Multimodal Machine Learning on Financial Data

Sanjiv R. Das*
Amazon Web Services

John He
Amazon Web Services

Shenghua Yue
Amazon Web Services

Xin Huang
Amazon Web Services

Li Zhang
Amazon Web Services

31 January 2026

Abstract

Multimodal machine learning (ML) is an important extension of unimodal machine learning, which uses only numerical data or only natural language (or other modalities). We present a simple framework hosted on Amazon SageMaker for machine learning on what we denote as TabText (the combination of tabular and text modalities). We explore several use cases of ML on multimodal financial data. Application Programming Interfaces (APIs) that facilitate multimodal ML such as data extraction from the SEC (Securities Exchange Commission), creation of TabText, NLP (natural language processing) scoring of text attributes, and automatic multimodal ML are presented. Pre-built solutions that improve on unimodal models are showcased. We discuss extensions that may be supported by end-to-end TabText analysis in the cloud. Multimodal ML exploits alternative data and data structures and the approaches presented in this article showcase various implementation techniques and pipelines.

Contents

*The authors may be reached at: sanjivda, hezhijia, yuesheng, xinxh, lzhangza}@amazon.com. All authors are at Amazon Web services.

Preprint. Under review.

1 The Motivation for Multimodal ML

"And God said, let us make Man in our image, according to our likeness." – Genesis 1:26

Humans are, from the outset, multimodal learners, using all our senses in a coordinated manner, on different modalities of data and stimuli. It is only natural that learning machines mimic human multimodality, according to our likeness.

At the simplest level, multimodal data in finance comprises the combination of numeric/tabular data with text. Finance companies use both in decision-making, though most econometric and machine learning models used by practitioners and academics tend to focus mostly on tabular data. Text data is handled separately from tabular data. Or, text is converted into numeric scores and then merged with other numeric data, discarding the raw text itself. This throws away context. It would be best to undertake statistical analyses of tabular data and text together, and we define the term "TabText" to be the composition of both types of data.

Further extensions of multimodal data in finance involve modalities such as voice, particularly in work that analyzes earnings calls, where the tone of the call is compared to that in the historical record to detect differences in case they signify an informative signal. The text from earnings calls can also be analyzed for useful signals and/or to develop metrics that are useful for the finance industry (Calomiris et al., 2020). A related modality to voice is sound, and the decibel level in trading pits has been found to be related to market volatility (Coval and Shumway, 2001). Both modalities are amenable to conversion to TabText structures, where earnings calls can be transcribed to long text and sounds can be devolved into numeric features.

Figure 1 depicts a trader accessing multiple modalities of data (text, tabular, images, speech, sounds, etc.) while making decisions. Humans are essentially multimodal decision makers, whereas till recently, machines were not. Humans are also few-shot learners, needing only a few examples to learn skills, whereas traditional machine learners typically are trained on large quantities of data. Only recently has few-shot learning quality truly achieved levels of accuracy comparable to many-shot learning, and that too, on specific use cases (Brown et al., 2020). Children have always learned rapidly from a few examples, possibly because they have greater context from multiple modalities. In this paper we argue that financial decision-making by machines will eventually become similar to human decisions, leading to the age of "homo-economicus-machinus," with humans and machines interacting to make vastly better decisions.

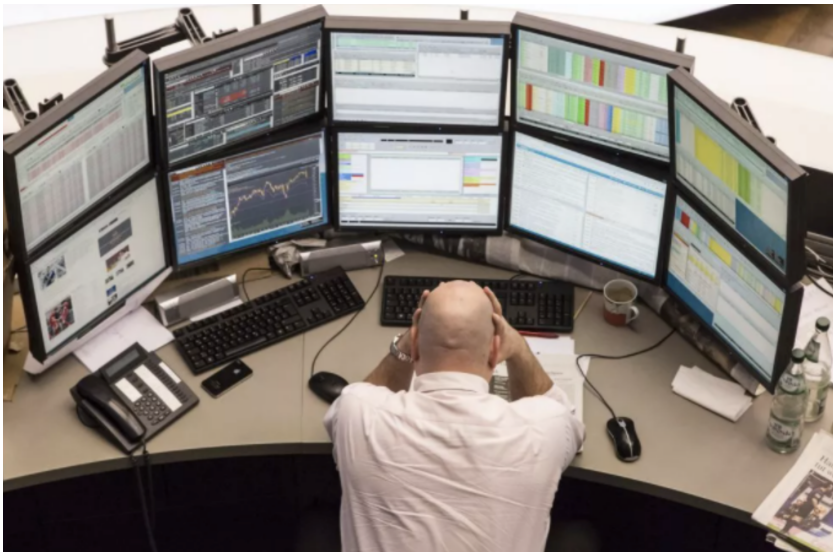

Figure 1: Multimodal decision-making is intrinsic to humans. Here we see a trader using various financial data modalities: tabular, text, images, speech, video.

Tabular data in finance is widespread, e.g., stock prices and returns, income-statement and balance-sheet data, macroeconomic variables, etc., which are all easy to obtain in large quantity and at varied frequencies. Textual data is vastly greater in quantity, e.g., financial news, tweets, analysts reports, regulatory filings, memoranda, research reports, etc. Some of these sources are public and others are private. Arguably text is the new frontier of financial data (Gentzkow et al., 2019). There is a growing literature on textual analysis in finance, see surveys by Li (2010b), Das (2014), Kearney and Liu (2014), Loughran and Mcdonald (2016), and Loughran and McDonald (2020).

The statistical analysis of financial data is primarily unimodal. Overwhelmingly, the modality is numerical and categorical data, used for various forms of regression analysis by academics and financial services researchers. The use of long-form text is often undertaken in a simplified manner by converting it into numerical scores like word counts of words from various lists that enable quantification of sentiment, litigiousness, risk, etc., as reviewed in Loughran and McDonald (2020). Modalities such as voice and images are sometimes used in a unimodal way, but not multimodal. In many use cases, e.g., ratings models, a combination of tabular data analysis combined with human judgment based on textual reports is used to generate a final prediction for a firm—inclusion of text in the model may improve the classification accuracy for ratings prediction. Further, the presence of text will also enable model interpretability as it can highlight the important words and sentences, which can be generated using techniques such as Shapley values (Shapley, 1952), permutation feature importance (for a critique, see Molnar et al. (2020)), and saliency maps (Wallace et al., 2019; Mundhenk et al., 2020).

Multimodal analysis entails the combined use of data of different types in a machine learning model. What if you could now run a logistic regression on the usual numeric dataset but simply add columns of text to it, and run the model as before? What does this require?

The process begins with tensor representations of the various types of data. For example, tabular data may be normalized and presented as a vector or tensor, even using transformers as in Huang et al. (2020). Likewise, text may be converted into BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2019) embeddings, another tensor. Images may be represented as pixel values in a tensor. These representations may be concatenated into a joint representation, or they may be combined in a weighted manner with the weights determined during model training. These representations may also be submitted to different models, which may then be ensembled as well, not just by voting schemes but weighted and stacked using weights that are also determined by training the model. True multimodal machine learning trains the weights on representations and ensembles. In this paper, we will discuss and present examples of multimodal machine learning, along with toolkits and frameworks that may be used to operationalize multimodal ML for financial services tasks.

In this article, we present one way of seeding projects with financial text, by providing an API to procure the most popular SEC filings. However, other forms of text procured by the user, such as news articles, memos, internal rating reports, tweets, earnings calls transcripts, etc., may be used just as well. The workflow using the APIs remains the same, irrespective of the source of text and tabular data.

The article proceeds as follows and offers a sequence of natural analyses that a multimodal analysis of financial data might entail. Section 2 discusses why textual analysis has widespread application in finance. In Section 3 we describe how the framework makes extraction of SEC filings facile, and large datasets can be generated with clean, parsed text with a single API call. Text summarizers are presented in Section 4. NLP scoring of the text is discussed in Section 5. Section 6 discussed the creation of TabText for multimodal ML, which is presented with several examples in Section 7. Section 8 presents domain-specific pre-trained financial transformers and their performance is shown to be superior to comparable standard transformers. Section 9 shows how multimodal ML provides for more accurate corporate credit rating prediction, and Section 10 shows how graph machine learning may be applied for credit ratings analysis, bringing in data in the network modality. Section 11 briefly discusses NLP explainability, and Section 12 shows how SEC filings may be used to domain adapt the GPT-J-6B LLM (large language model) by few-shot fine-tuning. Section 13 offers extensions and concluding remarks. We reference various papers that exemplify the frameworks of multimodal machine learning in finance. Since this is a chapter in a handbook, the goal is to offer a broad sweep of the multimodal landscape rather than a deep dive into a single aspect only.

