0\xa0\xa0' or val == '\n\n\t\t\t\t\t\xa0\n'
```

Figure []: Showcasing a portion of the different types of an unwanted characters that we remove

Above figure represents a portion of the python script used to format the tables. The script works by taking in the unformatted file as the first argument and another filename which will contain the formatted table as the second argument. We loop over each row in the csv file and initialize an array which will contain values for the formatted row. We then go through each cell in the row and check if the character matches any of the random unwanted characters or spaces and if it does, we just ignore it. We initialize a counter i to 0 in the beginning of the loop and this gives us the index of the value and just increment it by 1, after checking if doing this won‘t exceed the length of the row, to ignore the unwanted characters. If the cell has useful information, we append it to the new empty array which we initialized and then incre- ment the value of i. The unwanted characters which we are checking have all been hardcoded after going manually through all the reports as it would have otherwise been impossible to identify all the different types of these characters. After it goes through all the rows, we have another check in place to see if the length of the row after removing all the unwanted character exceeds 3 and only if this condition is true, we write it to the new file. This check is in place to help us eliminate those rows and csv files which dont actually have any useful CDS values that we require. An example of this is shown in the image below.



	A	B	C	D	E	F	G	H	I	J	K	L
1												
2	Fund	**	Maximum**Amount	**	**	Maximum**Amount	**					
3												
4	Global Alternatives Fund	**				**						
5	Over-the-counter counterparty credit risk	**				**						
6	Forward foreign currency contracts	**	$	14098719	****	**	$	7163923	****			
7	Collateral pledged to UBS AG	**	**	29530000	****	**	**	29530000	****			
8		**	**	**	**	**	**	**	**			
9	Total over-the-counter counterparty credit risk	**	**	43628719	****	**	**	36693923	****			
10		**	**	**	**	**	**	**	**			
11	Exchange traded counterparty credit risk	**				**						
12	Futures contracts	**	**	39564404	****	**	**	39564404	****			
13	Margin with brokers	**	**	119979543	****	**	**	119979543	****			
14		**	**	**	**	**	**	**	**			
15	Total exchange traded counterparty credit risk	**	**	159543947	****	**	**	159543947	****			
16		**	**	**	**	**	**	**	**			
17	Total counterparty credit risk	**	$	203172666	****	**	$	196237870	****			
18		**	**	**	**	**	**	**	**			

Figure []: Example of table containing useless CDs information

We see that this table came to be part of the current data set as it has the word counterparty in it and hence, it passed the first shell script. It however, doesnt contain any useful information that we require, and as it would have only 3 columns after formatting, with the final if condition, none of the rows would get written to the output file and hence it would only give a blank csv file as the output.

Blank.sh -

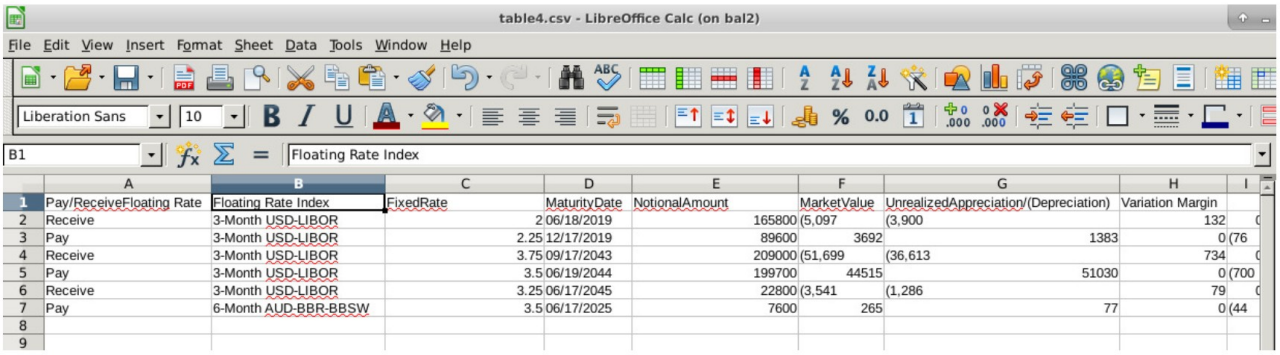
We run this script on the formatted csv files after running the format.py script on all the folders. This script is to delete all the blank csv files that are present after formatting. It first runs a python script on all the files and it outputs the word empty if the csv files are blank. The script then uses the command grep to check the output file of the previous python script and deletes the csv file if it finds the word empty.

Formatword.py -

This python script was written to remove those rows in the csv files which contained the word Total. This was done as after going through the data, it was found that most of the CDS tables had at the end of the table a row containing the Total Credit Default Swaps and Total Unrealized Apprecia- tion(Depreciation) etc with corresponding values. This row was not needed, and it was necessary to remove this row in all the tables in order to make a neatly formatted unified csv file at the end. There were also tables which didn‘t have Total or a description written in the last row but just had the values of the total credit default swaps. These rows were however already removed using the last if condition in the format.py script and hence, we dont have to do anything additional to them.

Libor.sh -

After running all the scripts above, we get a formatted table, but we also realized that along with Credit Default Swap tables, there were also a few tables containing Interest Rate Swaps and some other tables containing the overall figures for CDS being extracted and formatted. This was because the reporting style for those tables were similar and they had the same column headings as a CDS table. It was hence necessary to remove these csv files containing this other information.



	A	B	C	D	E	F	G	H	I
1	Pay/ReceiveFloating Rate	Floating Rate Index	FixedRate	MaturityDate	NotionalAmount	MarketValue	UnrealizedAppreciation/(Depreciation)	Variation Margin	
2	Receive	3-Month USD-LIBOR	2	06/18/2019	165800	(5,097	(3,900	132	(
3	Pay	3-Month USD-LIBOR	2.25	12/17/2019	89600	3692	1383	0	(76
4	Receive	3-Month USD-LIBOR	3.75	09/17/2043	209000	(51,699	(36,613	734	(
5	Pay	3-Month USD-LIBOR	3.5	06/19/2044	199700	44515	51030	0	(700
6	Receive	3-Month USD-LIBOR	3.25	06/17/2045	22800	(3,541	(1,286	79	(
7	Pay	6-Month AUD-BBR-BBSW	3.5	06/17/2025	7600	265	77	0	(44
8									
9									

Figure []: Example of an interest rate swap table extracted which we need to filter
This script searches for keywords such as LIBOR which is used to report Interest rate swaps in the Floating Rate Index column as seen in the above image, in the csv files and also other words such as Convertible Bonds and Forward which signifies Forward swaps. It then deletes the files which contains any of these words. This helps us to get a data set containing only Credit Default Swap tables and information.

Emptydir.sh - This is a simple script which is run on all the folders after all the above scripts have been executed. It uses the find command to check if any of the folders are empty and then removes those folders. This was done as the blank.sh and libor.sh script would be removing the useless csv files and there were many instances in which the folder after that would be empty as all the csv files in it were useless. Hence it was necessary to remove these empty folders.

Master.sh -
This is a master script which runs all the above scripts together one after the other. It runs the following scripts in order
- pythonExtractor.py script on all the files
- checkTable.sh
- format.sh (shell script to run the format.py script on all the csv files)
- blank.sh
- formatword.sh (shell script to run the formatword.py script on all the csv files)
- libor.sh
- Emptydir.sh

We can run this script on any file and it will extract and completely format the CDS tables and output them. The accuracy of this script was 76% (running this script on 100 random csv files containing the tables led to 76 files being completely formatted). The rest of the 24 files were around 90 percent formatted with there being some discrepancies in the way the column headers were reported. To ensure that the database consisting of all the tables was accurate and to help us build the unified csv file, we also manually went through each file and corrected the formatting of those files which weren't fully formatted by the scripts.

Unified CSV file: The next step in the process is to take all of the data that we have and to merge all of it into one CSV file together. This would be helpful to present the data, for future research and also to help us do the time series analysis. The first challenge in merging the data was that there were different ways in which the column headers were written. The header for Notional Amount could be "Notional Amount Due on Default" in one report, "Notional Value" in another or even "Notional Amount (000s)" in another report. Hence, we first gauge the different type of headers possible in all the reports and then come up with a master list containing the headers for the unified CSV file.

Generate conditions for each type of Header

In this section, we will decide the final size of the unified CSV that we are going to generate for each time user searches for something.

```
column_headers = ["CIK", "Reporting Type", "Reporting Year", "Counterparty", "Notional Amount", "R
eference Entity/Obligation", "Fixed Rate", "Expiration Date", "Appreciation/Depreciation", "Upfron
t Payments Paid/Received", "Implied Credit Spread", "Buy/Sell Protection", "Description"]
unified_csv = pd.DataFrame()
CIK_csv = pd.DataFrame(columns=["CIK"])
Reporting_Type_csv = pd.DataFrame(columns=["Reporting Type"])
Reporting_Year_csv = pd.DataFrame(columns=["Reporting Year"])
Counterparty_csv = pd.DataFrame(columns=["Counterparty"])
Notional_Amount_csv = pd.DataFrame(columns=["Notional Amount"])
Reference_Entity_csv = pd.DataFrame(columns=["Reference Entity/Obligation"])
Fixed_Rate_csv = pd.DataFrame(columns=["Fixed Rate"])
Expiration_Date_csv = pd.DataFrame(columns=["Expiration Date"])
Appreciation_csv = pd.DataFrame(columns=["Appreciation/Depreciation"])
Upfront_Payments_csv = pd.DataFrame(columns=["Upfront Payments Paid/Received"])
Buy_Sell_csv = pd.DataFrame(columns=["Buy/Sell Protection"])
Description_csv = pd.DataFrame(columns=["Description"])
```

Figure []: Code containing the master list for the unified csv

We also normalize the values in the Notional Amount column such that every time its reported in (000s), we add (000) to the value present in the column. The below image illustrates the code to achieve this.

```
def normalizeValues(filelist):
    for filename in filelist:
        with open(filename, 'r') as f:
            df = pd.read_csv(filename, engine='python', dtype=str)
            headers = list(df)
            for header in headers:
                if '000' in header:
                    df[header] = df[header].astype(str) + '000'
            print (df)
            print("Filename", filename)
            df.to_csv(filename, index=False, header=True)
```

Figure []: Code for normalization

We then merge all the csv files and perform basic data cleaning and add NaN for cells which don‘t contain values. We end up with a total of 16813 rows after merging all the files.

Tagging of unstructured data for natural language processing:

After extracting and formatting the structured data, we move on to preparing the unstructured data for Natural Language Processing. We first need to remove all the HTML tags from the reports in order for us to get only the text out to be used for the NLP. We wrote a basic python script which would strip out all the HTML tags and then ran it on all the unstructured folders using a bash script. Once we got only the text from the report, we realized that we would run into a class imbalance problem as we would have a lot of text which would not be tagged along with a few useful unstructured sentences containing information about CDS in the report. We hence decided to write a python script using regex and substring to extract only those sentences which contained relevant information as this would greatly help increase the accuracy while training the data after tagging. Below is an extract of the script wherein we have defined a set of words and only extracted the sentences which contains these words.

```
import string
import sys
import re

from nltk.tokenize import sent_tokenize
from nltk.tokenize import word_tokenize

# reading and tokenizing the file
arguments = sys.argv[1:]
filename = arguments[0]
f = open(filename).read()
f = re.sub(r'\-+', '.', f)
f = re.sub(r'\=+', '', f)
f = re.sub(r'\(+', '', f)
f = re.sub(r'\)+', '', f)

sentences = sent_tokenize(f)
my_sentence = []

# defining word list

word_list = ["notional amount", "Notional Amount", "Pay", "Receive", "Counter Party", "Reference
Entity", "reference entity"]


def sentenceFinder(sentences, word_list):
    for word in word_list:
        for sent in sentences:
            if word in sent:
                my_sentence.append(sent)

    return my_sentence


output = sentenceFinder(sentences, word_list)
```

Figure18: Code snippet of script to extract unstructured sentences containing keywords

Once we had the unstructured sentences, we could begin the tagging of the data. As this a cumbersome process, extreme precision and proper formatting of output data(post tagging) needs to be present in order to ensure higher success rate of the manual work being put. Keeping these as our requirements, we built a Text Annotation Tool based on the Django framework which allows us to upload and tag datasets manually. It supports up to 26 different entity labelling.

