SIC Find!: Matching Contract Clauses to Industry of Origin through SIC Industry Code Prediction

Timothy Bierer
Student, University of Maryland, College Park
College Park, MD
tbierer@umd.edu

***Abstract*— This project utilized the Apache Spark framework to annotate, augment, and analyze a large dataset of legal contract clauses. The result indicated that while there were some industry types which were detectable through the natural language, the majority of the industry types were not. The likely cause of this result is the commonality of daily operations of the sort filed with the SEC between companies of disparate types.**

Keywords— Apache Spark, contracts, natural language processing, nlpaug, Spark NLP

I. PURPOSE AND OVERVIEW

The purpose of this project was an attempt to predict the origin industry of contract clauses through natural language processing while utilizing the Apache Spark engine. This work utilized the LEDGAR [1] dataset, which provided both the JSONL clause file, but also scraped web materials which allowed for the automated annotation of the clauses to match them with their related industries.

If successful, the plan was to apply the model created here to collateral work being done by the author through an independent study with Adobe Research Lab. The primary project concerns creating symbolic understanding of contracts through their natural language, with the final goal being the creation of meaningful synthetic contracts. The ability to predict the associated industry would assist in that effort by narrowing the scope of clauses being utilized given the end purpose of the contract.

The Apache Spark ("Spark") framework was utilized wherever possible given the large size of the dataset. The dataset was augmented to correct imbalances between the clause types, and then a model trained to predict the associated Standard Industry Classification ("SIC") code prefix, which relates to the industry category in question. The results were mixed however, with some industries not being able to be predicted, but others were able to be detected with a high rate of accuracy.

II. OVERVIEW OF THE TECHNICAL COMPONENTS

A. LEDGAR Dataset

The LEDGAR dataset, referenced above, was created in an effort to extract labels from the text of contract clauses in an effort to categorize them. Approximately 1.85 million clauses were categorized in this manner and were placed in a JSONL file, the primary component of the dataset, along with the associated labels and a relative link to the scraped source file from which it was extracted.

The clause text was pulled from contracts posted on the Securities and Exchange Commission's ("SEC") EDGAR filing system, and the researchers in that project used web scraping techniques to extract the language used to create the clause JSONL file. The relative link points to the scraped source of the contract, saved in .html files, which was made available in a repository of folders which could be downloaded from the same source as the JSONL file. These repository files often also contained the SIC codes for the industries in question, and, where it did not, it lists the trading names for the company which registered the contract with the SEC. As will be detailed further below, this allowed for the automated labeling of the dataset.

B. Spark NLP

To understand the nature and approach of the project a brief overview of the Spark engine is necessary. Spark is a multi-language engine for big data applications [2], and this project utilized the Python variant, specifically the version put forth by John Snow Labs for natural language processing ("Spark NLP"). Spark utilizes distributed computing, which involves a cluster manager responsible for partitioning and distributing instructions to worker nodes to execute. The system utilizes a large memory footprint to accomplish this, which allows for faster execution but with the drawback of requiring enhanced system requirements.

Spark NLP was specifically chosen for its integrated toolkit for performing the complex natural language processing ("NLP") tasks within the Spark framework. Given the large size of the datasets involved in the project, the enhanced computational capabilities of the system was extremely important to perform the operations in a timely and efficient manner. Utilizing specialized and purpose-built NLP tools also simplified the categorization task and gave greater assurances of viable results rather than scratch-building an evaluation framework from other more general machine learning toolkits.

C. Web Scraping and SIC Codes

Web scraping is the automated process of obtaining the source from web pages, parsing through the data, and extracting meaningful entries from it. In this project, this technique was used to both extract SIC codes from local sources from the companion folder repository which came with the dataset, but also allowed for any missing SIC codes to be looked up online without any human interaction.

SIC codes are a series of numbers which are used to identify the primary industry activity of companies. The first two numbers represent the broad industry category, and the remaining numbers offer greater granular insight into the specific activity within that category. For the purposes of this project, the first two numbers of the SIC code were utilized to identify the category in question, and will be used synonymously with the full SIC codes.

D. Data Augmentation

Imbalanced datasets produce poorer results, as certain aspects of the dataset are more represented to the model than other areas, leading to bias toward the more represented aspects. Data augmentation is the process of producing synthetic data within the dataset to reduce the level of imbalance between the categories within the dataset, thereby improving performance [3]. The synthetic data may or may not be constructed in a manner which is readable or comprehensible to humans, but that isn't a requirement for the balancing of a dataset.

E. Accuracy Metrics

While a number of accuracy metrics were captured in the training and prediction of the models produced, there are three primary metrics which will be discussed in this report: precision, recall, and F1 score [4]. Precision represents the prediction value of the model and pertains to the number of correct predictions out of the total positive predictions made (the total including both the true and false positive predictions). Recall represents the proportion of the true positive predictions that the model managed to find within the data. The F1 score represents the harmonic mean between precision and recall, which gives a better indicator than precision and recall would do individually.