In the next section, we discuss the use of text in finance and briefly summarize the extensive academic literature that supports the role of multimodal machine learning, which is rapidly growing in importance, especially with the advent of large multimodal models (Huang et al., 2023; OpenAI, 2023).

2 The Usefulness of Text in Finance

Recent work, e.g., Araci (2019), Desola et al. (2020), Yang et al. (2020), on language models trained on financial text deliver good sentiment prediction. Other papers show that text-based numerical features such as sentiment, readability, positivity, negativity, riskiness, litigiousness, etc., see Loughran and McDonald (2020), offer a way to reduce text to tabular features to be combined with other tabular data, reducing multimodal data to unimodal data of tabular form.[2] These are very sparse representations and other more contextually rich representations based on transformer architectures offer better results at higher computational cost.

This article discusses a new age of multimodal data analysis in finance with the addition of text to tabular data. Text offers several important advantages for financial analysis, enumerated as follows.

1. Text is *forward looking*. This often makes it more useful than tabular time-series data. Financial text in regulatory filings is explicitly couched as a forward outlook, news articles usually report on current markets in order to throw light on forecasts, opinions in tweets are meant to be prognosticative, analyst reports are detailed statements of the future paths of firms, etc. As a secondary factor, structural shifts in the world economy caused by

[2]The Loughran-McDonald word lists ae available here: https://drive.google.com/file/d/1cfg_w3USlRFS97wo7XQmYnuzhpmzboAY/view.

large-scale events (such as the pandemic) may render historical tabular data less useful. See Li (2010a).

2. Text is also *plentiful* and *multifaceted* in comparison to tabular (numerical) data (Gentzkow et al., 2019), so can cover more dimensions of financial activity.

3. Text offers the potential to enhance pure numeric/tabular data for machine learning, thereby *upgrading existing models*. For example, (i) current credit analysis of firms by rating agencies uses models for the probability of default that only use numerical data. See for example Levy and Kumar (2021). Additional text in SEC filings and news potentially improves tabular-only models of lenders, rating agencies, corporate bond funds, see Bodnaruk et al. (2015); Bonsall et al. (2017); Ertugrul et al. (2017). (ii) Asset managers may use NLP for prediction and portfolio construction, see Routledge (2019), Hafez et al. (2020), Hafez et al. (2020), Cong et al. (2019), Cong et al. (2019), Chebonenko et al. (2018). Based on text features, they can identify semantically similar firms. Or, semantic distance may be used as a metric for portfolio diversification. (iii) Corporate performance is predictable using textual features such as sentiment, readability, tone, size, risk, uncertainty, etc. These may be deployed for rank ordering the future performance of firms, see Antweiler and Frank (2004); Das and Chen (2007); Hoberg and Phillips (2016), Bach et al. (2019), etc. Text differences across time may help in ranking investment opportunities, as in Cohen et al. (2020).

4. Proprietary text offers a *competitive edge*. Whereas tabular data such as stock returns and macroeconomic variables are easily and widely available, high quality forward-looking text is hard to come by. Companies invest extensively in internal reports by their analysts and this can be used for modeling. A growing NLP literature suggests that quarterly earnings and stock performance rankings are predictable. See: Antweiler and Frank (2004); Bodnaruk et al. (2015); Bonsall et al. (2017); Brown and Tucker (2011); Bushee et al. (2018); Das and Chen (2007); Ertugrul et al. (2017); Hoberg and Phillips (2016); Jegadeesh and Wu (2013); Li (2010a); Li and Zhao (2014); Loughran et al. (2009); Loughran and Mcdonald (2011); Loughran and McDonald (2013); Loughran and McDonald (2014); Loughran and McDonald (2015); Price et al. (2012); and Tetlock et al. (2008). This competitive edge explains the growing interest in the text modality in financial machine learning.

5. Natural language processing is *scaling well*. Only recently have cloud services become easy to use for NLP. There is now a wide range of pre-trained models (see Zheng et al. (2020) for fast pre-training), such as word embeddings (word2vec, fastText, Joulin et al. (2016)) and language models in the BERT class (Devlin et al., 2019; Liu et al., 2019), achieving state-of-the-art (SOTA) results on NLP tasks including Question Answering (SQuAD v1.1), Natural Language Inference (MNLI), and others. Multi-Genre Natural Language Inference (MNLI)) is a large, popular dataset of 433,000 sentence pairs used to train and evaluate models on understanding textual entailment, where a "premise" sentence implies, contradicts, or is neutral to a "hypothesis" sentence, across diverse genres like fiction, government reports, and conversations.

6. *Democratization of NLP*. While pre-training these language models is expensive, the presence of a host of pre-trained models such as those in the Hugging Face ecosystem (https://huggingface.co) has made scaling and use of NLP models much easier and not only accessible to ML experts. AutoML (Automatic ML) frameworks are also improving ease of use, taking on the heavy lifting of fitting ML models to a dataset starting from cleaning data to producing and selecting the best models with validation metrics. Financial academics and industry analysts all have excellent analytical skills but lag behind ML computer scientists in their ability to build NLP models. Making facile use of pipelines to harvest text and undertake machine learning in a simple and easy-to-use way will go a long way to increasing the use of text in multimodal data analysis.

7. Growing research on *Long Text*. Simply scoring text for modal words (positive, negative, etc.) or converting text into count vectors or reweighted frequency vectors (TF-IDF, or Term-Frequency Inverse Document Frequency, Salton and Buckley (1988)) is fast but ignores text structure and context, as does using word embeddings. However, transformer-based (Vaswani et al., 2017) models in the BERT class (Devlin et al., 2019) are only able to deal with a small number of words, so while they maintain context, they need modification to deal with long documents, which characterize much of financial text, e.g., regulatory filings, analyst reports, legal analysis, securities litigation depositions, etc. There is a nascent NLP

literature dealing with long documents, using aggregation methods (Liu et al., 2018), and Lee and Hsiang (2019) for patents, Wan et al. (2019) for legal documents, or (Pappagari et al., 2019) for transcripts of earnings calls. Generating long sequences with sparse attention transformers is another approach, see Child et al. (2019), Zaheer et al. (2020). These are important advances that may be applied to NLP on SEC filings, which extend to thousands of words. Recently, the input length constraint has been relaxed considerably using new approaches, as discussed in Chung et al. (2025), using libraries such as in Press et al. (2022), where attention scales linearly and not quadratically.

8. *Large Language Models* (LLMs) are changing the amount of textual data needed for high-quality models. Instead of large corpora for training NLP models, with or without tabular data, a small number of labeled textual examples is sufficient to attain high accuracy levels on selected tasks. This is the result of the massive context embedded in LLMs. More recently, multimodal LLMs are being used for a wide range of multimodal tasks (Huang et al., 2023). We examine the use of these LLMs briefly at the end of this article.

These motivations for using text in particular and multimodal analysis in general lead naturally to the notion of a processing pipeline for extracting quality text, scoring it, and developing representations for multimodal data that can be used to build machine learning models. In finance, the legal and ethical milieu further demands that the models be interpretable, and generating explanations for multimodal models is an interesting exercise. In this article, we present `Amazon SageMaker JumpStart`[3], a machine learning (ML) hub to help accelerate ML with built-in algorithms, pretrained foundation models, and prebuilt solutions to solve common use cases. To support multimodal ML for finance, JumpStart offers specific tooling, presented here: https://docs.aws.amazon.com/sagemaker/latest/dg/studio-jumpstart-industry.html. In this one-stop portal, you can see links to the special purpose SDK for multimodal ML in finance, research papers, documentation, blogs, etc. A range of solutions is provided, from classification models to graph machine learning, all multimodal. Recent work on domain adaptation of large language models is also presented.

Let's begin the multimodal ML journey with the first step, i.e., the curation of high-quality text for finance, i.e., SEC filings.

3 SEC Filings

An important step in using text with tabular data is gaining access to high quality financial text. Financial NLP is a subset of the rapidly increasing use of ML in finance, but it is the largest, and the starting point for a vast amount of financial NLP is text in SEC filings. The SEC requires companies to report different types of information related to various events involving companies. The full list of SEC forms is here: https://www.sec.gov/forms.