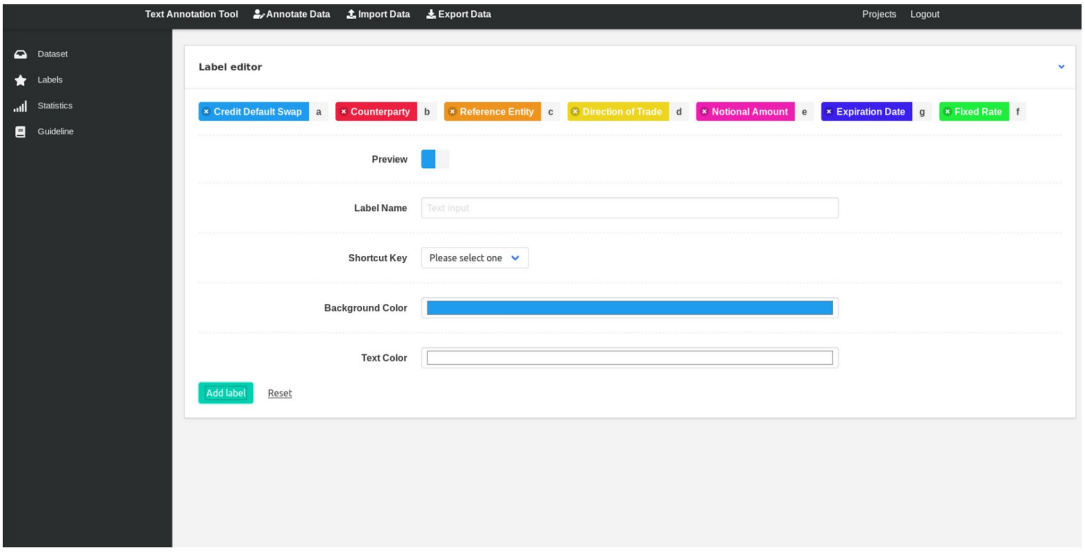


Figure []: Defining the labels

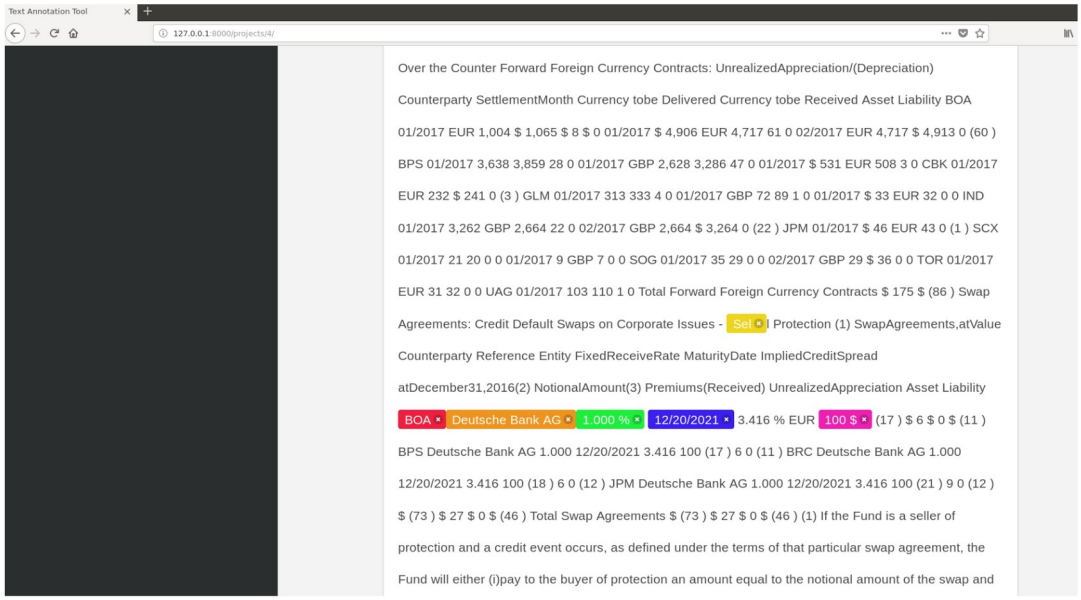


Figure []: Tagging

Finally, it gives us a graphical user interface (GUI) for tagging datasets and helps us automatically format the manually tagged dataset into our desired format. Upon successfully tagging a report, you can export the final tagged report to contain sentences in either a .csv or .json format and download them. These tagged files are then used to train the CRF models which is discussed in the NLP report.

Results:

Corpus generation and table extraction:

We had reduced the size of the original data set from 146 GB to 71GB (44GB for N-CSR, 15 GB for N-CSRS, 12 GB for N-Q) which is a 51.37 % reduction in size and made a corpus of the folders. We then ran our script to extract tables, the results of which are shown below.

Type of report	**Total number in corpus**	**Total Extracted as Structured**
N-CSR	7750	5198
N-CSRS	3304	2168
N-Q	3670	2720

Table []: Number of reports which have structured data

To determine the accuracy of the script in extracting the data, we manually went through 100 random reports in each of the N-CSR, N-CSRS and N-Q folders and checked whether all the information was extracted.

Type of report	**Number of reports Checked**	**Data accurately extracted**
N-CSR	100	93
N-CSRS	100	89
N-Q	100	92

Table []: Accuracy test

We take a weighted average of all three folders and see that the overall accuracy of the script is 91.33%. Since the accuracy of the script is dependent on the HTML tags also, it might have been the case that due to some inaccuracies in the tags, the accuracy has gone down. We then created a master script which would take in a financial report and extract and format the CDS tables and return it.

CRF Classifier for CDS Reporting Dataset - An implementation of Conditional Random Field for the CDS dataset developed as well as techniques for hyper-optimisation using 3-fold cross-validation.

Information extraction has often been tackled with rule-based approach extraction which includes scripting with a wide range of rules accounting for every possible combination of a sentence in order to extract the required information. For example, Sheikh and Conlon, employed a rule-based approach by studying a sample space of financial documents and developing rules to extract them (Sheikh and Conlon, 2010). These efforts often led to high precision and recall immediately. However, when presented with newer formats of similar documents and dealing with high linguistic variety, the performance was lack-luster. Furthermore, the amount of manual effort going into rule-based approaches can be justified for a small and known sample of documents. But they do not account for a small and known sample of documents. But they do not account of

minor variations in input data which leads them to not being able to extract information when presented with new information. However, it is important to note that it is not completely possible to eliminate rule-based approaches. Many a times, this approach is required to be employed if we are dealing with structured data or data with a certain set of formats. This forms a perfect use case for Named-Entity Recognition systems, specifically sequence labelling. Since sequence labelling leans the features of the entities or words to be extracted in the sentence, it allows us to employ this technique to the kind of data where the variations in the data cannot be accounted for by rule-based approaches. Sequence Labelling involves the task of assigning a single label to each element in each sentence. This element could be a word or a group of words. The labels could be parts of speech tags or predefined labels such as one given in figure [].

```
John  lives in New   York  and works for the European Union
B-PER O     O  B-LOC I-LOC O   O     O   O   B-ORG    I-ORG
```

Figure[]: Example of a sentence with its corresponding labels (Li, 2018)

Here, the entities are LOC, ORG, PER, and MISC for location, organisation, person and miscellaneous. The no-entity tag is shown by O tag which means that specific element has no label. Because some entities (like New York) have multiple words, we use a tagging scheme to distinguish between the beginning (tag B-..), or the inside of an entity (tag-I..) . So sequence labelling can be treated as a combination of classification tasks where the algorithm classifies each element in the sentence into a label from a predefined set of labels. The accuracy of this algorithm can be greatly improved by designing a sequence labelling algorithm which takes into account the features of its nearby elements and then classifies the current element into one of the labels. This solves the problem of rule-based methods being unable to connect to account for variations in input data. Sequence labelling algorithms are mostly based on probabilistic or deep learning methods. The probabilistic method like Conditional Random Fields (CRFs) and Hidden Markov Models (HMMs) assign labels based on the probability of a particular tag sequence occurring. Deep Learning methods involve using recurrent neural networks which allow retaining contextual information of the sentence and at the same time retaining the information of distributional representations of the sentence. Due to this, deep learning methods often prove to be more accurate in assigning labels to each element in the sentence. However, owing to the low number of the unstructured sentences in our use case, deep learning methods cannot be employed on our dataset. Therefore, probabilistic methods are a better fit for our problem statement.

Conditional Random Fields

Probabilistic classifiers mainly fall into two categories, discriminative and generative classifiers models. The significant difference between discriminative and generative is that discriminative models model conditional probability distribution i.e. P(y|X) while the generative models try to model a joint probability distribution, i.e., P(X,Y) (Sutton, 2010). Our use case requires us to account of elements nearby the current element that we trying to classify, so it is imperative for us to consider conditional probability distribution instead of joint probability distribution. The objective of a sequence labelling problem is to find the probability of a sequence of labels (y) given an input of sequence of vectors (X). This probability is denoted by P(y|X). This makes Conditional Random Fields the perfect tool to serve our purpose. Lets assume that the training set consists of input and target sequence pairs (Xi, *yi*). The *ith* sequence of vectors is *Xi* = [*x1, ...xl*]. The *ith* target sequence of labels is *Yi* = [*y1,...yl*] and *l* is the length of the sequence. For a general mathematical understanding, lets assume that for sample (X,y), in a standard sequence labelling problem using classification we compute P(y|X) by taking the product of the probability of element at the nth position in the sequence where 1 <= n <= l.

$$P(y|X) = \prod_{k=1}^{l} P(y_n|x_n)$$

can be expanded to

$$P(y|X) = \frac{exp(\sum_{n=1}^{l} \bigcup(x_n, y_n))}{\prod_{n=1}^{l} Z(x_n)}$$

The expanded equation is essentially modelling P(yk|Xk) with a normalized exponential. This imitates the softmax operation widely used in neural networks and mimics its output (Lafferty & Macallum, 2001). Also here, U(x,y) is known as emissions scores which is essentially the score generated for a label y given the x vector at nth timestep. The x vector in practice is the concatenation of the surrounding elements to the element that we are considering. Z(x) is known as the partition function which is normalisation factor since we would want the total probability to be equal to 1. (Zhu, 2010). So, now we have established a regular sequence labelling model with a function which mimics the softmax activation to generate probabilities for each element in the input. This means that we are modelling the relationship between successive labels. To implement that, we simply multiply or previous probability by P(yk+1|Xk) and rewrite the **emission scores** U(x,y) and add learnable ***transition scores*** T(x,y). This gives us

$$P(y|X) = \frac{exp(\sum_{n=1}^{l} \bigcup(x_n, y_n) + \sum_{k=1}^{l-1} T(y_k, y_{k+1}))}{\prod_{n=1}^{l} Z(x_n)} \quad (3)$$

T(x,y) is essentially a matrix where each element in it is a learnable parameter which represents the transition from the ith label to jth label. This gives us the linear-chain Conditional Random Field where (3) is the conditional probability for each element (Zhu, 2010). After a comprehensive and thorough look on sequence labelling and CRF, we have made some key decisions on using linear-chain CRF to classify each element in a sentence into predefined labels. The initial process includes the data collection and cleaning which has been thoroughly described in the Data processing report by Varun Vamsi Saripalli. This section assumes that we have our training data labelled and ready to train a linear-chain CRF model.
As a part of data pre-processing, we have **parts of speech tag generation** as follows: Parts of Speech form the building blocks of understanding the context of each element in a sentence. For example, if we do not include POS tags in our data, our trained model will not be able to capture the difference between “I like a potato” and “I am like Helen” where the former has a verb context while the latter has a preposition context. Furthermore, POS tags have been deemed to be useful in extracting relations between words and also building lemmatisers to reduce a word its root form. However, in our case, we are using POS tags to extract relationship between consecutive words. In order to do so, we are using the NLTK library provided as an open source library Stanford’s CoreNLP API. In order to tag our words, we simply call the pos tag(word) to generate a POS tag the word in our tagged dataset. Once, we do that for the entire corpus of sentences, our dataset would look as shown in Figure 8.

Figure []: Generated POS tags with other columns in the data

	Token	NE	POS
0	50,00,000	B-Notional Amount	CD
1	USD	O	NNP
2	10/15/03	B-Expiration Date	CD
3	Agreement	O	NNP
4	with	O	IN
5	Deutsche	B-Counterparty	NNP
6	Bank	I-Counterparty	NNP
7	AG	I-Counterparty	NNP
8	dated	O	VBD
9	1/21/03	O	CD

Filtering labels: As shown in the figure 8, the words have a label associated to them which was generated during the data preprocessing step. These labels serve as one of the inputs into the Conditional Random Field model that we will implement to develop classifier to label words.

Label	Count
B-Counterparty	491
B-Direction of Trade	504
B-Expiration Date	492
B-Fixed Rate	511
B-Notional Amount	488
B-Reference Entity	498
I-Counterparty	843
I-Expiration Date	97
I-Fixed Rate	1
I-Notional Amount	2
I-Reference Entity	1100
O	15094

Table 1: Label Distribution in our data

However, from table 1 we can infer that almost 15,094 labels are of the O label or no-entity label which means these words have no significance in our analysis as they do not carry information that we want to extract. This could bring in class imbalance and would inflate the accuracy of our CRF model as it would tend to classify other label as O(no-entity label). This would lead to a high theoretical accuracy but it would classify most words as O label. To mitigate this, we could either oversample the other non-O labels or under-sample the O-label. We chose to under-sample the O-label by simply dropping the words which carry the label O. So the final labels that we will classify all the words into are : B-Direction of Trade, B-Reference Entity, I-Reference Entity,B-Fixed Rate, B-Counterparty, I-Counterparty, B-Expiration Date, I-Expiration Date, B-Notional Amount, I-Notional Amount and I-Fixed Rate.

Feature Extraction: Here, we generated POS tags for every word in our corpus of sentences. This is because they convey important information about the word or group of words in a sentence like its semantic meaning and its position in the sentence. This would allow us to develop feature function which is one of the significant aspects of CRF. As we are essentially building linear-chain CRF, the feature function would look like:

$$f_i(z_{n-1}, z_n, x_{1:N}, n) \quad (4)$$

where zn-1, zn are adjacent states or words in a sentence and the whole sentence sequence is denoted by x1:N. Lets take an example specific to our use case to further understand its significance.
