Information Content Analysis and Text Classification of ICO White Papers

Agamdeep Chopra, Ilesh Sharda, Arnold Yanga

BIA 660: Web Mining

Dr. Emily Liu

GSA: Jujun Huang

School of Business, Stevens Institute of Technology

Introduction

The scope of this project is to perform an empirical analysis on the utility of white paper content as metric for measuring the characteristics of an Initial Coin Offering (ICO). Initial Coin Offerings are a relatively new fundraising phenomenon used to launch new companies or fund a development project on a global scale. The ICO market provides companies, whom may not have the necessary network, audience, or resources, the opportunity to effectively raise funds to support the development of their product and overall growth.

A typical ICO process involves the creation and sale of a class of cryptographically protected digital records implemented on a blockchain, or a contract to deliver such tokens in the future. The tokens can later be used for a number of different purposes, depending on the setup of the ICO. Often their purposes are to redeem a product or service in the future, or to serve as a medium of exchange among users on a platform to be launched later. Tokens can be "utility" coins that serve a particular use, "security" coins that entitle holders to a certain "cash flow", or even provide voting rights within the company.

As thousands of new ICOs are created each year and fraudulent coins are ever present due to lack of regulation. It is important to explore and build upon current models in order to protect users and maintain the integrity of crowdfunding on the blockchain. Quality control within the ICO market is necessary to protect consumers and to revolutionize the traditional ways in which startups can promote growth and innovation in today's ever-evolving market. The remainder of this mid-term report is structured as follows. In Section 1, we highlight the literature that drives the motivation and methods of our work. In Section 2, we cover the data and methodology of the project. We highlight the dataset used, cleaning/preprocessing methods, information content analysis, and deep learning model architecture. In Section 3 we cover our results, and close with the future outlook of our work.

I. Background

Since the ICO market is only several years into its existence, research into risk assessment of ICOs is in its infancy. Several research groups worldwide have attempted to develop a diverse set of metrics to measure the effectiveness of ICO's based on code availability, whitepaper content, founder backgrounds, etc. In addition, websites such as ICObench and coincheckup provide users with Expert Reviews on ICO's however the sheer volume of new coins introduced yearly far outweighs the capability of human teams to properly assess the potential a particular coin has to successfully develop its product and fulfill the promises of return it promotes to their investors. Analysis on the effectiveness of online analysts and the prospect of a market-based certification process is the essence driven by the research conducted by Lee, Li, Shin 2021.

The detection of fraudulent ICO's based on key features requires analysis of a wide spread set of variables that can all be obtained on the internet but are seemingly impossible to process without the utilization of Natural Language Processing and Machine/Deep Learning methods. Bian et. al. 2018 proposed a reliable learning-based cryptocurrency rating system called ICORating. They utilized a supervised-learning model to correlate the lifespan and price change of cryptocurrencies with white paper content, code availability, websites ,etc. The team utilized a Latent Dirichlet Allocation (LDA) model on collected whitepapers to cluster whitepapers based on word co-occurrence into latent topics and assessed the success rate based on each topic. They highlighted a particular workflow that is effective in mapping whitepaper content and transforming it into a vector representation compatible with neural networks.

Additionally, the research conducted in Adhami et al 2018 elaborates on the effectiveness of particular features in measuring the probability of success of an ICO. In particular, they emphasize the importance of source code availability, token presale, adopted blockchain, etc. They used a Logit model where the dependent variable was the success of the ICO. They propose that the "whitepaper" of an ICO may not be the most revealing factor in determining whether a coin offering is fraudulent or doomed for failure. Their work aimed to characterize the ICO market and build a transparent view of an otherwise opaque and highly unregulated market.

In contrast to Adhami, Floysiak and Schandlbauer (2019) argue that white papers are the most crucial source of informa-

tion that an investor can obtain with regard to ICO investments. While fraud ICO issuers can mask their scheme by producing glamorous and convincing white papers or by running effective social media campaigns, Floysiak and Schandlbauer postulate that the distribution of informative textual content vs. standard textual content in white papers can be a strong indicator for potential fraud. Their work demonstrates that informative content reduces information asymmetry between the investor and the issuer, which in turn leads to a more transparent experience. By training a model to generate standard content and informative content coefficients for ICOs based on white paper content, they were able to observe the importance of white papers in the crypto market compare their metric to measurements such as Expert ratings which can be potentially biased.