SEC filings are widely used by financial services firms as a source of information about companies in order to make trading, lending, investment, and risk management decisions. Because these filings are required by regulation, they are of high quality and veracity. They contain forward-looking information that helps with forecasts and are written with a view to the future, as required by regulation.

There has been an exponential growth in downloads of SEC filings. As noted in the paper "How to Talk When a Machine is Listening: Corporate Disclosure in the Age of AI" (**?**), the number of machine downloads of corporate 10-K and 10-Q filings increased from 360,861 in 2003 to 165,318,719 in 2016. This number has exploded with the use of generative AI and LLMs on these documents today.

The SEC retrieval API described below in this section is deployed on Amazon SageMaker JumpStart, given the compute and storage needs of a large quantity of text. Since the intention is to ultimately undertake machine learning and NLP with big text, we used Amazon SageMaker to deploy the functionality in a processing container. SageMaker is a fully managed service that provides developers and data scientists with the ability to build, train, and deploy machine learning (ML) models quickly. Amazon SageMaker removes the heavy lifting from each step of the machine learning process to make it easier to develop high-quality models. JumpStart makes it easy to train and deploy financial ML models in Amazon SageMaker with several different machine learning and deep learning frameworks, including PyTorch.

[3] https://aws.amazon.com/sagemaker/jumpstart/getting-started/

Downloading SEC filings is done from the SEC's Electronic Data Gathering, Analysis, and Retrieval (EDGAR) website, which provides open data access. EDGAR is the primary system under the U.S. Securities And Exchange Commission (SEC) for companies and others submitting documents under the Securities Act of 1933, the Securities Exchange Act of 1934, the Trust Indenture Act of 1939, and the Investment Company Act of 1940. EDGAR contains millions of company and individual filings. The system processes about 3,000 filings per day, serves up 3,000 terabytes of data to the public annually, and accommodates 40,000 new filers per year on average.

There are several ways to download the data, and some open source packages available to extract the text from these filings. However, these require extensive programming and are not always easy to use. Below we describe a simple API call that will create a dataset in a few lines of code, for a user-specified period of time, and for a large number of tickers listed by the user.

The retrieval functionality is built into a SageMaker processing container and is called from a Jupyter notebook to enable users to download a dataset of filings with meta data such as filing dates, tickers, etc. The parsed plain text can then be used for machine learning using other SageMaker tools. Users only need to specify a date range and a list of ticker symbols and the processing container will do the rest.

As of now, the solution supports extracting a popular subset of SEC forms in plain text (excluding tables). These are 10-K, 10-Q, 8-K, 497, 497K, S-3ASR, and N-1A. For some of these, we provide examples below and for the 10-K and 10-Q forms, filed every quarter, the tool also retrieves the Management Discussion and Analysis (MD&A) section, which is the primary forward-looking section in the filing. This is the section that has been most widely used in financial text analysis. This is provided automatically in a separate column of the dataframe alongside the full text of the filing.

3.1 Install the Software Development kit (SDK)

The SEC data discussed above is load to an extracted dataframe is written to S3 storage and to the local notebook instance. In order to communicate with the processing container, there is a Python package that is easily installed using pip as follows (note that the AWS region is us-west-2). This functionality is delivered through a client-side SDK (https://sagemaker-jumpstart-industry-pack.readthedocs.io/en/latest/index.html). The first step requires pip installing a Python package (https://pypi.org/project/smjsindustry/) that interacts with a SageMaker processing container. The retrieval, parsing, transforming, and scoring of text is a complex process and uses many different algorithms and packages. In order to make this seamless and stable for the user, the functionality is packaged into a SageMaker container. This lifts the load of installation and maintenance of the workflow, reducing the user effort down to a pip install followed by a single API call. In the code below we copy the necessary files and then install the SDK.

```
! aws s3 cp s3://gecko-jumpstart/smfinance-0.0.1-py3-none-any.whl ./
! aws s3 sync s3://gecko-jumpstart/gecko-notebooks/
    SEC_Retrieval_Summarizer_Scoring_Data/ .

!pip install smfinance-0.0.1-py3-none-any.whl
```

Filings downloaded from the SEC EDGAR data repository are in malformed HTML/XML format. These are parsed into plain text, a non-trivial task. We developed a generic html/xml parser, which is easily extended to handle any kind of SEC forms, such as 10-K, 10-Q, 8-K, 497K, etc. The parser also extracts the Management's Discussion and Analysis section from 10-K, 10-Q filings. To foster ease of use, the filings are extracted using a single function call shown below in examples over the ensuing subsections, which will evidence the simplicity of the SEC extractor API.

3.2 Security Requirements

SageMaker JumpStart provides the capability of running processing containers in customers' VPCs. More specifically, when calling JumpStart APIs, customers can specify their VPC network configurations such as subnet-ids and security-group-ids. SageMaker will launch JumpStart processing containers in the VPC implied by the subnets. The inter-container traffic will be specified by the security groups.

Customers can also secure data at rest using their own CMKs (customer managed keys). SageMaker encrypts containers' EBS (Elastic Block Store) volumes and s3 (simple storage service from AWS) data using customer-provided CMKs with `volume_kms_key` and `output_kms_key`, respectively (KMS = Key Management Services).

Extracting SEC filings is also described in detail in the following blog:
https://aws.amazon.com/blogs/machine-learning/create-a-dashboard-with-sec-text-for-financial-\nlp-in-amazon-sagemaker-jumpstart/.
The blog also shows how to prepare a dashboard for textual analysis of financials of companies.

3.3 Forms 10-K/Q

These forms are quarterly reports required to be filed by companies. They contain full disclosure of business conditions for the company and also require forward-looking statements of future prospects, usually written into a section known as the "Management Discussion & Analysis" section. There may also be a section called "Forward-Looking Statements." See: https://www.investor.gov/introduction-investing/investing-basics/glossary/form-10-k.

Each year firms file three 10-Q forms (quarterly reports) and one 10-K (annual report). Thus, there are four such forms each year. The structure of the forms is displayed in a table of contents. The API we built enables easy creation of multimodal datasets.

The following block of code makes easy the downloading of the full text of the forms into a dataframe using a SageMaker session.

```
%%time
import boto3
import pandas as pd
import sagemaker
from smfinance import SECDataSetConfig, DataLoader

data_loader = DataLoader(
    sagemaker.get_execution_role(),          # loading job execution role
    1,                                       # number of ec2 instances to run the loading job, only supports 1 instance for now
    'ml.c5.2xlarge',                         # ec2 instance type to run the loading job
    volume_size_in_gb=30,                    # size in GB of the EBS volume to use
    volume_kms_key=None,                     # KMS key for the processing volume
    output_kms_key=None,                     # KMS key ID for processing job outputs
    max_runtime_in_seconds=None,             # timeout in seconds. Default is 24 hours.
    sagemaker_session=sagemaker.Session(),   # session object
    tags=None)                               # a list of key-value pairs

dataset_config = SECDataSetConfig(
    ['amzn','goog', '27904', 'FB'],              # list of stock tickers or CIKs, 27904 is the CIK for DELTA AIR LINES, INC
    ['10-K', '10-Q'],                            # list of SEC form types
    '2019-01-01',                                # starting filing date
    '2020-12-31',                                # ending filing date
    's3://yours3bucketexample/your_S3_folder',   # output s3 prefix (both bucket and folder names are required), this should be replaced by your own S3 prefix
    'dataset_10k_10q.csv')                       # output file name

data_loader.load(
    dataset_config,
    wait=True,
    logs=True)

client = boto3.client('s3')
client.download_file('yours3bucketexample', 'your_S3_folder/dataset_other_forms.csv', 'dataset_other_forms.csv')
data_frame = pd.read_csv('dataset_other_forms.csv')
data_frame
```

The code is self-explanatory, and offers various choices to the user. We note the following.

- The data loader accesses a container that processes the request. There is some latency that occurs when spinning up the container, which accounts for a few initial minutes, after which the actual filings extraction is undertaken.
- Users are not charged for the waiting time used when the instance is initializing (this takes around 3-5 minutes).
- The name of the processing job is shown in the run time log.

To begin using the SEC extractor on SageMaker, users will need to create an `s3` bucket to store the results of the extraction. In the code block below, note that the title of this `s3` bucket needs to be entered in two places, one for the output `s3` prefix, and two, for the downloaded file storage location in `s3`. SageMaker already come pre-installed with Jupyter notebooks and the `pandas` package, which enable implementation of the extractor code.