Let's assume that a simple feature function which produces binary values for if the current word is JPMorgan and the current label is B-Counterparty. The CRF model will utilise this feature function with it's corresponding weight λ1 . In this case, if λ1 > 0 and if the current word is JPMorgan and current label is B-Counterparty then our feature function will be active. This is interpreted by the CRF model as increase in probability of labelling a word as B-Counterparty if the word is JPMorgan. Similarly, if λ1 < 0, then the CRF model will have a lower probability of tagging a word as B-Counterparty if the word is JPMorgan. Now the λ1 can either be specified through the process of labelling or learning from the corpus or both. In our use case, we will learn λ1:n from training data, for which we developed feature dictionaries from our training data for the CRF model to train on. Since, we are not able to implement LSTMs to extract the word features and pass it to successive units ahead, there is a need for feature dictionary which would consist of word features of every element that would be extracted from each sentence. A list of feature dictionaries for each word token in a sentence can be extracted, corresponding to a list of labels for each word token in a sentence.

1. Previous Parts of Speech Tag
2. Current Parts of Speech Tag
3. Word.isdigit() [True if word is a number false otherwise]
4. Word.isUpper() [True if word is a upper false otherwise]
5. Previous Word
6. Word.isLower() [True if word is a lower false otherwise]
7. Word.isLower() [True if word is in title case false otherwise]

```
First token features:
---------------------
{'Token.lower()': 'upon', 'Token.isupper()': False, '+1:Token.lower()':
'a', 'Token[-2:]': 'on', '+1:Token.istitle()': False, 'Token[-3:]': 'po
n', 'POS[:2]': 'IN', 'POS': 'IN', 'Token.istitle()': True, '+1:POS[:2]':
'DT', 'BOS': True, '+1:Token.isupper()': False, '+1:POS': 'DT', 'bias':
1.0, 'Token.isdigit()': False}
```

Figure[]: Example of Feature Dictionary for the word ‘upon’

Let’s visualize a feature dictionary for the word upon in figure 9. Using a feature dictionary like this for each word, we synthesized a list of feature dictionaries for all the words in the data. Furthermore, we have a used standardized format for the feature dictionary as dictated in python-crfsuite documentation in order to ensure easy replication of similar results.
1. State-of-the-art training method using methods like Limited-methods BFGS
2. Linear-chain (first-order Markov) CRF
3. Performance evaluation on training
Following this, using predefined functions, X and y were denoted as a list of feature dictionaries for each word in each sentence and as a list of labels for each word in each sentence. Making use of scikit-learn library’s test train split function, the data was split into training and testing data with 80% of the data dedicated for training while 20% for testing.
Serving as an extension to the content covered in the literature review for CRF, the partition state or normalization factor used to compute the probability in the end which is denoted by Z in equation(3) can be expressed as following:

$$Z(X) = \sum_{y_1^{'}} \sum_{y_2^{'}} ... \sum_{y_n^{'}} exp(\sum_{k=1}^{l} U(x_k, y_k^{'}) + \sum_{k=1}^{l-1} T(y_k^{'}, y_{k+1}^{'}))) \quad (5)$$

However, the computation of the partition function Z is computationally intensive as it has a lot nested loops (Zhu, 2010) . There are l! computations required over the label set. This gives us a total complexity of O(l!|y| 2) . However, CRFSuite library makes it easy to tackle this by providing the use of forward or backward algorithm as simple function argument while training a CRF model. Depending on the order of iteration for a sequence, we can choose the algorithm we want to use. Given that we are employing Linear-Chain CRF, we would use forward-backward algorithm. Finally, for optimisation, standard optimisation algorithms like Stochastic Gradient De- scent or Limited-Memory BFGS could be used. We used L-BFGS as the library CRFSuite provided support for the it. So in order to express our problem in a mathematical form, let us assume that our fully labelled data is represented as (w(1), t(1), s(1))..., (w(n), t(n), s(1)), where w(i) = w(i) are the sequence of words present in our unstructured 1:N sentences containing Credit Default Swaps, t(i) = t(i) are labels for each 1:N corresponding word in the sentence, and s(i) = s(i) are the corresponding 1:N parts-of-speech tag for the corresponding word, respectively.
We already know that in CRFs, the objective of parameter learning is to maximize the conditional likelihood on the basis of training data. This can be represented as:
In order to stop over-fitting, we conduct penalization on log-likelihood with a zero-mean Gaussian Distribution over the parameters. This makes equation (6) as:

$$\sum_{j=1}^{M} logp(t^{(j)}|w^{(j)}, s^{(j)}) \quad (6)$$

As equation (7) is concave in nature, we can deduce that λ would have unique set of optimal values. With the help of L-BFGS's gradient, we learn the parameters. So training the CRF model would allow us to find the optimal values of λ for the training data. We provide numerical results in the form of F1-Score, Precision and Re- call. Precision here is defined as the fraction of relevant data points from the retrieved data points while Recall is defined as the fraction of relevant data points returned from the total number of relevant data data points. So precision and recall will serve as a good metric to understand and measure the relevance of our results ("Precision-Recall scikit-learn 0.20.3 documentation," n.d.). Therefore, F1 score is the measure of a test's accuracy and is defined as the weighted harmonic mean of the precision and recall of the test. With this, we first trained the CRF Model on the CONLL 2003 dataset and tried to validate performance by benchmarking with other state of the art implemen- tations. Benchmarking allows us to validate the conditions and settings that we have used to train our CRF model. From Table 2, we can incur that our CRF model closely represents the Conv-CRF(Collobert et al., 2011) in terms of both the settings used for traning as well as accuracy.

System	**Accuracy**
Conv-CRF (Senna + Gazetter) (Collobert et al., 2011)	89.59 %
Early CRF Models (MacCullum, Li (2005))	84.04%
Conv-CRF (Collobert et al., 2011)	81.47%
CRF with Lexicon Infused Embeddings (Passos et al., 2014)	90.90%
CRF (Our)	**81.21%**

Table 2: Benchmarking performance with other state-of-the-art CRF models

Performance on Unstructured CDS Reporting:

As mentioned in the methodology, the next step was to train our CRF model for the unstructured Credit Default Swap reporting from 2004-2017. This primarily includes sentences like those shown in figure (3). The training itself took about 20 minutes on a 2.3GHz i7 processor and returned with following results shown in table 3.

Label	**Precision**	**Recall**	**F1-Score**	**Support**
B-Notional Amount	0.98	0.94	0.96	99
B-Expiration Date	0.96	0.97	0.97	102
B-Counterparty	0.98	0.98	0.98	101
I-Counterparty	0.97	0.96	0.96	182
B-Direction of Trade	0.96	0.97	0.97	106
B-Fixed Rate	0.98	1.00	0.99	105
B-Reference Entity	0.98	0.97	0.98	104
I-Reference Entity	0.96	0.93	0.94	245
I-Expiration Date	0.94	0.89	0.92	19
I-Notional Amount	0.00	0.00	0.00	1
I-Fixed Rate	0.00	0.00	0.00	1
Micro Average	**0.97**	**0.96**	**0.96**	**1065**
Macro Average	**0.79**	**0.78**	**0.79**	**1065**
Weighted Average	**0.97**	**0.96**	**0.96**	**1065**

Table []: Precision, Recall and F1-score of the CRF model on unstructured CDS reporting with scores for each label

One interesting aspect to note is that there were certain labels like I- Notional Amount and I-Fixed Rate which had only one instance in the entire dataset. Therefore, our model rejects the labels with only one instance and reports an F1-score of 0. This allows us to establish that the CRF model that we have developed and trained on the unstructured CDS report could be used to extract key information which is often hidden in

the context of unstructured sentences. However, it could be concluded that there is no prior work conducted on extracting unstructured reporting of CDS and therefore making it difficult to benchmark our performance with other studies.

Studies Concluded	**F1 Score**
(Alvarado, Verspoor and Baldwin, 2013)	0.827
(Wang, Xu, Liu, Gui and Zhou, 2015)	0.857
Bankruptcy Prediction using CRF	0.859
Our implementation	**0.96**

Table 4: Studies published in the finance domain which used CRF to extract information and their reported F1 Scores

But we chose benchmark our performance with similar studies which implemented CRF on finance-specific datasets. we have gone over three major studies in this direction mentioned in table 4. These studies go through studies conducted on similar financial documents such as loan agreements and contracts using Conditional Random Field to extract unstructured information.

Optimizing hyperparamters:

Under this section, we experimented with different values of C1 and C2 values for the elastic net regularization. In order to achieve this we used cross-validated randomized search. To avoid a computationally intensive task, we limited the iterations to 50 and use a 3-fold cross-validation. This in turn would mean that we essentially trained a 150 models. Following the optimization, we noticed that lower values (increased regu- larization strength) for both C1 and C2 values result in the best performing model - particularly for C1. After optimizing the hyperparameters, the CRF model was be evaluated again. The results of the new model have been disclosed in table 5. We noticed that there isn't a significant improvement in the F1-score of the trained model after optimization. However, this is attributed to the size of data that we began with. We firmly believe that if the training data was much larger then there would have been a considerable affect of the hyperparamter optimization.

Studies Concluded	**C1**	**C2**
96.81%	0.001	0.001

Table 5: Best Score and Best parameters after conducting Randomized- CVSearch

Analysis:

One of the main goals of this project was to answer some questions surrounding Credit Default Swap reportings and the data processing and extraction process was supposed to allow us to have enough data to answer questions like

1. What were the trends of Credit Default Swap reporting before financial crisis, during and after the financial crisis ?
2. What are the patterns in usage of index CDS, Sovereign CDS and Single-name CDS ?
3. Do funds who have a more structured format of reporting CDS have a higher profit margin as compared to funds who used unstructured sentences to report CDS ?

In order, to answer these questions, the Credit Default Swap dataset which was one of the deliverables of this project is supposed to visualise the trends in CDS reporting in a graphical manner.

The first question deals with the reporting trends of CDS from the time period of 2004-2017 and after performing a simple groupby by column name "Reporting Year" and the trends have been reported in figure 10. We notice that the CDS reporting from the time period of 2004-2008 have an exponential rising trend. This trend could be attributed to the housing market bubble in the United States. Housing market was believed to be one of the safest and risk-free investments from a consumer standpoint. Using this trend in the market, a lot of financial institutions bought Credit Default Swap in the expectation of earning more profits through them as the thought of the financial markets collapsing was simply considered to be least probable. However, as 2008 came along and when the housing market collapsed we see significant drop in the CDS reporting and from there on it has been going down to a record low in 2017. This mirrors the real-world circumstances as with heavy regulation from the SEC and the housing market not deemed as safe as it was at

its peak, the financial institutions investing in CDS has slowly gone down and they have now started looking towards alternate credit derivatives which allows them to hedge their risk and diverse their portfolios in a better method.

The second question dives deeper into the trend of different types of Credit Default Swaps being used during 2004-2017 time period. We notice all three types of CDS were again very popular in the 2004-2008 period when the market for Credit Default Swaps was booming.