The key takeaway is that there is no uniform methodology in place to assess the trustworthiness or likelihood of success for an ICO project. While no uniform method exists, the general consensus is that as long as information asymmetry is prevalent in the ICO market, investors will be susceptible to scams and fraud. With little to no governmental regulation, further study is necessary to retain and maintain the integrity of transactions completed through blockchain technology.

II. Data and Methodology

A. Data

Our dataset is separated into two key categories: ICO white paper text data, and ICO feature data. We use the Token Offering Research Database (TORD) developed by P.P. Momtaz. TORD is the most comprehensive database for information on ICOs, IEOs, and STOs available online and is continuously updated. Using the ICO dataset from TORD, we filtered all rows with missing white paper links and kept only the name and links columns. From there we developed a script that would send a request to the url and extract the corresponding PDF file. From there, we read each file and store the text data into a csv file. Initially we encountered runtime issues due to the fact that the initial dataset contained over 6,000 samples. With our initial script, it took over 24 hours to compile the dataset due to the presence of broken links and complex PDF files. To alleviate this issue, we implemented an asynchronous algorithm to read, process and store the data. We implement asynchronous programming using the async IO (asyncio) package in python. Asyncio utilizes "concurrent multitasking" to combat wait times for tasks through coroutines. In short, coroutines have the ability to suspend its execution before returning and indirectly pass control to other coroutines for a given time. We utilize this feature to mitigate the effects of broken links and complex PDF files by continuously switching between tasks. This is to say that multiple calls of a coroutine need not wait for each other to complete in succession. This methodology can reduce the processing time of files in half. By implementing asyncio into our code, the time it takes to import and process the text data is reduced from a scale of days to hours. After successfully mining the set of white papers, we filtered out the rows that failed to import and exported the results into a .csv file. This left us with a corpus of 929 white paper documents.

Cleaning and Initial Text Analysis

After successfully collecting all available white paper text data from TORD, we implemented our text cleaning and analysis pipeline. First, we defined functions to load and process the data. As we process the text data, we make sure to store the data as a list. From there, we defined a function to filter out stop words, non-alphanumeric strings, newlines, etc., and tokenize each remaining word for each document in our corpus. Next we created a function to define the token count of each word, allowing us to establish a document term matrix from our corpus. Using the defined functions, we then defined a function to calculate the smoothed term frequency-inverse document frequency matrix (smoothed TF-IDF). To achieve this calculation we first obtained the term frequency matrix by taking the values of our document term matrix and divide it by the length of each respective document. This operation is expressed mathematically by the following equation:

$$term\ frequency:\ tf(w,d) = \frac{freq(w,d)}{|d|} \tag{1}$$

From there, we calculate the smoothed inverse document frequency matrix as expressed below:

$$smoothed\ IDF:\ idf(w) = \ln\left(\frac{|D|+1}{df(w,D)+1}\right)+1 \tag{2}$$

where $|D|$ represents the total number of white papers in the corpus and df(w,D) denotes the number of white papers with a term "w" in them. We then calculate the smoothed TF-IDF by taking the product of the term frequency and the smoothed idf and normalize it by dividing it by its Euclidean norm.

$$tfidf(w,d) = \frac{s(w,d)}{\|s(w,d)\|_2} \tag{3}$$

where

$$s(w,d) = tf(w,d) \times idf(w) \tag{4}$$

The TF-IDF matrix allows us to assign a weight to each word in each white paper. In this way, we have created a feature space encompassing our entire corpus of white papers and are ready to build a model to extract information and keywords from the white papers as well as to assess the similarity between specific documents.