Once the notebook is open and ready, users may update any of the settings in the `data_loader` section of the code block below, and in the `dataset_config` section. For a very long list of tickers, the job will run for a while, and a progressive dots (...) stream will indicate activity as it proceeds. Users may enter tickers or CIKs (a Central Index Key is a unique SEC identifier for a company), as show below.

Important note: It is recommended to use CIKs as the input in preference to tickers. If tickers are supplied they will be internally converted to CIKs according to this mapping file: https://www.sec.gov/include/ticker.txt. One ticker may map to multiple CIKs, but the API above can only support the latest ticker to CIK mapping. It is best to provide the old CIKs in the input when you want older historical filings. This is only an issue when a ticker changes CIK (e.g., company name change, etc.), which is mostly unlikely.

The output of the DataLoader processing job is a dataframe with the columns shown below. There will be 32 filings (4 companies for 8 quarters). The filing date comes within a month of the end date of the reporting period. Both these dates are collected and displayed in the dataframe. The column "text" contains the full text of the report but the tables are not extracted. The values in the tables in the filings are balance-sheet and income-statement data (numeric/tabular) and are easily available elsewhere as they are reported in numeric databases. The last column of the dataframe comprises the Management Discussion & Analysis section. The file is written to the s3 bucket and the notebook instance. The top of the resulting dataframe is displayed in Figure 2.

Figure 2: Output of the DataLoader processing job. The column "text" contains the full text of the report. The last column of the dataframe comprises the Management Discussion & Analysis section. The file is written to the s3 bucket.

	ticker	form_type	accession_number	filing_date	text	mdna
0	AMZN	10-K	0001018724-19-000004	2019-02-01	PART I\n\nItem 1.\n\nBusiness \n\nThis Annual...	Management's Discussion and Analysis of Financ...
1	AMZN	10-K	0001018724-20-000004	2020-01-31	PART I\n\nItem 1.\n\nBusiness \n\nThis Annual...	Management's Discussion and Analysis of Financ...
2	AMZN	10-Q	0001018724-19-000043	2019-04-26	PART I. FINANCIAL INFORMATION\n \nItem 1.\n\nFi...	Management's Discussion and Analysis of Financ...
3	AMZN	10-Q	0001018724-19-000071	2019-07-26	PART I. FINANCIAL INFORMATION\n \nItem 1.\n\nFi...	Management's Discussion and Analysis of Financ...
4	AMZN	10-Q	0001018724-19-000089	2019-10-25	PART I. FINANCIAL INFORMATION\n \nItem 1.\n\nFi...	Management's Discussion and Analysis of Financ...
5	AMZN	10-Q	0001018724-20-000010	2020-05-01	PART I. FINANCIAL INFORMATION\n \nItem 1.\n\nFi...	Management's Discussion and Analysis of Financ...

3.4 Form 8-K

Next, we move on to extraction of the 8-K. This form is filed for material changes in business conditions. See https://www.sec.gov/fast-answers/answersform8khtm.html which describes the form and the various conditions that trigger a 8-K filing requirement.

The API call below is the same as for the 10-K forms and we just change the form type to 8-K. We may also include the 8-K extraction along with the 10-K extraction and download all the forms at the same time. The code below is the same API as before, with small changes for the 8-K forms.

```
%%time

data_loader = DataLoader(
    sagemaker.get_execution_role(),          # loading job execution role
    1,                                       # number of ec2 instances to run the loading
    job, only supports 1 instance for now
    'ml.c5.2xlarge',                         # ec2 instance type to run the loading job
    volume_size_in_gb=30,                    # size in GB of the EBS volume to use
    volume_kms_key=None,                     # KMS key for the processing volume
    output_kms_key=None,                     # KMS key ID for processing job outputs
```

```
    max_runtime_in_seconds=None,            # timeout in seconds. Default is 24 hours.
    sagemaker_session=sagemaker.Session(),  # session object
    tags=None)                              # a list of key-value pairs

dataset_config = SECDataSetConfig(
    ['amzn','goog', '27904', 'FB'],             # list of stock tickers or CIKs, 27904 is
     the CIK for DELTA AIR LINES, INC
    ['8-K'],                                    # list of SEC form types
    '2019-01-01',                               # starting filing date
    '2020-12-31',                               # ending filing date
    's3://yours3bucketexample/your_S3_folder',  # output s3 prefix (both bucket and
    folder names are required), this should be replaced by your own S3 prefix
    'dataset_8k.csv')                           # output file name

data_loader.load(
    dataset_config,
    wait=True,
    logs=True)

client = boto3.client('s3')
client.download_file('yours3bucketexample', 'your_S3_folder/dataset_other_forms.csv', '
    dataset_other_forms.csv')
data_frame = pd.read_csv('dataset_other_forms.csv')
data_frame
```

The number of 8-K forms is not predetermined. It will depend on the number of events that occur and require material business changes to be reported. In the example above, we obtain 108 8-K forms filed in total by the four chosen tickers in calendar years 2019 and 2020. The resulting output is shown in Figure 3.

Figure 3: Output of the DataLoader processing job. The column "text" contains the full text of the report. The file is written to the s3 bucket.

	ticker	form_type	accession_number	filing_date	text
0	AMZN	8-K	0001018724-19-000002	2019-01-31	FORM 8-K\n\nCURRENT REPORT\n\nPursuant to Sect...
1	AMZN	8-K	0001018724-19-000008	2019-02-04	FORM 8-K\n\nCURRENT REPORT\n\nPursuant to Sect...
2	AMZN	8-K	0001193125-19-049914	2019-02-25	FORM 8-K \n\nCURRENT REPORT\n\nPursuant to Sec...
3	AMZN	8-K	0001193125-19-097640	2019-04-04	FORM 8-K \n\nCURRENT REPORT\n\nPursuant to Sec...
4	AMZN	8-K	0001193125-19-103013	2019-04-11	FORM 8-K \n\nCURRENT REPORT\n\nPursuant to Sec...
...	...	...	...	...	...

Several other SEC filing types may be downloaded, such as Form 497 (disclosure of material information by mutual funds), 497K (summary prospectus), S-3 ASR (for automatic shelf registrations), N-1A (registration form for establishing open-ended management companies), etc. We do not show examples of these for parsimony.

Thus far we have seen that an important part of multimodal financial ML comprises data collection and preparation. We have seen tooling that makes this seamless and user-friendly. Next, we turn to various transformational uses of the data. For textual modality an important use case for finance is summarization. The next section highlights specific issues in financial text summarization and tooling that we developed for it.

4 Text Summarizers

Text summarization is an important step in multimodal understanding of alternate financial data. Text may come from SEC reports, earnings calls transcripts, shareholder letters, financial news, speech transcripts from the Federal Reserve, internal analyst memos, etc.

The summarizer API provides a concise summary while preserving key information content and overall meaning. SageMaker provides two types of summarizer: (i) a Jaccard Summarizer and (ii) the KMedoids Summarizer. The Jaccard Summarizer is executed with the smfinance SDK, while KMedoids Summarizer is performed within a SageMaker processing job in the container. In order to achieve good performance and better user experience, the KMedoids Summarizer processing job is able to run with multiple machine instances. The API operates on any column of text, irrespective

of whether it was obtained from SEC filings, news, etc. Both summarizers are extractive because abstractive summarizers often change the legal meaning of the sentences from that intended in the original text. This is an important issue noted in legal and financial text summarization.

4.1 Jaccard Summarizer

The Jaccard summarizer, based on (Jaccard, 1901; Costa, 2021), returns the main theme of the document by extracting the sentences with the greatest similarity to the other sentences. The metric calculates the number of common words between two sentences normalized by the size of the super set of the words in the two sentences. Summary_size, summary_percentage, max_tokens and cutoff parameters can be used to limit the size of the summarized docs. Users can also provide their own vocabulary to calculate Jaccard similarities between sentences.[4] The API call is shown below.

```
%%time
jaccard_summarizer_config = JaccardSummarizerConfig(summary_percentage = 0.1)

jaccard_summarizer = Summarizer(
    role = sagemaker.get_execution_role(),                # loading job execution role
    instance_count=1,                                     # instances number, limit
     varies with instance type
    instance_type='ml.c5.2xlarge',                        # instance type
    sagemaker_session=sagemaker.Session())                # Session object

jaccard_summarizer.summarize(
    jaccard_summarizer_config,
    'text',                                                          # text column
     name
    './dataset_10k_10q.csv',                                         # input file path
    's3://{}/{}/{}'.format(bucket, sec_processed_folder, 'output'),  # output s3
     prefix (both bucket and folder names are required)
    'Jaccard_Summaries.csv',                                         # output file
     name
    new_summary_column_name="summary")                               # add column "
     summary"
```

The result of the API is an additional column in the dataframe with the summary. See Figure 4.