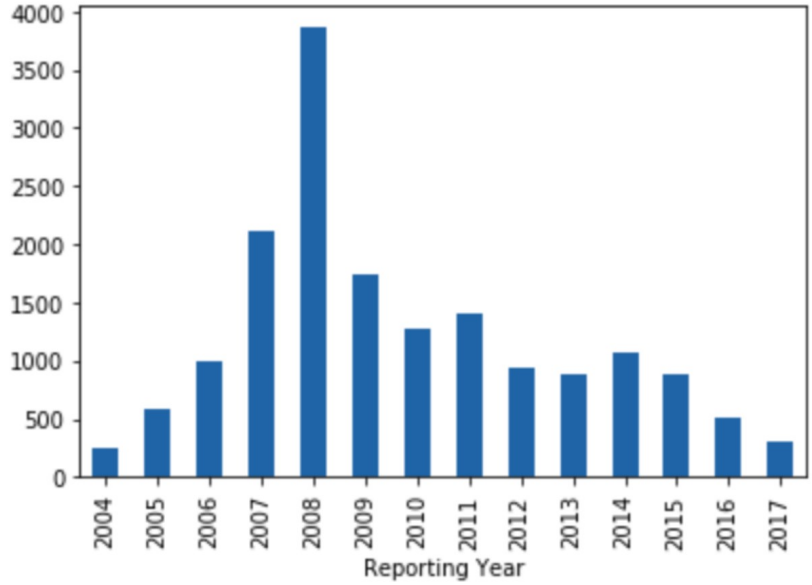


Figure 11: Trend of CDS reporting for every year from 2004-2017

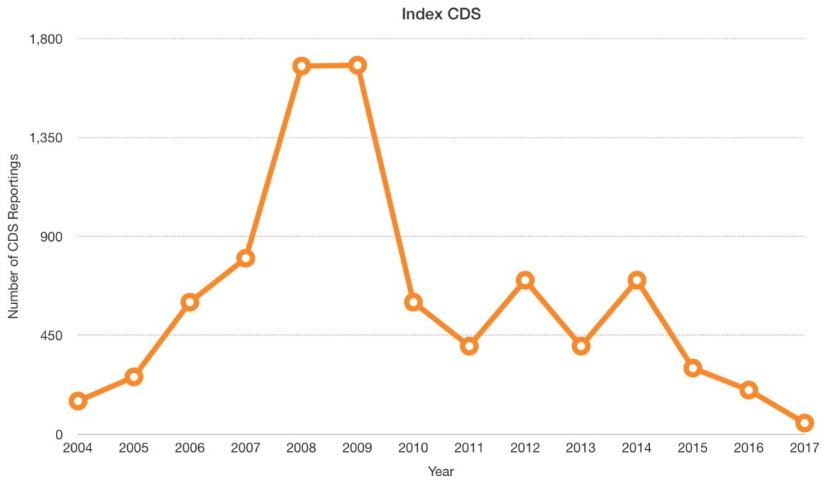


Figure 12: Trend of CDS reporting by Index CDS

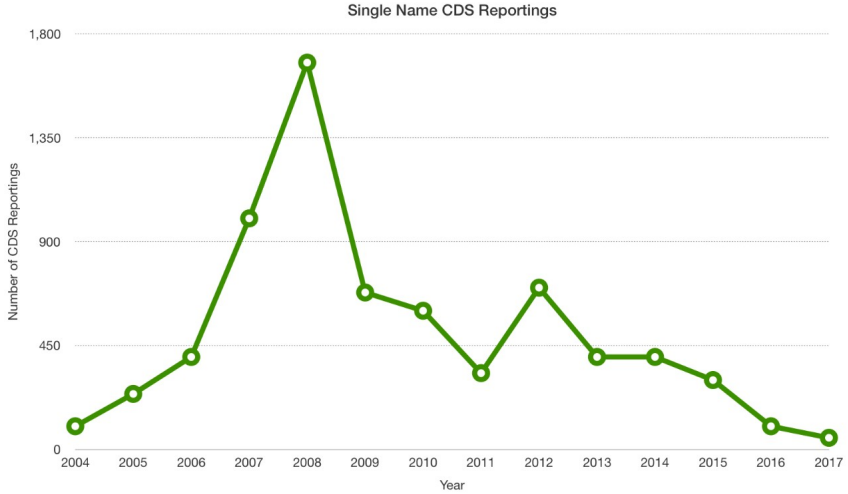


Figure 13: Trend of CDS reporting by Single Name CDS

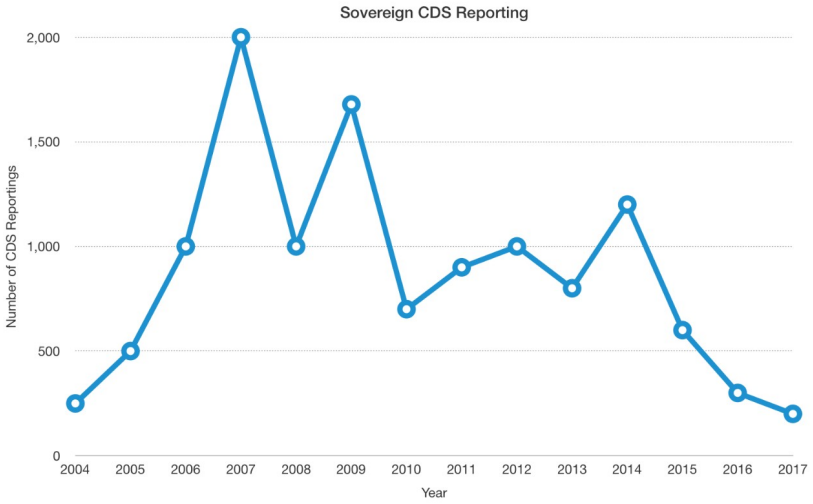


Figure []: Trend of CDS reporting by Sovereign CDS

For sovereign CDS, we noticed a similar trend in reporting with Reference Entities including countries like China and Russia were one of the most reported Credit Default Swaps. Furthermore, Single Name and Index Name CDS, both represent a similar reporting pattern throughout the years as it clearly correlates with the rise and the fall of the CDS market.

The third question involves the comparison of structured and unstructured reporting to find if funds practicing unstructured method of reporting earn higher margin on CDS or the funds with the structured method reporting. Answering this question would require significant analysis on both structured and unstructured information, however, as explained in the section earlier, we only 1,200 reporting of unstructured CDS information from the total 16,813. With this size of unstructured data, it is not possible to conclude that the funds with unstructured method reporting have a higher margin of profit as the sample space for comparison is too small. However, with more unstructured CDS reporting, this could be made possible.

We investigate how mutual funds leverage credit derivative by studying their routine fillings to the U.S. Securities and Exchange Commission. Credit derivatives are used to transfer credit risk related to an underlying entity from one party to another without transferring the actual underlying entity.

Instead of studying all credit derivatives, we focus on Credit Default Swap (CDS), one of the popular credit derivatives that were considered the culprit of the 2007-2008 financial crisis. A credit default swap is a particular type of swap designed to transfer the credit exposure of fixed income products between two or more parties. In a credit default swap, the buyer of the swap makes payments to the swaps seller up until the maturity date of a contract. In return, the seller agrees that, in the event that the debt issuer defaults or experiences another credit event, the seller will pay the buyer the security's premium as well as all interest payments that would have been paid between that time and the security's maturity date. CDS is traded over-the-counter, thus there exists little public information on its trading activities for the outside investors. However, such information is valuable. CDS is designed as a hedging tool that the buyers use to protect themselves from potential default events of the reference entity. Besides, it is also used for speculation and liquidity management especially during a crisis.

Before SEC has requested more frequent and detailed fund holdings reporting at the end of 2016, mutual funds filed the forms in discrepant formats. This made it extremely difficult to effectively extract information from the reports for carrying out further analysis. There exist some previous studies that explored how mutual funds have made use of CDS (Adam and Guttler, 2015, Jiang and Zhu, 2016), but only examined a fraction of institutions over a short period of time. In this project, we aim to extract as much CDS-related information as possible from all the filings available to date to enable more thorough downstream analysis. This information appears not only in the form of charts but also in words, thus Natural Language Processing (NLP) is the key.

We implemented bidirectional LSTM-CRF for Named Entity Recognition relationship on custom corpus using custom word embeddings using tensor-flow.

Named Entity Recognition:
Steps that we follow are-

Data pre-processing:
(a) loading necessary library functions
(b) parts of speech generation : We get Token, NE and POS values
(c) visualising tag distribution : We can visualise tag distributions like named-entity
(d) extracting sentences from the dataset: it's useful to have the sentences as a list of lists, with each sublist containing the token, POS tag, and NE label for every word token in the sentence
Extract features from the sentences (Feature engineering):
(a) current parts of speech tags
(b) previous and next parts of speech tags
(c) current words
(d) previous words
(e) next set of words
Training a Conditional Field model:
using predefined functions, X and y can be extracted as lists of feature dictionaries for each word token in each sentence, and as a list of NE labels or each word token in each sentence, respectively. scikit-learn's 'test_train_split' function can then be used to split X and y into training and test sets, split 80% training to 20% test.
Evaluating the trained CRF model:
The trained model can now be used to make predictions based on the test data, which can in turn be compared to the expected labels from the test data to produce a classification report (precision, recall and F1 scores).
Optimising the parameters:
Under this section, we will experiment with different values of C1(a parameter grid to experiment with different values of C1- initialised to values from 0.0001 to 1000.00) and C2(similar to C1 values) values for the elastic net regularisation. In order to achieve this we will use cross-validated randomised search. To avoid a computationally intensive task, we will limit the iterations to 50 and use a 3-fold cross-validation. This in turn would mean that we are essentially training a 150 models.
Following the optimisation, we can see that lower values (increased regularisation strength) for both C1 and C2 values result in the best performing model - particularly for C1
Evaluate the optimised CRF model:
We evaluate the already built model with metrics like precision, recall, f1-score, support

Dataset:

final_csv

CIK	Reporting Type	Reporting Year	Counterparty	Notional Amount	Reference Entity/Obligation	Expiration Date	Appreciation/Depreciation	Upfront Payments Paid/Received	Buy/Sell Protection	Description	Fixed Rate
315774	N-CSR	17	Barclays	1,00,00,000	Eastman Chemical Co, 7.60%, 02/01/27	12/20/21	NaN	35,394	NaN	NaN	1.00%
315774	N-CSR	17	Barclays	50,00,000	Host Hotels & Resorts, 4.75%, 03/01/23	12/20/21	NaN	50,649	NaN	NaN	1.00
315774	N-CSR	17	Barclays	1,00,00,000	Macy's Retail Holdings, 7.45%, 07/15/17	12/20/21	NaN	5,07,117	NaN	NaN	1.00
315774	N-CSR	17	BNP Paribas	1,00,00,000	Host Hotels & Resorts, 4.75%, 03/01/23	06/20/22	NaN	-24,159	NaN	NaN	1.00
315774	N-CSR	17	BNP Paribas	50,00,000	International Paper Co, 7.50%, 08/15/21	12/20/21	NaN	-49,431	NaN	NaN	1.00

Probability of default:
Predicting the financial health of a company is something that financial analysts have longed for a very long time. In specific to the CDS dataset, this sort of analysis has a lot of relevance in the practical real world where in financial crisis could be evaded if such a framework devised in this research paper could be implemented and the data so extracted could be studied.
To help serve as an example for future research and analysis, we have developed a novel method using one of a kind Credit Default Swap dataset that we have developed. In order to predict the probability of default of Credit Default Swap, there are some key factors which play a role. These include, the recovery rate R and credit spread S. The dataset that we have put together allows us to extract the credit spread S for all the credit default swaps reported from 2004-2017 and hence allows us to compute the default probability of the firms that the user wants to search for by simply typing their name. What simplifies the entire process is the RESTful API which is running in the backend computing the default probabilities for all the three possible

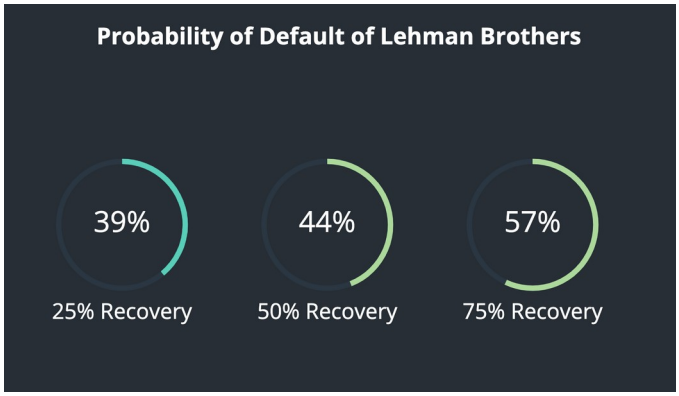
recovery rates and allowing the user to understand the likelihood of a specific financial institution defaulting on their credit default swap portfolio. Applications like this show how useful a Credit Default Swap dataset would be to trade in the CDS market and also shows the power of computation by using existing quantitative finance formula.

calculation

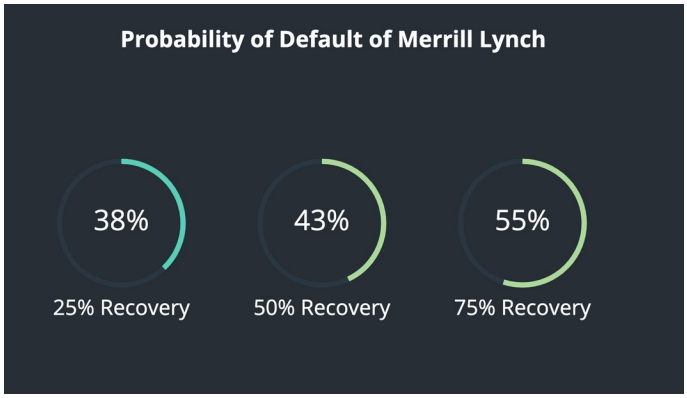
CIK	Reporting Type	Reporting Year	Counterparty	Notional Amount	Reference Entity/Obligation	Expiration Date	Appreciation/Depreciation	Upfront Payments Paid/Received	Buy/Sell Protection	Description	Fixed Rate	Spread	25% Recovery Rate	50% Recovery Rate	75% Recovery Rate
315774	N-CSR	17	Barclays	10000000	Eastman Chemical Co, 7.60%, 02/01/27	12/20/21	NaN	35394	NaN	NaN	1.00%	7.6	10.1333333333	15.2	30.4
315774	N-CSR	17	Barclays	5000000	Host Hotels & Resorts, 4.75%, 03/01/23	12/20/21	NaN	50649	NaN	NaN	1	4.75	6.3333333333	9.5	19
315774	N-CSR	17	Barclays	10000000	Macy's Retail Holdings, 7.45%, 07/15/17	12/20/21	NaN	507117	NaN	NaN	1	7.45	9.9333333333	14.9	29.8
315774	N-CSR	17	BNP Paribas	10000000	Host Hotels & Resorts, 4.75%, 03/01/23	06/20/22	NaN	-24,159	NaN	NaN	1	4.75	6.3333333333	9.5	19
315774	N-CSR	17	BNP Paribas	5000000	International Paper Co, 7.50%, 08/15/21	12/20/21	NaN	-49,431	NaN	NaN	1	7.5	10	15	30

Figure[]: There are additional columns added like Spread, 25% Recovery rate, 50% Recovery rate, 75% Recovery rate

We calculate first probability, second probability, third probability as:
firstprobability = (sum(search_results['25% Recovery Rate'])/(search_results.shape[0] - search_results[search_results.Spread=='NaN'].shape[0])) + 30
secondprobability = (sum(search_results['50% Recovery Rate'])/(search_results.shape[0] - search_results[search_results.Spread=='NaN'].shape[0])) + 30
thirdprobability = (sum(search_results['75% Recovery Rate'])/(search_results.shape[0] - search_results[search_results.Spread=='NaN'].shape[0])) + 30



Figure[]: Self-explanatory



Figure[]: Self-explanatory

Classification performance after label correction:

Model	Accuracies per Task (%)	
	Noisy	**Corrected**
guse-dense	57.97	70.63
bert-dense	48.86	70.13
bert-bilstm	56.71	68.35
bert-cnn	55.70	70.13

Classification performance:

Metrics (%)	Model Combinations			
	guse-dense	bert-dense	bert-lstm	bert-cnn
Acc	70.58	67.73	69.34	71.14
Rec	71.35	69.92	64.87	72.99
Prec	68.88	66.89	71.08	71.18
F1	71.01	70.45	68.56	71.92
AUC	72.09	73.31	74.23	75.45

Final evaluation results:

Metrics (%)	UMAP-KMeans				UMAP-GMM			
	guse-dense	bert-dense	bert-bilstm	bert-cnn	guse-dense	bert-dense	bert-bilstm	bert-cnn
Acc	**91.16**	71.65	73.11	72.27	**93.08**	81.74	82.16	82.95
Prec	**92.67**	81.88	82.01	86.88	94.76	**93.15**	91.01	93.36
Rec	**90.78**	74.63	76.00	74.76	**96.16**	82.08	86.07	85.98
F1	**90.07**	78.24	79.15	80.09	**95.44**	88.98	88.96	89.43
AUC	**96.12**	75.81	78.91	76.98	**96.88**	80.97	83.05	81.19

Another use case is testing this framework on south east asian news agency dataset (tagged dataset):
The use case that we are discussing is of Vietnamese news agency which has already tagged labels with the dataset.