B. Information Content Analysis

The methodology implemented for our information content analysis is an adaptation of the text analysis first introduced by Hanley and Hoberg (2010) and later applied to ICO white papers by Florysiak and Schandlbauer (2019). The main idea is to generate metrics for the standard and informative content communicated by the ICO issuer in their white paper. Starting with our original corpus, we first filter out the tokens that fail to report either the start or end dates of their ICO phase. Next, we clean the data by removing punctuation, symbols, new lines, etc. After this initial round of clearing, it was apparent that our corpus contained values corresponding to either misspelled words or a series of words connected to one another. To mitigate these effects, we used the wordninja

package to separate such terms and store them as their own We then tokenize each document and lemmatize the resulting tokens. We then generate the term frequency-inverse document frequency (tf-idf) matrix for our corpus.

As mentioned in Florysiak and Schandlbauer (2019), we analyze our data based on industry classification and ICO start dates. They postulate that standard information content is characterized by the terms used in concurrent white papers as well as those in the same industry. Due to the lack of uniformity in the categorization of ICOs along with the absence of industry standards, we use an unsupervised approach to generate industry labels. In particular, we investigate the performance of MiniBatch K-Means Clustering and Gaussian Mixture Models in clustering the ICO white papers.

After clustering the white papers into industry categories, we also partition the ICOs into sets based on ICOs that have been released up to 90 days prior to the start of the particular token's release. The goal of anlayzing these sets is to estimate the amount of exposure an ICO issuer has had to the content of recent ICOs. To achieve such an estimation, we take the average of the normalized document term vectors as shown below:

$$\text{norm}_{rec,i} = \frac{1}{K}\sum_{k=1}^{K}\text{norm}_{tot,k} \tag{5}$$

As mentioned before, K is the set of all ICOs released up to 90 days before ICO i's release. Additionally, we also take the average of the normallized document term vectors for ICOs within the same industry.

$$\text{norm}_{ind,i} = \frac{1}{P}\sum_{p=1}^{P}\text{norm}_{tot,p} \tag{6}$$

After calculating the average industry and recent normalized document term vectors, we run a regression model in which our target variable is the total normalized document term vectors.

$$\text{norm}_{tot,i} = \alpha_{rec,i}\text{norm}_{rec,i} + \alpha_{ind,i}\text{norm}_{ind,i} + \epsilon_i \tag{7}$$

From the model results, we take the sum of the norm coefficients to calculate the standard content score.

$$\alpha_{standard,i} = \alpha_{rec,i} + \alpha_{ind,i} \tag{8}$$

We then take the absolute sum of the residuals as the informative content.

$$\alpha_{informative,i} = \sum|\epsilon_i| \tag{9}$$

This measurement represents the terms that deviate from both industry and recent content. This material is neither good or bad, but may be an indicator for how reliable the ICO may be. After calculating these metrics, we have effectively quantified information content with respect to ICOs.

C. ICO Text Classification with Deep Learning

Word Embeddings

To train deep neural networks, we need a way to encode the words in the corpus in a representation that the ANN "understand" and can extract as much meaningful information from. One approach would be one hot encoding but given our dataset, that would be a very memory inefficient approach. Another alternative would be to train a shallow neural network to map each word to a vector space. The idea is that words with similar context will be mapped to vectors with similar value. The computational cost required to train such transformation is immense as, due to the time constraints, we only train the model for 20 epochs on a fraction of our dataset. For this project, we believe that these constraints should not invalidate any preliminary results from this analysis. We plan to make our code more efficient and train for longer epochs over the entire dataset in future.

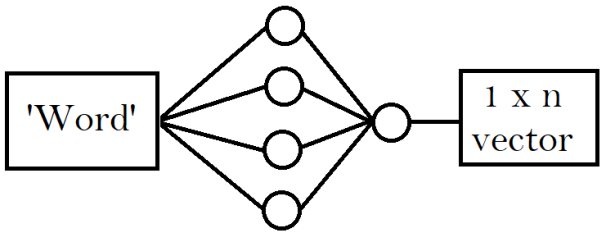

Fig. 1. Word Embedding Model Architecture

We utilize PyTorch for creating a word embedding model. The first step was to give each word some context for the model to learn. This was done by supplying each word with 2 words preceding it and 2 words in its succession as context. We shall refer to this tuple as the word-context ngram. The next step was to assign each word a unique key/index to lookup its vector implementation after training the embedding model. We created a NGramLanguageModular model as described in the PyTorch documentation to generate our word embeddings.