Figure 4: Output of the Jaccard summarizer API, where the summary is written into an addition column as shown.

	ticker	form_type	accession_number	filing_date	text	mdna	Jaccard_Summarizer
0	AMZN	10-K	0001018724-19-000004	2019-02-01	PART I\n\nItem 1.\n\nBusiness \n\nThis Annual...	Management's Discussion and Analysis of Financ...	In our opinion, the consolidated financial sta...
1	AMZN	10-K	0001018724-20-000004	2020-01-31	PART I\n\nItem 1.\n\nBusiness \n\nThis Annual...	Management's Discussion and Analysis of Financ...	Our valuation allowances primarily relate to f...
2	AMZN	10-Q	0001018724-19-000043	2019-04-26	PART I. FINANCIAL INFORMATION\n\nItem 1.\n\nFi...	Management's Discussion and Analysis of Financ...	The effect on our net sales, operating expense...
3	AMZN	10-Q	0001018724-19-000071	2019-07-26	PART I. FINANCIAL INFORMATION\n\nItem 1.\n\nFi...	Management's Discussion and Analysis of Financ...	The increase in North America operating income...
4	AMZN	10-Q	0001018724-19-000089	2019-10-25	PART I. FINANCIAL INFORMATION\n\nItem 1.\n\nFi...	Management's Discussion and Analysis of Financ...	The increase in North America operating income...

An interesting extension of the Jaccard summarizer enables the user to overweight sentences that contain specific words. This is a simple way in which the summary can be tilted to sentences that are more relevant to a particular type of financial persona. For example, providing words that are equity market related will extract a summary that has equity market content. Or, we may want sentences that have positive and negative sentiment over other sentences; this may be accomplished by passing in a few positive and negative words that are likely to arise in text, as shown in the following modified API call (see in particular lines 1–5):

```
positive_word_list = pd.read_csv('positive_words.csv')
negative_word_list = pd.read_csv('negative_words.csv')
```

[4] https://sagemaker-jumpstart-industry-pack.readthedocs.io/en/latest/notebooks/finance/notebook1/SEC_Retrieval_Summarizer_Scoring.html

```
custom_vocabulary = set(list(positive_word_list) + list(negative_word_list))

jaccard_summarizer_config = JaccardSummarizerConfig(summary_percentage = 0.1, vocabulary
    = custom_vocabulary)

jaccard_summarizer = Summarizer(
    role = sagemaker.get_execution_role(),                # loading job execution role
    instance_count=1,                                     # instances number, limit
    varies with instance type
    instance_type='ml.c5.2xlarge',                        # instance type
    sagemaker_session=sagemaker.Session())                # Session object

jaccard_summarizer.summarize(
    jaccard_summarizer_config,
    'text',                                                          # text column
    name
    './dataset_10k_10q.csv',                                         # input file path
    's3://{}/{}/{}'.format(bucket, sec_processed_folder, 'output'),  # output s3
    prefix (both bucket and folder names are required)
    'Jaccard_Summaries_pos_neg.csv',                                 # output file
    name
    new_summary_column_name="summary")                               # add column "
    summary"
```

4.2 KMedoids Summarizer

The Jaccard summarizer captures the main thematic sentences of a document. However, this may miss some important one-off sentences where management buries some important information that is needed to be reported but which they do not wish to draw attention to. To rectify this shortcoming of the Jaccard summarizer approach, we developed the K-Medoids summarizer. The K-Medoids summarizer clusters sentences and produces as summary the medoid of each cluster. As before, the result of the API is an additional column in the dataframe with the summary. The API is as follows.

```
kmedoids_summarizer_config = KMedoidsSummarizerConfig(summary_size = 100)

kmedoids_summarizer = Summarizer(
    sagemaker.get_execution_role(),         # loading job execution role
    instance_count = 1,                     # instances number, limit varies with
    instance type
    instance_type = 'ml.c5.2xlarge',        # instance type
    volume_size_in_gb=30,                   # size in GB of the EBS volume to use
    volume_kms_key=None,                    # KMS key for the processing volume
    output_kms_key=None,                    # KMS key ID for processing job outputs
    max_runtime_in_seconds=None,            # timeout in seconds. Default is 24 hours.
    sagemaker_session = sagemaker.Session(),
    tags=None
)

kmedoids_summarizer.summarize(
    kmedoids_summarizer_config,
    "text",
                # text column name
    's3://{}/{}/{}/{}'.format(bucket, sec_processed_folder, 'output', 'dataset_10k_10q.
    csv'),          # input from s3 bucket
    's3://{}/{}/{}'.format(bucket, sec_processed_folder, 'output'),
                # output s3 prefix (both bucket and folder names are required)
    'KMedoids_summaries.csv',
                # output file name
    new_summary_column_name="summary",
                # add column "summary"
)
```

The KMedoids Summarizer processing job can support multiple instances. The name of the processing job is shown in the run time log. The output in an additional column in the dataframe is shown in Figure 5.

5 NLP Scoring

We define the term “NLP scoring” to provide a summary score for some attribute of the text. A special case of is sentiment scoring. The sentiment score itself may be a function of the positivity score and the negativity score of the text.

Figure 5: Output of the KMediods summarizer API, where the summary is written into an addition column as shown.

	ticker	form_type	accession_number	filing_date	text	mdna	summary
0	GOOG	10-Q	0001652044-21-000047	2021-07-28	FORM 10-Q\n\n(Mark One)\n\nFor the quarterly p...	MANAGEMENT'S DISCUSSION AND ANALYSIS OF FINANC...	Should Alphabet achieve a TSR Performance leve...
1	27904	10-Q	0000027904-20-000010	2020-07-15	FORM 10-Q\n\nFor the quarterly period ended Ju...	MANAGEMENT'S DISCUSSION AND ANALYSIS OF FINANC...	Capacity Reductions. Minimum Liquidity. Profit...
2	GOOG	10-Q	0001652044-20-000021	2020-04-29	FORM 10-Q\n\n(Mark One)\n\nFor the quarterly p...	MANAGEMENT'S DISCUSSION AND ANALYSIS OF FINANC...	Alphabet Inc. •Capital expenditures were $6.0 ...
3	AMZN	10-K	0001018724-21-000004	2021-02-03	PART I\n\nItem 1. Business \n\nThis Annual Re...	Management's Discussion and Analysis of Financ...	If the investment is impaired, we write it dow...
4	FB	10-Q	0001326801-20-000076	2020-07-31	PART I—FINANCIAL INFORMATION\n\nItem 1.\n\nFin...	Management's Discussion and Analysis of Financ...	This could negatively affect our business perf...

NLP scoring delivers a score as the fraction of words in a document that are in one of our internal scoring word lists. Customer can provide their own word list to calculate the NLP scores, such as negative, positive, risk, uncertainty, certainty, litigious, fraud, and safe word lists.

These numerical scores are added as new columns to the text dataframe. This creates a "multimodal" dataframe which is a mixture of tabular data and long form text, which we denote as **TabText**. As usual, when submitting this dataframe for ML, it is a good idea to normalize the columns of NLP scores, usually with standard normalization or min-max scaling.

The NLP score processing job can run in multiple machine instances. NLP scoring can be slow for massive documents such as SEC filings, which contain anywhere from 20K-100K words. Matching to word lists (usually ~200 words or more) can be time-consuming. This is why the API enables automatic distribution of the rows of the dataframe for this task over multiple EC2 (Elastic Compute Cloud) instances. In the example below, this is distributed over 4 instances and the run logs will show the different instances in different colors.