Illustrations mentioned in the dataset are as follows:
One of the records in the dataset consists of Vietnamese text as follows:
Vingroup báo lãi hơn 3.600 tỷ đồng, doanh thu thuần gần 38.500 tỷ đồng. Tập đoàn Vingroup (Mã:VIC) vừa công bố báo cáo tài chính hợp nhất quý II cho thấy sự tăng trưởng mạnh cả về doanh thu và lợi nhuận. Cụ thể, trong quý II vừa qua, tổng doanh thu thuần hợp nhất đạt 38.451 tỷ đồng, tăng 65% so với cùng kỳ năm trước. Hầu hết các lĩnh vực kinh doanh đều tăng trưởng doanh thu. Với sự tăng trưởng doanh thu, lợi nhuận trước thuế trong quý II của Vingroup đạt 3.618 tỷ đồng, tăng 34% so với cùng kỳ. Tại ngày 30/6, tổng tài sản Vingroup đạt 417.881 tỷ đồng. Vốn chủ sở hữu đạt 144.442 tỷ đồng. Thông tin về từng mảng hoạt động, ở lĩnh vực công nghiệp, VinFast tiếp tục tăng trưởng với doanh số đạt gần 16.000 xe bán ra trong 6 tháng đầu năm 2021. Theo đó, VinFast Fadil đạt doanh số hơn 10.000 xe, trở thành mẫu xe bán chạy nhất thị trường. Lux A2.0 và Lux SA2.0 cũng đều có sản lượng đứng đầu phân khúc. Mẫu xe điện đầu tiên của Việt Nam VF e34 đạt 25.000 lượt đặt cọc tính đến cuối tháng 7. Tính đến ngày 18/7, VinFast chính thức vận hành 35 showroom xe máy điện kết hợp trung tâm trải nghiệm Vin3S tại 24 tỉnh, thành phố trên cả nước, nâng tổng số điểm cung cấp dịch vụ lên hơn 200 showroom và đại lý. Với thị trường quốc tế, VinFast đưa vào hoạt động các chi nhánh tại Hoa Kỳ, Canada, Pháp, Đức và Hà Lan, mở rộng mạng lưới đối tác nhằm chuẩn bị ra mắt thị trường toàn cầu hai mẫu ô tô điện thông minh VF e35 và VF e36 vào năm 2022 với công nghệ hỗ trợ lái tự động (ADAS) và hệ thống thông tin giải trí thông minh (Smart Info-tainment). Ở lĩnh vực bất động sản nhà ở, mô hình kinh doanh O2O (Online to Offline) đã thu hút 16.000 lượt xem tại sự kiện livestream mở bán dự án The Metrolines tại Vinhomes Smart City. Bên cạnh đó, ứng dụng dành cho đại lý bán hàng từ khi đi vào thử nghiệm từ tháng 5 đã ghi nhận hơn 500 giao dịch đặt cọc thành công, với gần 50 đại lý và hơn 6.000 người dùng. Nhằm tiếp tục mục tiêu chuyển đổi số, Vinhomes đã ra mắt ứng dụng dành cho cư dân với nền tảng đồng bộ nhiều chức năng linh hoạt như đặt lịch bàn giao nhà, thanh toán hóa đơn, yêu cầu dịch vụ, xem lịch xe buýt, kiểm soát ra vào căn hộ từ xa, ghi nhận hơn 8.000 lượt đăng ký sử dụng dịch vụ thành công. Trong lĩnh vực bất động sản cho thuê, trong quý II, Vincom Retail tiếp tục chào đón các thương hiệu lớn như Muji và chuẩn bị khai trương 3 trung tâm thương mại đón đầu giai đoạn phục hồi của thị trường bán lẻ sau khi dịch được kiểm soát, bao gồm: Vincom Mega Mall Smart City, Vincom Plaza Mỹ Tho, và Vincom Plaza Bạc Liêu với sự có mặt của các thương hiệu Kohnan, Mango, Nike, FILA, Levi’s, Pizza 4P’s, McDonald’s. Lĩnh vực khách sạn – giải trí chịu ảnh hưởng nặng do dịch COVID-19 bùng phát khiến nhu cầu du lịch giảm mạnh. Để nhanh chóng thích nghi, Vinpearl đã chuyển hướng sang đón các lượt khách cách ly, giảm công suất hoạt động nhằm tiết kiệm chi phí.
English translation of the above text:
Vingroup reported a profit of more than 3,600 billion VND, net revenue of nearly 38,500 billion VND. Vingroup Corporation (Code: VIC) has just released its second quarter consolidated financial report showing strong growth in both revenue and profit. Specifically, in the second quarter, total consolidated net revenue reached VND 38,451 billion, an increase of 65% over the same period last year. Most business areas have increased revenue. With revenue growth, Vingroup's pre-tax profit in the second quarter reached VND 3,618 billion, an increase of 34% over the same period. As of June 30, Vingroup's total assets reached VND 417,881 billion. Equity reached 144,442 billion VND. Information about each operational segment, in the industrial sector, VinFast continues to grow with sales reaching nearly 16,000 vehicles sold in the first 6 months of 2021. Accordingly, VinFast Fadil

achieved sales of more than 10,000 vehicles, becoming a model Best selling car on the market. Lux A2.0 and Lux SA2.0 also have leading output in the segment. Vietnam's first electric vehicle model VF e34 reached 25,000 deposits by the end of July. As of July 18, VinFast officially operated 35 electric motorbike showrooms combined with Vin3S experience centers in 24 provinces and cities. streets across the country, bringing the total number of service provision points to more than 200 showrooms and agents. For the international market, VinFast put into operation branches in the United States, Canada, France, Germany and the Netherlands, expanding its partner network to prepare for the global market launch of two VF smart electric car models. e35 and VF e36 in 2022 with automatic driving assistance technology (ADAS) and smart infotainment system (Smart Info-tainment). In the field of residential real estate, the O2O (Online to Offline) business model attracted 16,000 views at the livestream event to open the sale of The Metrolines project at Vinhomes Smart City. In addition, the application for sales agents, since being tested in May, has recorded more than 500 successful deposit transactions, with nearly 50 agents and more than 6,000 users. To continue the goal of digital transformation, Vinhomes has launched an application for residents with a synchronous platform with many flexible functions such as scheduling house handover, paying bills, requesting services, viewing bus schedules. , controlling access to apartments remotely, recorded more than 8,000 successful registrations to use the service. In the field of rental real estate, in the second quarter, Vincom Retail continued to welcome big brands such as Muji and prepare to open 3 shopping centers to anticipate the recovery phase of the retail market after the epidemic. Control, including: Vincom Mega Mall Smart City, Vincom Plaza My Tho, and Vincom Plaza Bac Lieu with the presence of brands Kohnan, Mango, Nike, FILA, Levi's, Pizza 4P's, McDonald's. The hotel and entertainment sector was severely affected by the COVID-19 outbreak, causing tourism demand to plummet. To quickly adapt, Vinpearl has shifted to welcoming quarantined guests, reducing operating capacity to save costs.

Label provided for this particular record: VIC

This particular dataset is different from other datasets as the Labels provided for them were categorical and numeric – either they consist 0 and 1 (binary), or three levels (0,1 and 2), so we did threshold based correction accordingly. But for this dataset we have labels as stock ticker values.

To explain this further:

VIC is the stock market ticker for largest retail shopping groups in Vietnam (part of Vingroup). As we can refer the text mentioned, the news is related to Vingroup (with code VIC), hence accordingly the label has been mentioned as VIC.

Another record that has been mentioned here:

Vietnamese text (another record):

HAGL Agrico (HNG) lỗ quý thứ 5 liên tiếp. Giá giá mua phân bón, vật tư nông nghiệp, bao bì cùng chi phí vận chuyển leo thang và lỗ tỷ giá khiến HAGL Agrico lỗ ròng quý I. Theo báo cáo tài chính hợp nhất quý I của CTCP Nông nghiệp Quốc tế Hoàng Anh Gia Lai (HAGL Agrico - Mã: HNG), doanh thu thuần của công ty đạt 214 tỷ đồng, giảm 18% so với cùng kỳ năm ngoái và mới đạt 12,4% mục tiêu năm. Doanh nghiệp cho biết sản lượng trái cây thu hoạch trong quý là 13.087 tấn, giảm 19% so với quý I/2021. Trong đó, sản lượng chuối là 13.013 tấn, dứa 16 tấn. Khai thác mủ cao su đạt 1.149 tấn, giảm 3% so với cùng kỳ năm ngoái. Lợi nhuận gộp chưa tới 11 tỷ, trong khi các chi phi ăn mòn hết lợi nhuận gộp đặc biệt là chi phí tài chính (hơn 135 tỷ) nên công ty ghi nhận khoản lỗ ròng 113 tỷ đồng, cùng kỳ năm ngoái, doanh nghiệp vẫn có lãi gần 7 tỷ. Trong đó, lỗ thuần từ hoạt động sản xuất kinh doanh là 44,6 tỷ còn hạch toán lỗ do đánh giá chênh lệch tỷ giá cuối kỳ là 68 tỷ. Tới 31/3, HAGL Agrico lỗ luỹ kế 3.539 tỷ đồng. Đây cũng là quý thua lỗ thứ 5 liên tiếp của HAGL Agrico. Năm nay, HNG dự kiến ghi nhận doanh thu thuần 1.731 tỷ đồng và thua lỗ trước thuế 2.713 tỷ đồng do chi phí chuyển đổi vườn cây lớn. Giải trình về việc thua lỗ trong quý I, công ty đưa ra ba nguyên nhân. Thứ nhất, giá mua phân bón, vật tư nông nghiệp tiếp tục tăng 130%, bao bì đóng góp trái cây tăng 15% so với đầu năm. Nguyên nhân thứ nhai là về vận chuyển, công ty cho biết tình trạng thiếu hụt container lạnh để xuất khẩu trái cây, chi phí container lạnh tăng cao, thời gian vận chuyển và thông quan tăng từ 12 ngày đến 35 ngày làm ứ hàng, tăng chi phí kho bãi và giảm chất lượng trái cây. Chi phí vận chuyển tiếp tục tăng cao vào đầu năm nay, cụ thể là chi phí vận chuyển đường bộ tăng 26% (từ 19 triệu đồn/container lên 24 triệu đồng/container) và chi phí vận chuyển đường biển tăng 237% (từ 785 USD/container lên 2.650 USD/container) so với cùng kỳ năm ngoái. Thứ ba, tại ngày 31/3, tỷ giá đồng LAK/VND tại Lào giảm 6,6% so với thời điểm cuối năm 2021, vì vậy căn cứ điều 69 Thông tư 200 về chênh lệch tỷ giá hối đoái và Chuẩn mực kế toán số 25 về báo cáo tài chính hợp nhất, công ty hạch toán ghi nhận lỗ chênh lệch tỷ giá 68 tỷ đồng. Trong quý II, doanh nghiệp dự kiến sản lượng trái cây thu hoạch đạt 24.621 tấn, trong đó

chuối 24.601 tấn, khai thác mủ cao su dự kiến 1.367 tấn mủ. Doanh thu thuần mục tiêu quý II là 331 tỷ đồng. Quý II, công ty dự kiến triển khai trồng mới 100 ha dứa để tiếp tục nhân giống cho năm 2023. Tổng giá trị chi đầu tư dự kiến nửa đầu năm là 405 tỷ đồng. Về tình hình tài chính, tại ngày 31/3, tổng tài sản của HAGL Agrico đạt 13.698 tỷ đồng. Tổng nợ đi vay cuối quý là 5.932 với dư nợ từ ngân hàng là 3.244 tỷ đồng còn lại là vay chủ yếu từ HAGL (Mã: HAG) hơn 2.088 tỷ, 593 tỷ từ Thagrico. Vốn chủ sở hữu cuối kỳ giảm xuống còn 5.345 tỷ đồng do khoản lỗ luỹ kế 3.539 tỷ đồng, chênh lệch tỷ giá hối đoái 3.371 tỷ.