After training this model with the corpus subset, we created a few helper functions that translate the corpus information to our models in a readable vector format. This was done by initializing a torch tensor with zero values and shape (num_docs, vector_length, 5000). Here 5000 is an arbitrary number to fix the size of our training sample for simplicity. This length was chosen by dividing the total words in the subset by the number of documents and rounding to the nearest 1000. As one might expect, the actual document length might be lower or higher than this, hence the zero initialization. The words were then converted to vector formats using the trained embeddings and copied over to this tensor. A few more helper functions then restructured this tensor for the models they were used in.

Perceptron

The perceptron is probably the simplest Feedforward neural network. It was set up as a benchmark model but gave decent results in the prototyping phase, so we decided to include it in this report as a benchmark to compare our model with. Our Perceptron consists of a singular "neuron" that takes the input per document as a flattened 1d array and

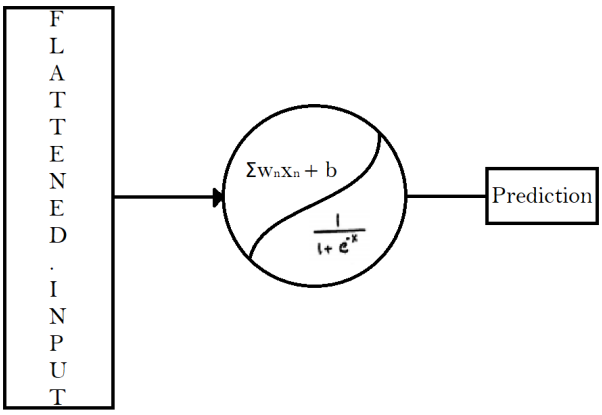

Fig. 2. Perceptron Architecture

uses sigmoid nonlinearity to predict a classification for the document. The model was coded to have a train, evaluate, save, and load functions to reduce complexity for the end user. The model utilizes ADAM optimization and BCE Loss (Log loss for binary classification) to update its parameters during backpropagation.

CNN Model

The CNN (Convolutional Neural Network) model utilizes concepts from computer vison and applies them to the word vector document implementation. By using filters and windows, we can drastically reduce computational cost and parameter size while achieving similar or better performance as traditional dense neural nets. CNNs are a class of sparce networks since they share parameters per learnable window. The model architecture can be seen in the figure below Figure 3 . The model was coded to have a train, evaluate, save, and load functions to reduce complexity for the end user. The model utilizes ADAM optimization and BCE Loss (Log loss for binary classification) to update its parameters during backpropagation.

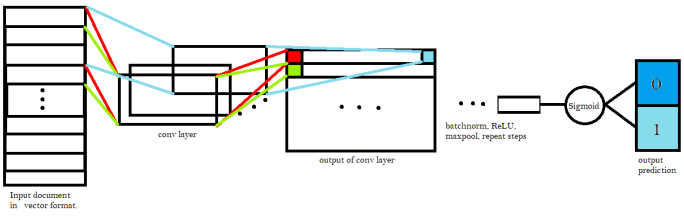

Fig. 3. Convolutional Neural Network Architecture

Long Short-Term Memory (LSTM) Model

LSTM model implements PyTorch's provided LSTM class. Each LSTM module has 1 hidden layer with 3 nodes. Before passing the data to the LSTM layer during training, we drop 20% of the data randomly which translates to a probability of 0.8 for each input word to be passed through. This is known as dropout and in theory with large datasets, it should give better performance by reducing overfitting. The final hidden

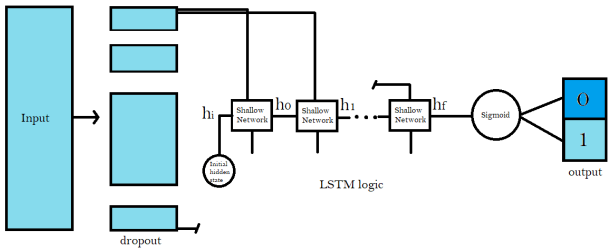

Fig. 4. LSTM Network Architecture

state of the LSTM model is passed to a linear to sigmoid sequence that outputs the predicted binary classification. The model architecture can be seen in Figure 4. The model was coded to have a train, evaluate, save, and load functions to reduce complexity for the end user. The model utilizes ADAM optimization and BCE Loss (Log loss for binary classification) to update its parameters during backpropagation.