The API call to the JumpStart container for NLP scoring is shown below. Since this scoring operation is CPU (central processing unit) intensive, we prefer to use a `c5.2xlarge` machine which is optimized for CPU usage.

```
%%time
# Example 1
from smfinance import NLPScoreConfig, NLPScore, NLPScoreType

nlp_score_processor = NLPScore(
    sagemaker.get_execution_role(),          # loading job execution role
    4,                                       # number of ec2 instances to run the loading job, can support multiple instances
    'ml.c5.2xlarge',                         # ec2 instance type to run the loading job
    volume_size_in_gb=30,                    # size in GB of the EBS volume to use
    volume_kms_key=None,                     # KMS key for the processing volume
    output_kms_key=None,                     # KMS key ID for processing job outputs
    max_runtime_in_seconds=None,             # timeout in seconds. Default is 24 hours.
    sagemaker_session=sagemaker.Session(),   # session object
    tags=None)                               # a list of key-value pairs

score_types = [NLPScoreType.POSITIVE, NLPScoreType.NEGATIVE, NLPScoreType.LITIGIOUS,
               NLPScoreType.POLARITY, NLPScoreType.RISK, NLPScoreType.READABILITY,
               NLPScoreType.FRAUD, NLPScoreType.SAFE, NLPScoreType.CERTAINTY,
               NLPScoreType.UNCERTAINTY, NLPScoreType.SENTIMENT]

nlp_config = NLPScoreConfig(score_types,                          # list of score types
    's3://yours3bucketexample/your_S3_folder',  # output s3 prefix (both bucket and folder names are required), this should be replaced by your own s3 prefix
    'all_scores.csv')                           # output file name

nlp_score_processor.calculate(
    nlp_config,                                      # nlp score config
    'text',                                          # the column name of the file needs to be scored
    s3_bucket='yours3bucketexample',                 # the S3 bucket for the file to be summarized
    s3_key='your_S3_folder/dataset_10k_10q.csv')     # the S3 key for the file to be summarized
```

The resulting dataframe contains the original columns of text as well as the new numeric columns with NLP scores. See Figure 6.

Customers can also calculate their own scoring type by specifying a new word list. Below you can see the specification of custom positive and negative word lists as an example.

```
import smjsindustry
from smjsindustry import NLPScoreType, NLPSCORE_NO_WORD_LIST
from smjsindustry import NLPScorer
```

Figure 6: Output of the NLP Scorer API, where the attribute scores are written into an addition columns as shown.

	ticker	form_type	accession_number	filing_date	text	mdna	certainty	fraud	litigious	negative	polarity	positive	readability	risk	safe	sentiment	uncertainty
0	GOOG	10-K	0001652044-19-000004	2019-02-05	PART I \n\nITEM 1.\n\nBUSINESS \n\nOverview\...	MANAGEMENT'S DISCUSSION AND ANALYSIS OF FINANC...	0.050874	0.037249	0.039512	-0.039628	0.404270	0.093412	13.044649	0.053091	0.065076	0.062	0.032631
1	GOOG	10-K	0001652044-20-000008	2020-02-04	PART I \n\nITEM 1.\n\nBUSINESS \n\nOverview\...	MANAGEMENT'S DISCUSSION AND ANALYSIS OF FINANC...	0.045198	0.036123	0.037863	-0.036389	0.434447	0.092296	12.475074	0.045926	0.061519	0.081	0.029037
2	GOOG	10-Q	0001652044-19-000015	2019-04-30	FORM 10-Q\n\n(Mark One)\n\nFor the quarterly p...	MANAGEMENT'S DISCUSSION AND ANALYSIS OF FINANC...	0.046680	0.032048	0.032463	-0.039571	0.393829	0.090990	12.143007	0.050352	0.060068	0.081	0.031041
3	GOOG	10-Q	0001652044-19-000023	2019-07-26	FORM 10-Q\n\n(Mark One)\n\nFor the quarterly p...	MANAGEMENT'S DISCUSSION AND ANALYSIS OF FINANC...	0.045378	0.031815	0.032206	-0.038792	0.408259	0.092320	12.142303	0.049844	0.060728	0.090	0.030531
4	GOOG	10-Q	0001652044-19-000032	2019-10-29	FORM 10-Q\n\n(Mark One)\n\nFor the quarterly p...	MANAGEMENT'S DISCUSSION AND ANALYSIS OF FINANC...	0.045566	0.032360	0.032906	-0.038527	0.410684	0.092224	12.063071	0.050259	0.060955	0.089	0.030505
5	GOOG	10-Q	0001652044-20-000021	2020-04-29	FORM 10-Q\n\n(Mark One)\n\nFor the quarterly p...	MANAGEMENT'S DISCUSSION AND ANALYSIS OF FINANC...	0.048543	0.041417	0.039604	-0.047168	0.327390	0.093086	13.281167	0.064954	0.064141	0.047	0.037884
6	GOOG	10-Q	0001652044-20-000032	2020-07-31	FORM 10-Q\n\n(Mark One)\n\nFor the quarterly p...	MANAGEMENT'S DISCUSSION AND ANALYSIS OF FINANC...	0.047182	0.040774	0.038809	-0.047212	0.328316	0.093366	13.231845	0.064591	0.063473	0.060	0.037630
7	GOOG	10-Q	0001652044-20-000050	2020-10-30	FORM 10-Q\n\n(Mark One)\n\nFor the quarterly p...	MANAGEMENT'S DISCUSSION AND ANALYSIS OF FINANC...	0.047057	0.040868	0.039134	-0.045681	0.337955	0.092320	13.214240	0.063290	0.063739	0.061	0.037071

```
from smjsindustry import NLPScorerConfig

custom_positive_word_list = ['good', 'great', 'nice', 'accomplish', 'accept', 'agree', 'believe', 'genius', 'impressive']
custom_negative_word_list = ['bad', 'broken', 'deny', 'damage', 'disease', 'guilty', 'injure', 'negate', 'pain', 'reject']

score_type_pos = NLPScoreType(NLPScoreType.POSITIVE, custom_positive_word_list)
score_type_neg = NLPScoreType(NLPScoreType.NEGATIVE, custom_negative_word_list)
score_type_list = [score_type_pos, score_type_neg]

nlp_scorer_config = NLPScorerConfig(score_type_list)

nlp_score_processor = NLPScorer(
    sagemaker.get_execution_role(),          # loading job execution role
    1,                                       # instances number, limit varies with instance type
    'ml.c5.18xlarge',                        # ec2 instance type to run the loading job
    volume_size_in_gb=30,                    # size in GB of the EBS volume to use
    volume_kms_key=None,                     # KMS key for the processing volume
    output_kms_key=None,                     # KMS key ID for processing job outputs
    max_runtime_in_seconds=None,             # timeout in seconds. Default is 24 hours.
    sagemaker_session=sagemaker.Session(),   # session object
    tags=None)                               # a list of key-value pairs

nlp_score_processor.calculate(
    nlp_scorer_config,
    "mdna",
                 # input column
    's3://{}/{}/{}/{}'.format(bucket, sec_processed_folder, 'output', 'dataset_10k_10q.csv'),          # input from s3 bucket
    's3://{}/{}/{}'.format(bucket, sec_processed_folder, 'output'),
                 # output s3 prefix (both bucket and folder names are required)
    'scores_custom_word_list.csv'
                 # output file name
)
```

Figure 7 shows the output from using the customized NLP Scorer API.

Figure 7: Output of the custom NLP Scorer API, where the attribute scores (positive, negative) are written into an addition columns as shown.

	ticker	form_type	accession_number	filing_date	text	mdna	positive	negative
0	FB	10-K	0001326801-20-000013	2020-01-30	PART I\n\nItem 1.Business\n\nOverview\n\nOur m...	Management's Discussion and Analysis of Financ...	0.002917	0.000233
1	AMZN	10-Q	0001018724-20-000021	2020-07-31	PART I. FINANCIAL INFORMATION\n\nItem 1. Finan...	Management's Discussion and Analysis of Financ...	0.001389	0.000000
2	27904	10-Q	0000027904-20-000010	2020-07-15	FORM 10-Q\n\nFor the quarterly period ended Ju...	MANAGEMENT'S DISCUSSION AND ANALYSIS OF FINANC...	0.000773	0.000077
3	AMZN	10-Q	0001018724-21-000020	2021-07-30	PART I. FINANCIAL INFORMATION\n\nItem 1. Finan...	Management's Discussion and Analysis of Financ...	0.001591	0.000000
4	27904	10-Q	0000027904-21-000009	2021-07-14	FORM 10-Q\n\nFor the quarterly period ended Ju...	MANAGEMENT'S DISCUSSION AND ANALYSIS OF FINANC...	0.000872	0.000109
5	AMZN	10-Q	0001018724-21-000028	2021-10-29	PART I. FINANCIAL INFORMATION\n\nItem 1. Finan...	Management's Discussion and Analysis of Financ...	0.001646	0.000000

6 Creating Multimodal TabText

The SEC extractor (or any other text source) is only the first part of the financial NLP pipeline. Since ML models in finance are increasingly using multimodal data, we enhance the SEC filings dataframe with numerical data that may be stock prices, income-statement and balance-sheet data, labels for classification, etc., and created what we denote as "TabText". For this, it helps to have a use case. In the first step join stock data with filings to create a larger data frame. We will then also extend this

dataframe with additional features and implement a machine learning model for classification. We use the Paycheck Protection Program (PPP) as an example. Note that the text will not be processed into a quantitative score (as was necessary in the early finance literature on text) but will be used as text itself. This helps retain much more context than is represented by a single summary score.