Its English translation:

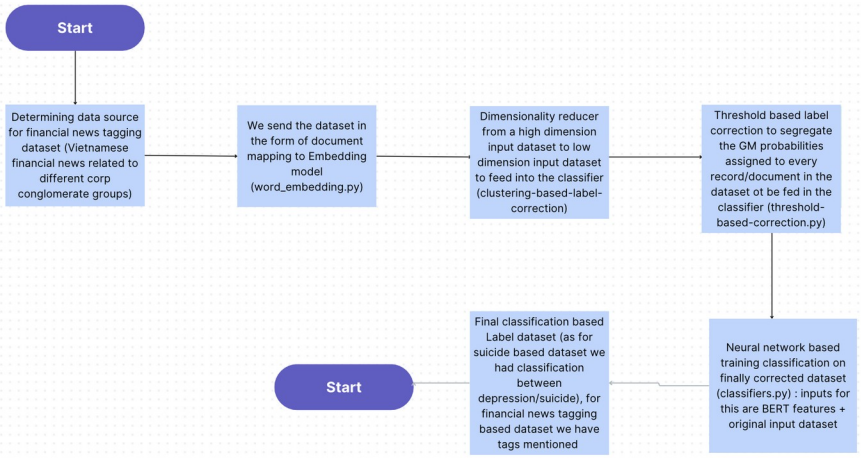
HAGL Agrico (HNG) lost the 5th consecutive quarter. The escalating purchase price of fertilizers, agricultural supplies, packaging, and transportation costs and exchange rate losses caused HAGL Agrico to have a net loss in the first quarter. According to the first quarter consolidated financial report of Hoang Anh Gia International Agriculture Joint Stock Company Lai (HAGL Agrico - Code: HNG), the company's net revenue reached 214 billion VND, down 18% over the same period last year and only reached 12.4% of the year's target. The enterprise said the fruit output harvested in the quarter was 13,087 tons, down 19% compared to the first quarter of 2021. Of which, banana output is 13,013 tons, pineapple output is 16 tons. Rubber latex exploitation reached 1,149 tons, down 3% over the same period last year. Gross profit is less than 11 billion, while expenses erode gross profit, especially financial expenses (more than 135 billion), so the company recorded a net loss of 113 billion, same period last year, the business still has a profit of nearly 7 billion. Of which, the net loss from production and business activities was 44.6 billion VND, while the loss due to assessment of exchange rate differences at the end of the period was 68 billion VND. By March 31, HAGL Agrico had accumulated losses of VND 3,539 billion. This is also the 5th consecutive quarter of loss for HAGL Agrico. This year, HNG is expected to record net revenue of 1,731 billion VND and a loss before tax of 2,713 billion VND due to large garden conversion costs. Explaining the loss in the first quarter, the company gave three reasons. First, the purchase price of fertilizers and agricultural supplies continued to increase by 130%, and fruit packaging increased by 15% compared to the beginning of the year. The second reason is transportation. The company said there was a shortage of refrigerated containers for exporting fruit, the cost of refrigerated containers increased, and shipping and customs clearance times increased from 12 days to 35 days, causing backlogs. increased storage costs and reduced fruit quality. Transportation costs continued to increase at the beginning of this year, specifically road transportation costs increased by 26% (from 19 million VND/container to 24 million VND/container) and sea transportation costs increased by 237% (from 785 USD/container to 2,650 USD/container) compared to the same period last year. Third, as of March 31, the LAK/VND exchange rate in Laos decreased by 6.6% compared to the end of 2021, so based on Article 69 of Circular 200 on exchange rate differences and Accounting Standards Accounting No. 25 of the consolidated financial statements, the accounting company recorded a loss in exchange rate differences of 68 billion VND. In the second quarter, the business expects fruit harvest to reach 24,621 tons, of which bananas are 24,601 tons, and rubber latex exploitation is expected to be 1,367 tons of latex. Target net revenue for the second quarter is 331 billion VND. In the second quarter, the company plans to plant 100 hectares of new pineapples to continue breeding for 2023. The total expected investment value in the first half of the year is 405 billion VND. Regarding the financial situation, as of March 31, HAGL Agrico's total assets reached 13,698 billion VND. Total debt at the end of the quarter was 5,932 with outstanding debt from banks of 3,244 billion VND, the remaining loan was mainly from HAGL (Code: HAG) more than 2,088 billion, 593 billion from Thagrico. Equity at the end of the period decreased to VND 5,345 billion due to accumulated losses of VND 3,539 billion and exchange rate differences of VND 3,371 billion.

Label assigned: HNG

Interpretation: This news article explains the costs financial information related to Hoang Anh Gia International Agriculture Joint Stock Company Lai. And the code assigned for HAGL Agrico is HNG (like a stock market ticker/symbol/indicator).

The two examples discussed above explains the different text dataset that is being consumed in this framework. The major challenge that we faced is with respect to setting threshold in threshold correction step for label. We experimented with different values of threshold probability values (as has been mentioned in Figure []).

Figure [] below explains the steps in the framework (as has been discussed above)



This diagram explains the step-by-step procedure explanation for the implementation of label correction based methodology deployed.

Another use case for News Article classification dataset (related to different categories):
<https://www.kaggle.com/datasets/banuprakashv/news-articles-classification-dataset-for-nlp-and-ml?resource=download>
This dataset offers comprehensive collection of news articles spanning various domains including Business, Technology, Sports, Education, Entertainment. The data is web-scraped from various news magazine “The Indian Express”. The multi-category news dataset offers a rich resource for NLP/ML practitioners, providing a diverse classification of textual data across multiple domains. Each category consists of 2000 unique news dataset – total number of rows in all files = 10000 and total number of columns = 5.
Column description is as follows:
Headlines = the headline or title of the news article
Description = A brief summary or description of the news article
Content = The full text content of the news article
URL = The URL link to the original source of the news article.
Category = The category or topic of the news article (e.g., business, education, entertainment, sports, technology).
The columns of our interest are: Category and description.

Steps of running the code files:

File_name	**Details of the file**
word_embeddings	This file contains the implementation of our BERT and Google Universal Sentence Encoder (GUSE) transformers

```
tokenized 0      [101, 2007, 1996, 8312, 1997, 1996, 9455, 5166...
1      [101, 1005, 1999, 3408, 1997, 3006, 3745, 1010...
2      [101, 2250, 2634, 2747, 2038, 12567, 6515, 294...
3      [101, 4803, 4491, 3140, 2911, 7347, 2000, 5136...
4      [101, 4237, 2013, 4170, 4935, 1010, 16798, 254...
                            ...
995    [101, 1999, 1037, 4518, 3863, 15242, 1010, 152...
996    [101, 3962, 2751, 2001, 2091, 1014, 1012, 1015...
997    [101, 5799, 2378, 2006, 9317, 2207, 2049, 1005...
998    [101, 8085, 3795, 1005, 1055, 4518, 2081, 2049...
999    [101, 2011, 9103, 1005, 1055, 2018, 1000, 2058...
Name: 0, Length: 1000, dtype: object
```

Figure []: We get tokenizer values

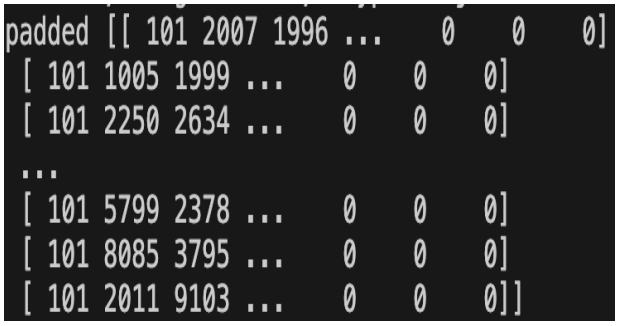


Figure []: Padding of the values

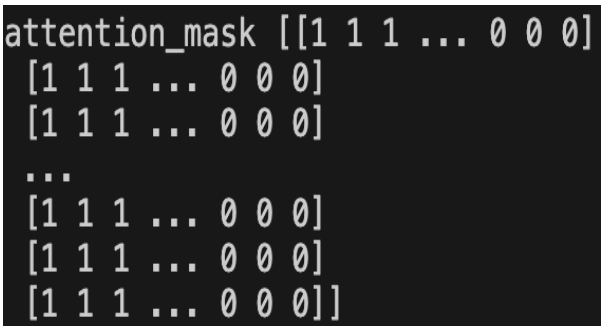


Figure []: Attention mask feature is added

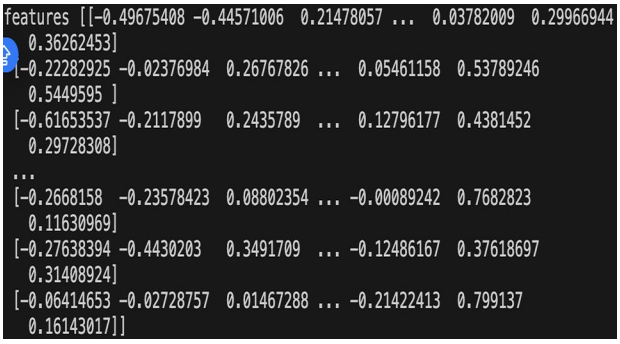


Figure []: Feature vector

clustering_based_label_correction

This file consists of loading and initialising all the clustering algorithms with the proper hyper-parameter set. We refer to dimensionality reduction algorithms like PCA, Deep Auto-encoder, UMAP and clustering algorithms like GMM, Kmeans, Subspace spectral clustering approaches

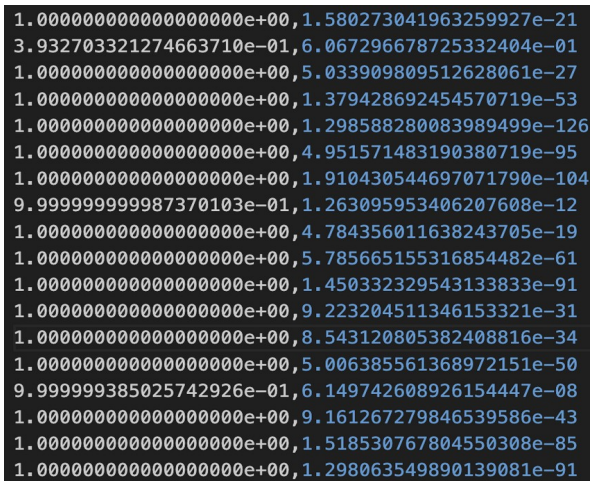


Figure []: These are the values from gaussian mixture approach.

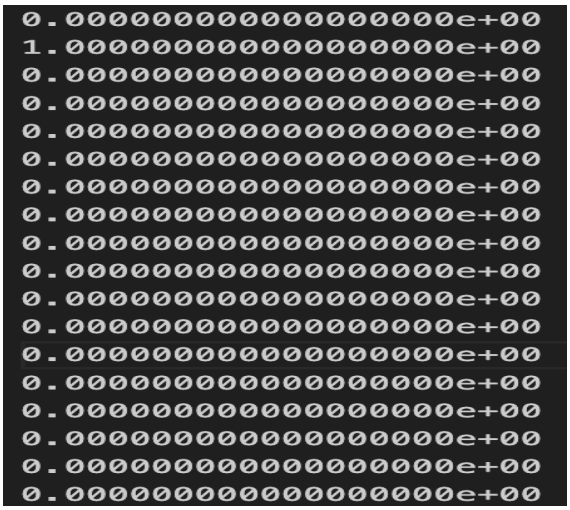


Figure []: Gaussian Mixture predictions (which would be referred in the next steps in modelling process)

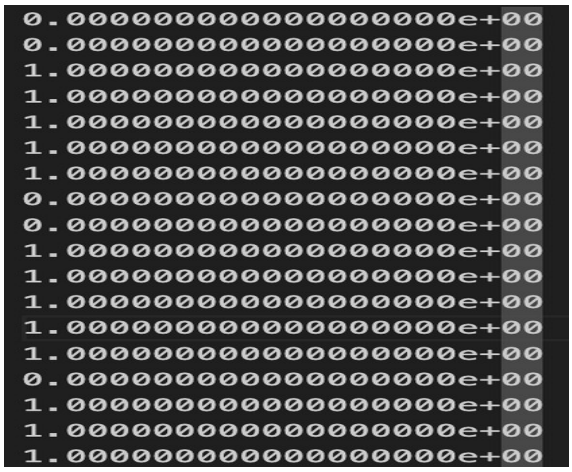


Figure []: kmeans prediction values

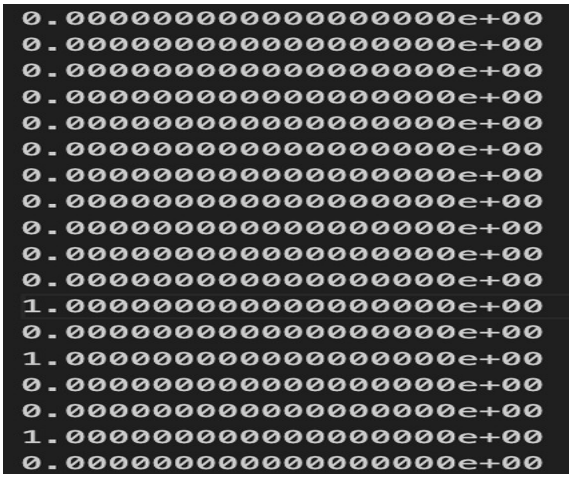


Figure []: Spectral subspace clustering algorithm

threshold_based_correction

This part corrects the ground-truth labels using clustering algorithm predictions which requires loading the ground-truth labels as well as the predicted labels exported from the clustering algorithms
The condition that we put to check this:
predicted_train_probs > tau (0.9 is the default value)
or predicted_train_probs < (1 – tau)

classifiers

This part contains implementation of our three deep neural classifiers with correct set of hyper-parameters.