III. Results

A. Unsupervised Learning

As mentioned in the previous section, we begin with Mini-Batch K Means Clustering. To evaluate our model, we use the Elbow Method, which plots the SSE with respect to the number of clusters defined.

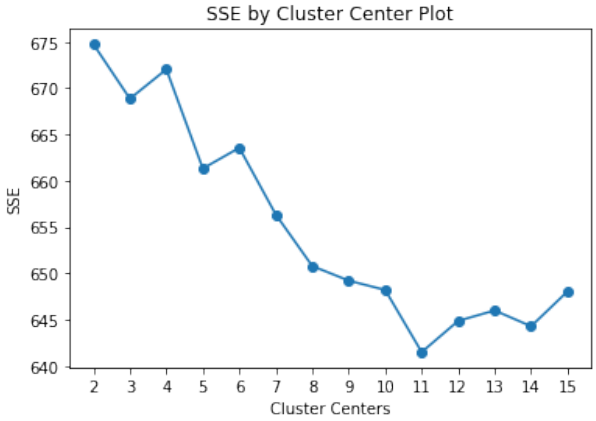

Fig. 5. Elbow Method

Based off of the results from the Elbow Method, we know that the elbow is in the range between 6 and 9 clusters. To measure the performance we use silhouette scoring. Through this method we found that 7 clusters yielded the highest silhouette score at around 0.01.

Furthermore, we implement a Gaussian Mixture Model using GridSearch to resolve the optimal number of clusters. The best model was determined by selecting the model with the lowest Bayesian Information Criterion Score. After completing the GridSearch we observed that the model selected accounted for 8 clusters. We perform PCA dimensionality reduction and plot the resulting labels.

From the plot we clearly observe that the Gaussian Model is not a good fit for our dataset. We acknowledge that although the results may not be conclusive based on the silhouette score

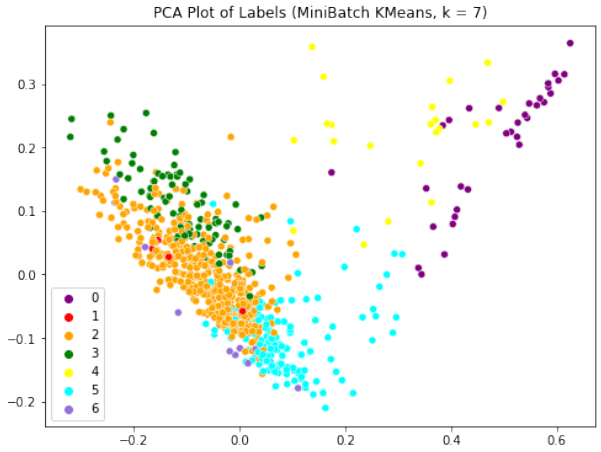

Fig. 6. PCA plot of

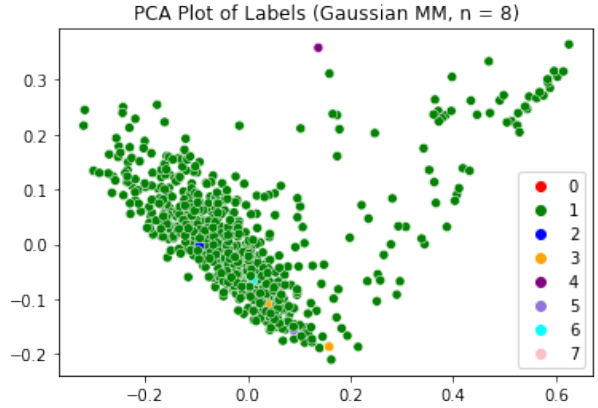

Fig. 7. PCA plot for Gaussian Mixture Model

of the K Means, we carry on with our analysis to demonstrate the process of resolving standard and informative content in our document corpus.