F. Cloud Computing Resources

Cloud computing services allow users to access hosted system resources for a variety of tasks, including machine learning. Given the size of the computations performed during this project, only paid services were feasible options to successfully train the models at play. Given the author's limited system resources and lack of a graphics processing unit ("GPU") capable of accelerated computing within the Spark NLP framework, the computations were done by means of Google Colab's Pro tier. This allowed the author 26 gigabytes of memory and usually a Tesla P-100 GPU.

This cloud computing system was chosen due to the peculiarities of other similar services at the time of writing. The Google Colab Pro+ tier was not a feasible option given a lack of available resources and usage throttling. Attempts to use Amazon's Sagemaker cloud computing service were also unsuccessful due to kernel idling during long-running operations. As such, despite large expenditures of time and treasure in pursuit of the higher, guaranteed resources other options would have otherwise provided, the Colab Pro tier was chosen as the most economical and reliable. However, given the relatively small amount of resources provided by this service and the setup of that system, some tradeoffs had to be undertaken as will be detailed in the related sections.

III. PROJECT STAGES

A. Annotating the Dataset

The first stage for creating the intended model was to create the dataset upon which it would be based. As indicated above, the dataset for this project is based upon the LEDGAR clause language, which was published in JSONL format. The author also downloaded the scraped materials repository along with the JSONL file, and utilized the relative links contained within the JSONL file to access the associated scraped materials for each clause. See generally the "SIC-annotator.py" file included with this report and the "sec_crawl_data.tgz" file available for download in conjunction with the dataset at the link provided in the "resources.txt" file.

During this process, if it was present, the SIC code was extracted from the index file present in the folder referenced by the relative link by means of local web scraping. If it was not, the company name was extracted and a website was queried with the company name and attempted to be extracted through that method. The code would then be added to the clause language and added as an entry to a newly created JSONL file. However, 344 contract sources were not able to have SIC codes found through either method, and were discarded. Attempts to find the codes through human interaction also failed as there was sufficient ambiguity present to make a meaningful choice impossible. Given the volume of contract sources present in the original dataset, the high amount of time required to search out this small proportion of missing codes, and the uncertain chances of success, their removal was deemed acceptable.

It bears mention this operation was required to be done locally due to the slowness with which Google Colab and Drive handle operations which require locating a great number of resources within subdirectories. Upon starting the script on the author's local machine, it was found that insufficient memory was present to undertake the necessary operations through the Spark framework. The operation was undertaken in the slower, but available, traditional manner, and the "SIC-annotated.jsonl" file was created.

B. First Model

With the dataset annotated, the first attempt at modeling could be undertaken. See generally the attached file now entitled "SIC-predict-imbalanced.ipynb" for reasons which will be discussed.

Within this script, a dictionary, with the clause text as the key, was utilized to load in the data in order to eliminate exact duplicate entries, which would be overwritten in the process. This operation reduced the clauses from the original approximately 1.85 million to about 1.075 million. It is theorized this level of duplication was due to boilerplate language and language reuse. Using a dictionary format not only reduced the number of repetitive entries which would add little to the modeling, but also to reduce the computational complexity of the model. The first two digits, representing the industry category attached to the clause was extracted and the data modified into training and testing datasets with an 80/20 split.

The data is then fed through a custom ClassifierDL pipeline, which consisted of seven stages (see generally [5]). These represent annotator functions, which prepare the inputted data for training and testing [6].

The first four stages transform the data into usable form by the Spark NLP framework, and otherwise clean the data. DocumentAssembler transforms the data into usable form. Tokenizer then tokenizes the text. The third element, Normalizer, cleans up the tokens and normalizes the inputs based upon a dictionary. StopWords then removes stop words from the input, which add little to the understanding of the sentence and are discarded. The stock values for these annotator functions were utilized.

The fifth element of the pipeline is LemmatizerModel, which used a pretrained model to lemmatize the inputted text utilizing the 100 million word British National Corpus as the foundational dictionary [7]. Lemmatization is the process by which words are grouped using their common meanings (such as "like" being grouped with "likes").

Sixth, vectorization embedding was done through WordEmbeddingsModel utilizing GloVe. GloVe is an unsupervised algorithm used to produce the vector representations of words to produce connections between them [8]. These connections allow for the connections between words to be derived.

Finally, SentenceEmbeddings was utilized to create an average of each clause in question. This provides the ability to measure the similarities between the clauses, even if they are of different lengths, which is very much the case in this project.

The pipeline was then run, utilizing the stock settings and ten epochs. The results were very poor, with the only non-zero F1 score at 0.22/1.0, representing SIC code prefix 28 (Chemicals and Allied Products). As there are 71 SIC prefixes, random represents approximately 0.014. It was realized that at least some of this poor result was likely due to the highly imbalanced classes- the most represented class had 135,824 entries (SIC prefix 28), whereas the smallest only had 32 (SIC prefix 41- Stone, Clay, Glass and Concrete Products). The output can be viewed in the attached file "imbalanced-output.txt."