The PPP (see Autor et al. (2022) for a comprehensive analysis) was created in 2020 by the Federal Government to enable employers struggling with Covid-related business adversities to make wage payments to their employees. We take the filing data from some of the companies that partook of the loans under this program. We merge this data with stock price data in order to demonstrate how to use the data join feature to merge text dataframes with numeric dataframes to create a dataframe for multimodal machine learning. A follow-up analysis will show how to use this dataframe for machine learning. For a more detailed analysis of the PPP, see Balyuk et al. (2020). The PPP program was controversial as many of the firms that borrowed money under the program ended up returning the money. The reason is interesting—the borrowing was penalized by the markets and returning the money helped these firms' stock prices recover, as well as enabled them to retain the option to borrow under the program later if needed. Further, these companies eliminated additional regulatory oversight that would follow if the PPP program was availed of.

The goal of the analysis is to examine how we may use TabText to model which companies returned the money and which firms did not. Is there enough information in TabText that is predictive? Therefore, the first step consists of creating TabText. The PPP program is reported in each company's 8-K, an SEC filing which is required when a public company experiences a material change in business conditions. In addition to a 8-K filing, 10-K and 10-Q filings, which present a comprehensive summary of a company's financial performance, are also used as source of inputs for this study, as these filings are often used to report PPP activity by companies and may contain information about whether these firms will return borrowed funds. Instead of only using text, stock data may also be helpful and can be downloaded using open source software. This is just an example and users can easily access stock prices in any way they prefer.

In total, for this example, we used 440 tickers of companies that borrowed funds under the PPP.[5] We used the SEC extractor (from Section 3) to prepare the dataframe containing 10-K/Q and 8-K forms for all these tickers. We stored these in a CSV file (351 MB) and read it in to a dataframe.

We collected over 4,297 filings for these firms in 2020 Q1 and Q2. The total time taken by the extractor to get and parse these filings is a little over an hour. Next, we downloaded a dataframe of daily stock prices and converted prices to percentage daily returns. Not all 440 tickers have stock price data, some tickers may be de-listed since the time of the PPP program. Stock splits can cause returns to be large and these are also removed in curating the dataset. We note that this is just an illustrative example and that losing some tickers from the dataset in order to get cleaner and complete data is better for the machine learning exercise to follow. The resultant stock data is saved in a file. We lose 12 tickers from the 440 we started with. We will drop a few more as they do not have complete stock data. We then use a small function to convert stock prices to returns. We get a final list of 396 tickers, after removing those with incomplete data.

For each date, we are also interested in having the return on that date along with the preceding 5 days returns as well as the following 5 days returns. In addition, since the returns on the S&P 500 index are also downloaded, we can get the net returns of the stock after deducting the return on the S&P 500 index. This net return becomes a useful feature in the dataset.

The TabText is now ready for multimodal machine learning. The next section brings fruition to the process of multimodal financial ML on TabText by introducing an API that automatically fits a collection of models and provides the best model.

7 Multimodal Machine Learning on TabText

We present a first example of machine learning on TabText. Our goal here is to analyze a subset of the companies that partook money offered under the Paycheck Protection Program (PPP) and to study how their stock returns and SEC filings are related to their later decisions to retain or return the money. To recap, the PPP program is reported in each company's 8-K, an SEC filing which is

[5]We are grateful to Professor N.R. Prabhala at Johns Hopkins University for giving us this list of tickers and the subset of these firms that returned the funds.

required when a public company experiences a material change in business conditions. In addition to a 8-K filing, 10-K and 10-Q fillings, which present a comprehensive summary of a company's financial performance, are used as source of inputs for this analysis.

In the next subsection, we run a plain vanilla analysis on TabText, and show that we achieve around 92% accuracy in classifying filings for companies that return PPP money. That is, lagged stock returns (tabular data) and the text of SEC filings is sufficient to discriminate between companies that took PPP money and subsequently returned it (for the reasons stated in the previous section) and companies that did not return the money because they actually needed it. Detailed documentation related to this analysis is here: https://sagemaker-jumpstart-industry-pack.readthedocs.io/en/latest/notebooks/finance/notebook2/PPP_TabText_ML.html.

7.1 Basic ML on TabText

We flag all filings of companies that returned the PPP money with a 1 and the others with a 0. Therefore, an ML model fit to these labels is teasing out whether the text for companies that retain PPP money is distinguishable from text of companies that return PPP money. We do not use filings for companies that return the money that appear on or after the date of return of the funds. We begin by reading in the TabText dataframe.

We begin with the cumulative return for 5 days preceding the filing as one feature. Then, we add in the text columns as well. This uses TabText and illustrates multimodal machine learning. We report all the usual metrics as well as the Matthews correlation coefficient (MCC), which is a more comprehensive metric for an unbalanced dataset. Note that it ranges from -1 to $+1$, where -1 implies perfect mis-classification and $+1$ is perfect classification. (Ref: https://bmcgenomics.biomedcentral.com/articles/10.1186/s12864-019-6413-7.)

In the TabText, there are 3911 rows of data and 43 features. Important columns are the `text` column and the label column `returned`. There is also the `mdna` column for the management discussion and analysis. There are columns for the raw return, market (S&P 500) return, and the net of the two for the 5 days before, the day of, and the 5 days after the filing date. There are also some columns for return text, etc., which were hand curated and not discussed here, as these are intermediate steps to get the labels and not part of the process flow. The total return for 5 days before and after the filing date are added as new columns to the TabText dataframe (both include the day of filing).

Counting the labels shows that there are 3191 rows with the label 0 and 720 rows with the label 1, i.e., the dataset has label imbalance. Interestingly, for label 0, the pre-filing returns are higher than post-filing, and it is the other way around for the label 1 filings. This makes sense for label 1, because when money is returned firms were rewarded by the markets for their better credit standing.

It is easy to undertake AutoML on multimodal data (TabText). As an example, we will use the open-source AWS library AutoGluon (AG), which is a part of the Gluon NLP family of tools. See https://nlp.gluon.ai/. In particular, we use AutoGluon-Tabular, which is designed for TabText and has superior performance, see Erickson et al. (2020). AutoGluon processes the data and trains a diverse ensemble of ML models to generate a trained model "predictor" which is able to predict the "returned" label in this data. Very few lines of code are required to (i) install the required libraries, (ii) train-test split the data, (iii) train the model, and (iv) report all model metrics on the test dataset. Please refer to the notebook in SageMaker JumpStart for full details for installation of the necessary packages.

We use two features, the leading total return up to and including the filing date and the text of the filing. Including the label column results in a simple TabText of 3 columns. The next code block is all that is required. AutoGluon only needs to know which column is the label and it then uses all the remaining columns as features.

```
# Model input: First5, text; Predicted object: returned
df_ag = df[["First5","text","returned"]]

#TRAIN THE MODEL
train_data, test_data = train_test_split(df_ag, test_size=0.2, random_state=123)
print("Train size =",train_data.shape[0]," | Test size =",test_data.shape[0])
predictor = TabularPredictor(label="returned").fit(train_data=train_data)
performance = predictor.evaluate(test_data)

# TEST OUT-OF-SAMPLE
y_test = test_data['returned']
test_data_nolabel = test_data.drop(labels=['returned'],axis=1)
```

```
y_pred = predictor.predict(test_data_nolabel)
performance = predictor.evaluate_predictions(y_true=y_test, y_pred=y_pred,
    auxiliary_metrics=True)
```

AutoGluon will size and manage the training process depending on the machine on which it is being run. The example here is run on a `ml.m5.2xlarge` machine instance (with 32 GB RAM). AG converts the text column into a TF-IDF representation, and combined with the numerical column produces a total of 10,038 features, the runtime is also reported.

Model metrics are strong, the accuracy is 92.8%. The Matthews correlation coefficient is high (73%). The MCC is a metric that uses all the cells of the confusion matrix. It ranges from $(-1, +1))$ where -1 means the model is wrong all the time and $+1$ is when it is always right.