After conducting the calculations for the information content described in the previous section, we plot a sample of our results in Table 1.

Name	Cluster	Standard Content	Informative Content
Mindsync	2	0.988	0.894
PointPay	2	1.562	0.676
Coinimp	2	1.443	0.783
SaTT	4	1.407	0.737

TABLE I
CLUSTER AND INFORMATION CONTENT SCORES FOR SAMPLE ICOS

B. Deep Learning Models

We train the Word Embedding Model on a sample of 6 documents. As described in our methodology section, the Word Embedding Model vectorizes the documents as input for the CNN, Perceptron, and the LSTM models. In the absence of price data, we generate labels based on whether the information of the collected ICOs were available online. This was facilitated by scraping websites such as CoinCheckup for their current and archived tokens. The thought process follows the notion that if information on a coin offering is no longer available, it is safe to assume that the issuers are not as trustworthy as those who are archived.

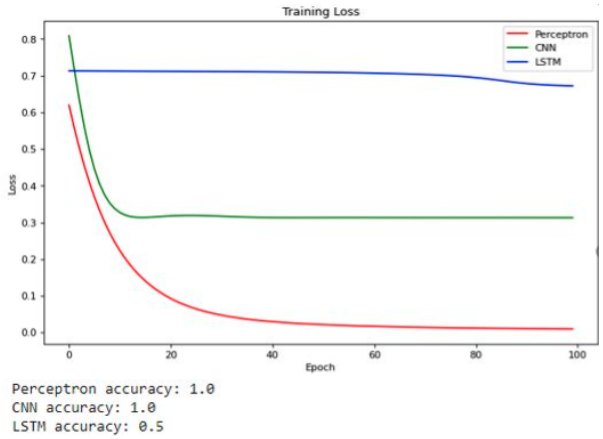

Fig. 8. Deep Learning Model Results

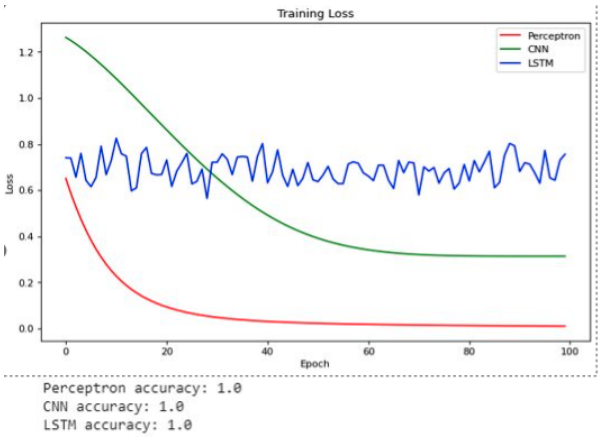

Fig. 9. Deep Learning Model Results

The results displayed above show the training loss over 100 epochs for the LSTM, Perceptron, and CNN models. The models are trained with a learning rate of 0.00001 The model accuracy is displayed below each plot. We note the instability of the LSTM model performance. We observed that although it accurately selects the correct label for the second plot, the results are not reliable due to the fact that it converges poorly over the maximum epochs. The Perceptron Model performs best out of the three although the results for the Convolutional Neural Network,

IV. CONCLUDING REMARKS

Overall, we have demonstrated several methods that utilize Natural Language Processing in unison with Deep/Machine Learning in order to extract informative insights on ICO profiles based on their white papers. While our results may suffer due to the size of our dataset, we stress that our research is a baseline for future work. We believe that our results highlight the power of text analytics in tackling the issue of information asymmetry. We note that the standard and informative content

can be utilize as an effective feature for classification tasks with respect to documents. In particular, we believe that we can extend our analysis to create a reliable metric in measuring ICO reliability. The future scope of our work involves the implementation of our scores to classify ICOs. We also aim to train our deep learning models on a larger dataset and compare the results to the classification task used with our content scores. With such research in place, the business value of our project is nested in the opportunity to create industry standards for ICO quality control. By leveraging NLP and Deep Learning, we aim to increase public trust in ICOs and lay the foundation for fair practices in the cryptocurrency market.