C. Data Augmentation

Given this imbalance, the dataset was then normalized by means of the nlpaug library, which was used to replace up to 25 words in each affected clause with synonyms, thereby adding additional clauses to under-represented categories with as little duplication as possible, although exact duplicates were allowed. See the attached file "nlpaug.ipynb." The types of targeted clauses and the number of replacements was calculated based upon the proportion of the largest number present in a category (135,284 as above) divided by the number of every other category, rounded to the nearest whole number. For example, in the case of prefix 41, 135,284 / 32 yielded 4,227.625 (rounded to 4,228) duplicates. The dataset was then looped through, with the appropriate number of synonymous duplicates created. The resulting "SIC-aug-list.jsonl" file, available for download by means of the link provided in "resources.txt," expanded to over 9.9 million entries as a result.

D. Second Model

After the data was augmented, a second model was created. See generally the attached file "SIC-predict-balanced.ipynb." This utilized the same pipeline as before, with the exception of only taking every 10th entry (for a total of 993,187 clauses) due to system constraints in Google Colab.

As shown in the attached file "balanced-output.txt," each category was much more evenly represented, with the lowest number present having 1,976 entries and the most having 3,878. While some imbalance is present from the nature of taking every 10th line, the limited disparity was deemed acceptable by the author in that the entries with the lowest values were actually those with the fewest additional entries added and the results between the imbalanced and balanced runs for those categories were the same or better with the additional balancing.

Results were much more favorable with the second model, with 25 non-zero F1 scores which ranged from 0.02 (several SIC prefixes) to 0.84 (prefix 41).

IV. ANALYSIS

While the balanced model produced better results, there were only 25 non-zero F1 scores out of 71 SIC codes. The

author, having been both legally trained as well as spending a considerable amount of time with the LEDGAR dataset due to other projects, was surprised the differences between the categories was as low as it is.

However, upon reflection, the author has come to the conclusion that, although there are certainly specialized clauses which are discernable, the commonalities of routine operations between companies is largely the same. For example, a fast-food giant, a chemical manufacturing firm, and a large clothing retailer have very different primary industry sectors, but would have similar SEC filings when it comes to executive compensation packages, ERISA plans, building rent, and the similar. Given this, there is a significant subset of industries which are able to be differentiated based upon their natural language. In short, the SEC only requires filings for specified events, and the contracts contained within only represent a narrow subset of what is actually produced. Once stated in those terms, the nature of the matter becomes clearer and the unexpectedness of the result becomes less so. However, it is for these reasons that quantitative analysis through analytics of this type is highly useful.

V. Future work

Given this differentiation, future work could be undertaken which would allow for better prediction. The nature of filings with the SEC limits the scope of the contract language available for each industry, and seeking out non-SEC-related data could allow for greater accuracy in detecting the difference between industry types. This would provide greater differentiation between industries as more niche matters would be contained within these documents.

However, it is recognized that such an undertaking would likely not have the benefit of the automated annotation process described above. Experience has shown that manual annotations are a time-consuming process when the contents of the annotations to be applied are readily knowable, and much more difficult when the information in question needs to be sought out. In spite of this, however, given this first blush of insight through this project, such an effort may prove to be worthwhile.

[1] D. Tuggener, P. von D¨aniken, T. Peetz, M. Cieliebak, "LEDGAR: A Large-Scale Multilabel Corpus for Text Classification of Legal Provisions in Contracts", Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020), pages 1235–1241, Marseille, (11–16 May 2020).

[2] M. Zaharia et al., "Apache Spark: A Unified Engine for Big Data Processing," Communications of the ACM, Volume 59, Issue 11, pp. 56-65 (2016).

[3] E. Maw, "Data Augmentation in NLP: Introduction to Text Augmentation," https://towardsdatascience.com/data-augmentation-in-nlp-2801a34dfc28, (April 12, 2009).

[4] J. Korstanje, "The F1 Score," https://towardsdatascience.com/the-f1-score-bec2bbc38aa6 (August 31, 2021).

[5] V. Kocaman, "Text Classification in Spark NLP with Bert and Universal Sentence Encoders," https://towardsdatascience.com/text-classification-in-spark-nlp-with-bert-and-universal-sentence-encoders-e644d618ca32 (April 12, 2020).

[6] John Snow Labs, "Annotators", https://nlp.johnsnowlabs.com/docs/en/annotators", (2021).

[7] John Snow Labs, "English Lemmatizer", https://nlp.johnsnowlabs.com/2021/11/22/lemma_antbnc_en.html, (2021).

[8] J. Pennington, R. Socher, and C. Manning, "GloVe: Global Vectors for Word Representation," (2014).