The inputs that we consider here are:
bert-training-features, final_train_labels (extracted from the previous step), final_test_labels (extracted from the previous step)
We also include the training hyper-parameters with:
values for epochs, batch_size
We add more parameters for approaches like CNN, Fully Dense Network, Bi-LSTM
to get the metrics (whose graphs + results would be shown in the succeeding figures)

As such we have been gaining our understanding of setting tau parameter depending on the classification condition at the output of the classifier. For detecting suicidal/depressive tweets, we had S/D values at the output of the classifier – if the sentence got categorized as suicidal, it would lean on to a more depressive tweet (classification output would be is suicide), and if the sentence got categorized as less suicidal, it would lean on to a less depressive tweet (classification output would be is not suicide or is just plainly depressive).
In terms of AUROC curve values, when we plot TPR vs FPR, the area under curve should lean more towards TPR and less towards FPR. This further implies, predicted train probs having default probability values as 0.9 and greater would mean a true possibility of the sentence/financial news (in this case) to lean towards a certain category (news agency label in this case). But the issue that we face with this dataset is, multiple labels are present and its not a classification problem anymore. Hence, we need to adjust tau value to attain a better balance between a sentence truly belonging to a certain news label or not.

The results are:

Tau values	**Probability values**
0.1	7418 records have probability values as 1, 7279 records have probability values as 0 This implies unequal distribution of 1s and 0s
0.2	7415 records have probability values as 1, 7282 records have probability values as 0 This implies unequal distribution of 1s and 0s
0.3	7095 records have probability values as 1, 7602 records have probability values as 0 This implies unequal distribution of 1s and 0s
0.4	7184 records have probability values as 1, 7513 records have probability values as 0 This implies unequal distribution of 1s and 0s
0.5	7484 records have probability values as 1, 7213 records have probability values as 0 This implies unequal distribution of 1s and 0s
0.6	7619 records have probability values as 1, 7078 records have probability values as 0 This implies unequal distribution of 1s and 0s
0.7	We have records with values 0,1 and 2 values Ideally, we should not have 2 as label
0.8	We have records with values 0,1 and 2 values Ideally, we should not have 2 as label
0.9	We have records with values 0,1 and 2 values Ideally, we should not have 2 as label
0.95	We have records with values 0,1 and 2 values Ideally, we should not have 2 as label

Table []: We have mentioned the probability distribution values for different tau values.

This dataset is a unique case as compared to other datasets, where we have ran our simulation results – the more we have tau values, the we have labels as 0,1,2 and lesser tau values only results in 0,1 values. Ideally, none of the tau cases would work since the labelling is not-unique

In traditional classification such as multi-class problems, accuracy is the most common evaluation criteria. Additionally, there exists a set of standard evaluation metrics that includes precision, recall, F-measure, ROC area defined for single label multi-class classification problems. However, in multi-label classification, predictions for an instance is a set of labels and, therefore, the prediction can be fully correct, partially correct (with different levels of correctness) or fully incorrect. None of these existing evaluation metrics capture such notion in their original form. This makes evaluation of a multi-label classifier more challenging than evaluation of a single label classifier. Depending on the target problem, evaluation measures for multi-label data can be grouped into at least three groups: *evaluating partitions, evaluating ranking, and using label hierarchy.* The first one evaluates the quality of the classification in to classes, the second one evaluates if the classes are ranked in order relevance and the third one evaluates how effectively the learning system is able to take into account an existing hierarchical structure of the labels.

Let *T* be a multi-label dataset consisting of *n* multi-label examples (xi, Yi), 1 <= *i* <= *n*, (Xi belongs to X, Yi belongs to Y = {0,1}k), with a label set L, |L| = *k.* Let *h* be a multi-label classifier and Zi = h(Xi) = {0,1}k be the set of label memberships predicted by *h* for the example xi.

Partitions: Evaluation of some learning algorithm is the measurement of how far the learning system predictions are from the actual class labels, tested on some unseen data. To capture the notion of *partially correct,* one strategy is to evaluate the average difference between the predicted labels and the actual labels for each test example, and then average over all examples in the test set. This approach is called *example based* evaluations. Seemingly, one could define a *label based* evaluation where each label is evaluated first and then averaged over all labels. It is important to note that such *label based* method would fail to directly address the correlations among different classes.

Example-based:

Exact Match Ratio (MR): As described above, evaluation of a multi-label classification algorithm is difficult mostly because multi-label prediction has an additional notion of being *partially correct.* One trivial way around would be just to ignore *partially correct* (consider them as *incorrect*) and extend the *accuracy* used in single label case for multi-label prediction. This is called *Exact Match Ratio*.

ExactMatchRatio MR = 1/n (sigma i= 1 to n I(Yi = Zi))

where, *I* is the indicator function. Clearly, a disadvantage of this measure is that it doesn't distinguish between *complete incorrect* and *partially correct* which might be considered as *harsh.*

In order to account for partial correctness, a paper was proposed following set of definitions for accuracy, precision, recall, F1-measure.

Accuracy (A): *Accuracy* for each measure is defined as the proportion of the predicted *correct* labels to the total number (predicted and actual) of labels for that instance. Overall *accuracy* is the average across all instances.

$$Accuracy, A = \frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i \cap Z_i|}{|Y_i \cup Z_i|}$$

Precision (P): *Precision* is the proportion of predicted correct labels to the total number of actual labels, averaged over all instances.

$$Precision, P = \frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i \cap Z_i|}{|Z_i|}$$

Recall (R): *Recall* is the proportion of predicted correct labels to the total number of predicted labels, averaged over all instances.

$$Recall, R = \frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i \cap Z_i|}{|Y_i|}$$

F1-measure (F): Definition for precision and recall naturally leads to the following definition for F1-measure (harmonic mean of precision and recall).

$$F_1 = \frac{1}{n}\sum_{i=1}^{n}\frac{2|Y_i \cap Z_i|}{|Y_i| + |Z_i|}$$

As in single-label multi-class classification, the higher the value of accuracy, precision, recall and F1-score, the better the performance of the learning algorithm.

Hamming Loss (HL): *Hamming Loss* reports how many times on average, the relevance of an example to a class label is incorrectly predicted. Therefore, *hamming loss* takes into account the prediction error (an incorrect label is predicted) and the missing error (a relevant label not predicted), normalised over total number of classes and total number of examples.

$$HammingLoss, HL = \frac{1}{kn}\sum_{i=1}^{n}\sum_{l=1}^{k}[I(l \in Z_i \wedge l \notin Y_i) + I(l \notin Z_i \wedge l \in Y_i)]$$

Where *I* is the indicator function. Ideally, we would expect *hamming loss, HL = 0*, which would imply no error, practically the smaller the value of *hamming loss*, the better the performance of the learning algorithm.

If |Yi| = 1 and |Zi| = 1, then we have a single label multi-class classification problem. It is easy to note that, in that case, *hamming loss* is 2/k times of the classification error.

Another use case that we consider is news articles classification dataset for NLP and ML:

This dataset offers a comprehensive collection of news articles spanning various domains including **Business**, **Technology**, **Sports**, **Education** and **Entertainment.** The data is web scraped from the famous news magazine "*The Indian Express*". This dataset contains a comprehensive collection of news articles focusing extensively on events, developments, and topics related to India.

This dataset contains a comprehensive collection of news articles focusing extensively on events, developments, and topics related to India. The use cases for this dataset spans in the multi-category news dataset which offers a rich resource for NLP and ML practitioners, providing a diverse collection of textual data across multiple domains – this dataset can be valuable for NLP and ML tasks which includes,

- text classification
- topic clustering
- topic prediction
- named entity recognition (NER)

Each category consists of 2000 unique news content.

Total number of rows in all files = 10000

Total number of columns = 5

Column Description:

Column	Description
Headlines	The headline or title of the news article
Description	A brief summary or description of the news article
Content	The full text content of the news article

URL	The URL link to the original source of the news article
Category	The category or topic of the news article (e.g.business, education, entertainment, sports, technology)

This dataset is similar to the use-case that we had discussed above – multi-class categorical labels/output.

Dataset illustrations:

headlines: Nirmala Sitharaman to equal Morarji Desai's record with her sixth straight budget

description: With the presentation of the interim budget on February 1, Nirmala Sitharaman will surpass the records of her predecessors like Manmohan Singh, Arun Jaitley, P Chidambaram, and Yashwant Sinha, who had presented five budgets in a row.

Content: Sitharaman, the first full-time woman finance minister of the country, has presented five full budgets since July 2019 and will present an interim or vote-on-account budget next week. With the presentation of the interim budget on February 1, Sitharaman will surpass the records of her predecessors like Manmohan Singh, Arun Jaitley, P Chidambaram, and Yashwant Sinha, who had presented five budgets in a row.

Desai, as finance minister, had presented five annual budgets and one interim budget between 1959-1964.

The interim budget 2024-25 to be presented by Sitharaman on February 1, will be a vote-on-account that will give the government authority to spend certain sums of money till a new government comes to office after the April-May general elections.

As the Parliamentary elections are due, Sitharaman's interim budget may not contain any major policy changes. Speaking at an industry event last month, Sitharaman had ruled out any "spectacular announcement" in the interim budget, saying it would just be a vote-on-account before the general elections.

A vote-on-account, once approved by Parliament, will authorise the government to withdraw money from the Consolidated Fund of India on a pro-rata basis to meet expenditure for the April-July period.

The new government, which is likely to be formed around June, will come up with a final budget for 2024-25 sometime in July. Usually, interim budgets do not contain major policy announcements, but nothing stops the government from taking steps which are necessary to deal with the urgent issues facing the economy.

After the Modi government came to power in 2014, Arun Jaitley took charge of the finance ministry and presented five budgets in a row from 2014-15 to 2018-19. It was in 2017, that Jaitley departed from the colonial-era tradition of presenting budget on the last working day of February to 1st of the month.

Piyush Goyal, who was holding the additional charge of the ministry due to ill health of Jaitley, presented the interim budget for 2019-20 on February 1, 2019. Goyal had hiked standard deduction for salaried taxpayers by Rs 10,000 to Rs 50,000. Also, the tax rebate for taxpayers whose annual taxable income did not exceed Rs 5 lakh was increased from Rs 2,500 to Rs 12,500.

After the 2019 general elections, in the Modi 2.0 Government, Sitharaman was given the charge of the finance portfolio. She became the second woman to have presented a budget after Indira Gandhi, who had presented the budget for the financial year 1970-71.

That year, Sitharaman did away with the traditional budget briefcase and instead went for a 'bahi-khata' with the National Emblem to carry the speech and other documents.

Under Sitharaman, India has weathered the Covid pandemic with an array of policy measures announced for the poor and continued its tag of the fastest growing major economy and a 'bright spot' in the world economy.

India is racing to become a USD 5 trillion economy by 2027-28 and USD 30 trillion by 2047. Former Prime Minister Morarji Desai who holds the tag of presenting 10 budgets — the maximum by any finance minister — had presented six of them, including one interim, in a row.

The first budget of Independent India was presented by the first finance minister R K Shanmukham Chetty.

Sitharaman, who will be presenting her sixth budget in a row, is expected to come up with some measures, especially to boost the rural sector as the agriculture sector growth in 2023-24 is estimated to decelerate to 1.8 per cent, from 4 per cent, in the preceding year.
Rakesh Nangia, Chairman, Nangia Andersen India said given the proximity to the elections, the budget is expected to focus on immediate fiscal needs rather than introducing broad long-term economic reforms.

In the last interim budget for FY 2019-20, while the overall tax structure remained unchanged, there were certain specific tax rebates and standard deduction concessions. “While major announcements may be deferred until after the 2024 Lok Sabha elections, the budget is expected to address ongoing concerns and lay the foundation for future economic growth. This strategic approach aims to balance immediate fiscal responsibilities with long-term economic objectives in a pre-election context,” Nangia said.
Category: business
As can be seen from above, the dataset/record consists of headlines, category, description, content: we are considering content and Category for our analysis using the framework.

Results:

Classification performance after label correction

Model	Accuracies per Task (%)	
	Noisy	**Corrected**
guse-dense	60.79	71.82
bert-dense	50.83	72.89
bert-bilstm	58.54	70.64
bert-cnn	57.32	72.13

Classification performance

Metrics (%)	Model Combinations			
	guse-dense	bert-dense	bert-lstm	bert-cnn
Acc	70.24	68.05	70.44	71.41
Rec	74.37	70.29	68.66	72.89
Prec	69.48	69.74	72.82	71.08
F1	71.16	70.52	68.70	71.29
AUC	74.67	74.34	76.18	75.53

Final Evaluation

Metrics (%)	UMAP-KMeans	UMAP-GMM						
	guse-dense	bert-dense	bert-bilstm	bert-cnn	guse-dense	bert-dense	bert-bilstm	bert-cnn
Acc	**91.61**	72.65	74.15	75.72	**91.80**	82.47	83.61	83.95
Prec	**94.61**	82.81	83.20	86.48	93.67	**94.51**	92.01	92.83
Rec	**95.85**	76.65	80.00	76.54	**95.61**	84.80	86.90	85.50
F1	**95.22**	78.25	80.91	80.46	**94.34**	87.99	88.91	89.54
AUC	**99.18**	75.37	79.39	79.96	**95.81**	80.79	84.80	81.19

4 Conclusions and Ethical Discussion

In this work, we introduce SDCNL, a novel deep neural network classification technique that uses unsupervised noisy label correction to distinguish between suicidal ideation and depressive sentiment. Deep neural networks make it possible to classify closely related classes on a task that has been shown to be challenging effectively. We present a novel approach to label correction via unsupervised clustering that can handle large-scale, web-scraped datasets by efficiently removing high volumes of noise from both benchmark and domain-specific datasets. Our comprehensive testing and ablative outcomes demonstrate the efficacy of our suggested model and its potential for practical diagnostic use.

Rather than serving as a screening tool only on social media platforms, our system is intended to be used in an applied setting where it can offer professionals an additional tool for diagnosing specific patients. Professional therapists may use SDCNL as a "second opinion," friends and family may use it as a screening tool before treating loved ones, and social media platforms may use it to spot users who pose a risk.

The findings of this paper are only appropriate for research purposes; it is not a clinical study. False negative and false positive predictions would be the primary ethical concern if our algorithm were to be used as a diagnostic tool. AI systems by themselves are insufficient to provide adequate screening, particularly when it comes to suicide, which is a situation where life or death is involved. Future researchers working with our paper or on our topic should be mindful of these ethical issues and refrain from taking significant action without appropriate clinical supervision. The purpose of this paper is only to illustrate the possible effectiveness of a suicide prevention strategy.

Throughout this study, we collected our data while protecting user privacy and maintaining ethical practices. We de-identified our dataset, which we made publicly available, by removing personal information such as usernames. Moreover, Reddit is a public and anonymous forum, meaning our data source was anonymised and in the public domain to begin with.

References

[1] Reddit C-SSRS Suicide Dataset. Zenodo (May 2019) 6[2] Bering, J.: Suicidal: Why we kill ourselves. U. of Chicago Press (2018) 2 [3] Bouveyron, C., Girard, S.: Robust supervised classification with mixture models: Learning from data with uncertain labels. Pattern Recognition 42(11), 2649–2658 (Nov 2009) 3[4] Cer, D., Yang, Y., et al.: Universal sentence encoder for english. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations. pp. 169–174 (2018) 3, 4

[5] De Choudhury, M., De, S.: Mental health discourse on reddit: Self-disclosure, social support, and anonymity. In: Proceedings of the International AAAI Conference on Web and Social Media. vol. 8 (2014) 2

[6] Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proc. of the 2019 North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 4171–4186 (Jun 2019) 3, 4

[7] Fr´enay, B., Verleysen, M.: Classification in the presence of label noise: A survey. Neural Networks and Learning Systems, IEEE Transactions on 25, 845–869 (05 2014) 2, 7

[8] Hendrycks, D., Mazeika, M., Wilson, D., Gimpel, K.: Using trusted data to train deep networks on labels corrupted by severe noise. In: Advances in Neural Information Processing Systems. vol. 31, pp. 10456–10465 (2018) 2

[9] Hendrycks, D., Mazeika, M., Wilson, D., Gimpel, K.: Using trusted data to train deep networks on labels corrupted by severe noise. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems. pp. 10477–10486 (2018) 3

[10] Ji, S., Pan, S., Li, X., Cambria, E., Long, G., Huang, Z.: Suicidal ideation detection: A review of machine learning methods and applications. IEEE Transactions on Computational Social Systems (2020) 1, 2

[11] Jiang, Z., Silovsky, J., Siu, M.H., Hartmann, W., Gish, H., Adali, S.: Learning from noisy labels with noise modeling network. arXiv preprint arXiv:2005.00596 (2020) 3

[12] Jindal, I., Pressel, D., Lester, B., Nokleby, M.: An effective label noise model for dnn text classification (2019) 2, 3

[13] Leonard, C.: Depression and suicidality. Journal of Consulting and Clinical Psychology 42(1), 98 (1974) 2

[14] Maas, A.L., Daly, R.E., Pham, P.T., Huang, D., Ng, A.Y., Potts, C.: Learn- ing word vectors for sentiment analysis. In: Proc. of the Association for Computational Linguistics. pp. 142–150 (2011) 6

[15] McInnes, L., Healy, J., Melville, J.: Umap: Uniform manifold approximation and projection for dimension reduction. arXiv:1802.03426 (2018) 5

[16] Parsons, L., Haque, E., Liu, H.: Subspace clustering for high dimensional data: A review. SIGKDD Explor. Newsl. 6(1), 90–105 (Jun 2004) 5
[17] Reimers, N., Gurevych, I.: Sentence-bert: Sentence embeddings using siamese bert-networks (2019) 3, 4
[18] Schrading, N., Alm, C.O., Ptucha, R., Homan, C.: An analysis of domes- tic abuse discourse on reddit. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. pp. 2577–2583 (2015) 6
[19] Shen, J., Rudzicz, F.: Detecting anxiety through reddit. pp. 58–65 (2017) 2
[20] Song, H., Kim, M., Park, D., Lee, J.G.: Learning from noisy labels with deep neural networks: A survey. arXiv preprint arXiv:2007.08199 (2020) 2, 7
[21] Steinbach, M., Ert¨oz, L., Kumar, V.: The challenges of clustering high dimensional data. In: New directions in statistical physics, pp. 273–309. Springer (2004) 3, 5
[22] Zheng, G., Awadallah, A.H., Dumais, S.: Meta label correction for learning with weak supervision. arXiv preprint arXiv:1911.03809 (2019) 2
https://github.com/sudhamstarun/Understanding-Financial-Reports-using-Natural-Language-Processing/tree/master
[23]A.Guettler ,T.Adam. Pitfalls and Perils of Financial Innovation: The Use of CDS by Corporate Bond Funds. Jan 10, 2015.
[24]Wang, B. (2018). Forecasting Bankruptcy Prediction using Conditional Random Fields (Unpublished doctoral dissertation). Ghent University.
[25] Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies* 5, 1 (2012), 1–167.
[26] Colm Kearney and Sha Liu. 2014. Textual sentiment in finance: A survey of methods and models. *International Review of Financial Analysis* 33 (2014), 171–185.
[27] Rui Mao, Kelvin Du, Yu Ma, Luyao Zhu, and Erik Cambria. 2023. Discovering the cognition behind language: Financial metaphor analysis with MetaPro. In *IEEE ICDM*. IEEE, 1211–1216.

A Baseline Model Evaluations (just for the first use-case that we have discussed)

Appendix A displays the full experimentation of baseline results for 6 transformers and 7 classification algorithms (Table A). The results are evaluated with 5 metrics: Accuracy (Acc), Precision (Pre), Recall (Rec), F1 Score (F1), and Area Under Curve (AUC). With these results, we selected the four strongest performing combinations of models to perform the remainder of the experimentation.

Embedding model	Metrics	CNN	Dense	BiLSTM	GRU	MNB	SVM	LogReg
	Acc	72.14	70.50	71.50	71.50	57.78	68.07	68.60
	Rec	73.99	71.92	67.78	68.91	53.37	73.58	70.98
BERT	Prec	72.18	70.77	74.28	73.86	59.54	66.98	68.50
	F1	72.92	71.25	70.70	71.05	56.28	70.12	69.72
	AUC	76.35	75.43	77.11	76.66	54.21	55.43	54.72
	Acc	68.65	68.87	69.55	70.77	59.37	68.34	63.85
	Rec	73.37	74.61	67.98	67.36	46.63	73.06	69.95
SentenceBERT	Prec	67.88	67.94	71.22	73.35	63.83	67.46	63.08
	F1	70.40	70.82	69.41	70.01	53.89	70.15	66.34
	AUC	73.52	73.70	74.00	74.99	56.13	53.12	51.33
	Acc	72.66	72.24	72.82	73.19	69.39	71.50	71.50
	Rec	78.96	76.37	78.03	77.10	67.36	75.65	74.61
GUSE	Prec	70.79	71.38	71.36	72.31	71.04	70.53	70.94
	F1	74.62	73.61	74.49	74.52	69.15	73.00	72.73
	AUC	77.82	77.76	77.41	76.33	47.68	49.47	50.75
	Acc	72.66	72.24	72.82	73.19	69.39	71.50	71.50
	Rec	78.96	76.37	78.03	77.10	67.36	75.65	74.61
TFIDF	Prec	70.79	71.38	71.36	72.31	71.04	70.53	70.94
	F1	74.62	73.61	74.49	74.52	69.15	73.00	72.73
	AUC	77.82	77.76	77.41	76.33	47.68	49.47	50.75
	Acc	69.18	68.92	68.13	67.86	66.75	65.43	63.32
	Rec	82.07	73.47	75.23	74.82	72.54	69.43	66.84
CountVec	Prec	65.91	68.32	66.62	66.41	65.73	65.05	63.24
	F1	73.06	70.61	70.59	70.31	68.97	67.17	64.99
	AUC	74.44	73.34	72.66	72.30	51.47	51.08	47.35
	Acc	67.86	67.44	65.49	65.17	65.96	66.23	66.75

	Rec	71.50	69.02	66.74	68.19	69.43	69.95	72.54
HashVec	Prec	67.58	67.79	66.34	65.30	65.69	65.85	65.73
	F1	69.27	68.31	66.17	66.47	67.51	67.84	68.96
	AUC	71.47	71.63	68.98	69.58	52.94	54.06	52.85

Table 4. Performance of all 42 combinations. 7 classifiers and 6 embedding models are used, with 4 deep classifiers and 3 deep embedding models

<how do we include results for finance based data – equities/currency/etfs/funds/indices/moneymarkets – how are these files related and what is the dependent variable amongst these – how are different classification methods going to be helpful in this case just the way we have charted suicide based data?

B Comparison to Conventional Task

Appendix B presents an additional table showing the performance of our 4 best models on both the conventional task of suicidal vs clinically healthy classification against our task of suicidal vs depression classification. The results demonstrate the difficulty of our task and justify our clinical motivations.

Metrics (%)	Proposed				**Standard**			
	guse-dense	bert-dense	bert-bilstm	bert-cnn	guse-dense	bert-dense	bert-bilstm	bert-cnn
Acc	72.24	70.50	71.50	72.14	92.28	57.46	92.78	93.11
Prec	76.37	71.92	67.77	73.99	93.56	80.70	92.16	92.26
Rec	71.38	70.77	74.28	72.18	92.46	58.28	94.54	95.07
F1	73.61	71.25	70.70	72.92	93.00	67.39	93.35	93.63
AUC	77.76	75.43	77.11	76.35	96.65	54.91	97.24	96.49

Table 5. Comparison of classification metrics between conventionally researched task of suicide vs clinically healthy against our proposed task of suicide vs depression. The better performing category is bolded. The standard task performs far better on the same models, highlighting how our proposed task is more difficult to categorize.

C Vectorisers and Clustering

How the number of extracted features from the vectorizers was selected is shown in Appendix C (Figure 6). Following the application of vectorizers with varying numbers of embeddings, the MNB classifier's AUC scores are displayed. The performance converges after about 400 features for each model, so for consistency, we decided to extract 768 features. A GMM's clustering using BERT embeddings and PCA reduction is shown in Figure 7.

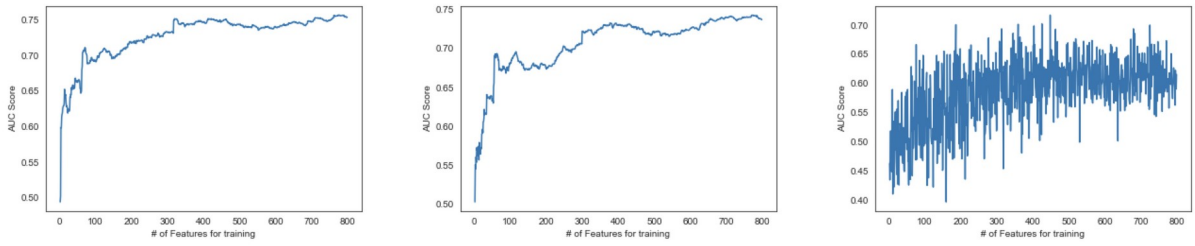


Fig 6. Area Under Curve (AUC) values at different number of extracted word embeddings/features from the three vectorizers (TFIDF, Cvec, Hvec) inputted into bayesian classifier. AUC plateaus at around 400 features, so we used 768 features to be consistent with other word embedding models.

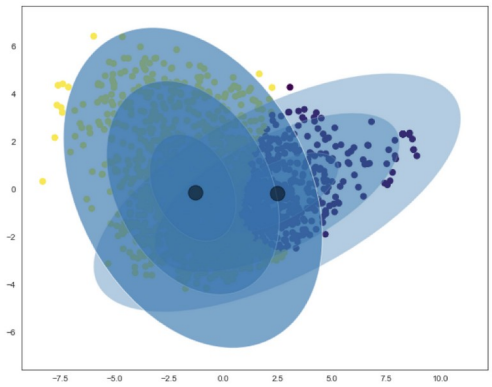


Fig 7. GMM clustering using BERT embeddings and PCA reduction to 2 dimensions show the difficulty of the clustering task, as there is little variety in the clusters and they heavily overlap. We use co-variance type “full”.