7.2 Extending TabText with NLP Scoring

TabText can be extended with NLP scoring. For example, the text can be scored by computing the percentage of words in the SEC filing that are litigious. This gives the litigiousness of the document and is added as a numerical column to the TabText dataframe. Likewise, we scored each filings for the following eight attributes: positivity, negativity, certainty, litigiousness, riskiness, safety, uncertainty, and fraud/negligence using word lists (lexicons) that were prepared using an automated lexicon builder. There are hand-curated word lists (Loughran and Mcdonald, 2011), which are in widespread use in finance, but we did not use those as they are limited, whereas our approach delivers many lists (e.g., fraud/negligence) that are not covered elsewhere. Some of these aspects are widely used in the finance literature for scoring text, and word lists exist to support these as well, see: https://sraf.nd.edu/textual-analysis/. For a review of the literature on scoring and financial decisions, see Loughran and McDonald (2020).

A column is added (called `cleanText`) to the dataframe by stripping the documents of numbers and punctuation, as well as stop words, and then setting all words to lower case. Stemming may be undertaken later as needed. The results remain as before with similar accuracy.

7.3 Combining AutoML with BERT models

We have undertaken ML on the PPP dataset using AutoGluon which ensembles various models (Erickson et al., 2020) but does not use transformer models Devlin et al. (2019); Liu et al. (2019). In this section, we will combine AutoML techniques with transformer embeddings for multimodal machine learning. We use transformer models from the open source domain so that these are easily available. These perform well on GPUs and we therefore use `mxnet` Chen et al. (2015) with CUDA (Compute Unified Device Architecture) support. (Recently, this deprecation of `mxnet` resulted in AutoGluon being refactored to run on `PyTorch`, fostering a successful new version.)

The entire model training and testing on the dataset from the previous section may be re-run as before with this small change. We do not repeat the results here as they also provide over 90% accuracy levels, slightly different depending on which machine the models are run on as AutoGluon changes the size of TF-IDF embedding vectors depending on memory availability.

Our next step is to combine various models used in AutoGluon (Random Forests, ExtraTrees, KNNs (k nearest neighbors), Light GBMs, CatBoost, XGBoost, and Neural Nets) with BERT models. We define two custom models that will use BERT embeddings. The BertEmbeddingLogisticRegression class will take the text column and extract BERT embeddings from it. Then the model will train a logistic regression on top of the embedding. The BertEmbeddingNN class will instead train a multilayer perceptron (MLP) on top of the BERT embedding. Note that for this definition to work, the text column in the data must be named "text" (case-sensitive).

We add the two classifiers with BERT above to AutoGluon and train them with the other default models. For the BertEmbeddingNN model, we use a MLP (multi-layer perceptron) with one hidden layer (64 hidden nodes) and train with the Adam optimizer for 100 epochs. The other hyper-parameters for the BERT classifiers can be set using the `sklearn` (Scikit-Learn) API.

Note that we will need to set `enable_raw_text_features=True` in the feature generator, that would allow us to keep the raw text after the pre-processing step of AutoGluon, and then our BERT classifiers will generate the BERT embeddings from the raw text. The generic column name for

the raw text is: `<original_column_name>_raw_text`. As before, minimal code is needed to implement the AutoGluon plus BERT ensemble, as we see below.

```
from autogluon.tabular.configs.hyperparameter_configs import get_hyperparameter_config
from autogluon.features.generators import AutoMLPipelineFeatureGenerator

custom_hyperparameters = {
  BertEmbeddingLogisticRegression: {"multi_class": "multinomial"},
  BertEmbeddingNN: {"hidden_layer_sizes": (64,), "max_iter": 100, "random_state": 1}
                        }

custom_hyperparameters.update(get_hyperparameter_config("default"))

predictor = TabularPredictor(label="returned").fit(
    train_data=train_data,
    hyperparameters=custom_hyperparameters,
    feature_generator=AutoMLPipelineFeatureGenerator(enable_raw_text_features=True),
    excluded_model_types=['FASTAI'] # excluded to preserve shared memory
)
```

This code will run all the AutoGluon models from before plus the two custom BERT models and create a weighted ensemble of them all. Once again, similarly high accuracy levels are achieved.

7.4 Multimodal ML on TabText

Rather than build custom models, we can also simply use the hyperparameter selection `multimodal` in the `fit()` function in AutoGluon, which uses TextNeuralNetV1Base embeddings. SEC filings are considerably large text, so this does take some time. First, set up the data.

Second, call the "fit" function and also use the automatic hyperparameter tuning option `presets=best_quality`. This will increase model training time but will also improve accuracy. Please note that the machine must have a GPU to use the `hyperparameters='multimodal'` option below.

```
%%time
#TRAIN THE MODEL
predictor = TabularPredictor(label="returned").fit(train_data=train_data, hyperparameters
    ='multimodal', presets='best_quality', excluded_model_types=['FASTAI'])
performance = predictor.evaluate(test_data)

# TEST OUT-OF-SAMPLE
y_test = test_data["returned"]
test_data_nolabel = test_data.drop(labels=["returned"], axis=1)
y_pred = predictor.predict(test_data_nolabel)
performance = predictor.evaluate_predictions(y_true=y_test, y_pred=y_pred,
    auxiliary_metrics=True)
```

The training took almost 3 hours. It returned improved accuracy of 94.1%. The model leaderboard is shown in Table 1.

7.5 Multiclass example with Long Documents (SEC 10-K/Q filings)

So far all examples we considered used binary labels. In order to show multimodal financial ML when labels are multiclass, we aim to predict industry of a firm from reading its 10-K/Q filings.

This example will demonstrate AutoML on train and test datasets that have been curated using the JumpStart SEC Forms Retrieval tool. We downloaded a large number of SEC 10-K/Q forms for companies in the S&P 500 from 2000 through 2019. A separate column of the dataframe contains the MD&A (Management Discussion & Analysis) section of the filings. The MD&A section was chosen for this analysis because it is the most popular section used in the financial services industry for NLP.

The task for this model is a simple one, can we train a model to detect the broad industry category of a company from the text of its MD&A section? Clearly this task is meant to illustrate the code and the mechanics of using the JumpStart SEC dataframe, and the labels themselves are simple and illustrative (we do not need an algorithm to assign firms to industries, but this is a good example of the use of long text for a classification task). However, a user may replace the labels with other labels of their choice and re-use this entire notebook with minimal changes.

Labeled datasets for SEC filings are mostly unavailable. The PPP example used in the preceding sections is a binary classification problem and the dataset contained around 4,000 filings. An extension

Table 1: Leader board of binary classification machine learning models fitted to the PPP dataset. The postfix "L1" and "L2" stand for L1 and L2 regularization.

ID	Model	Score Test	Score Val	Pred Time Test	Pred Time Val	Fit Time
0	LightGBMXT_BAG_L2	0.9413	0.9393	564.78	251.63	7053.26
1	XGBoost_BAG_L2	0.9374	0.9428	572.97	254.52	7027.02
2	LightGBM_BAG_L2	0.9361	0.9431	564.78	251.65	7066.35
3	WeightedEnsemble_L3	0.9361	0.9431	564.79	251.65	7068.12
4	LightGBMLarge_BAG_L2	0.9349	0.9405	564.83	251.61	7256.23
5	CatBoost_BAG_L2	0.9336	0.9399	566.07	259.39	7251.54
6	NeuralNetMXNet_BAG_L2	0.9221	0.9367	564.05	251.10	7089.86
7	LightGBMXT_BAG_L1	0.9144	0.9265	1.46	0.91	331.82
8	WeightedEnsemble_L2	0.9144	0.9265	1.47	0.92	333.82
9	LightGBM_BAG_L1	0.9068	0.9130	1.38	0.90	276.50
10	CatBoost_BAG_L1	0.8953	0.9003	3.47	8.58	1049.55
11	LightGBMLarge_BAG_L1	0.8889	0.9054	1.52	0.85	576.62
12	XGBoost_BAG_L1	0.8787	0.8958	9.96	3.66	191.09
13	NeuralNetMXNet_BAG_L1	0.8174	0.8181	0.50	0.38	158.88
14	TextNeuralNetwork_BAG_L1	0.8161	0.8248	545.15	235.40	4294.00

to this exercise with good labels is industry classification using SIC (standard industrial classification) codes.[6] This is a multiclass problem and will consider more filings, to provide a more extensive implementation example. Our dataset comprises firms that fall into the following categories and the count of number of firms is also shown in Table 2.

Table 2: Class counts for the `TextTab` based industry classification example.

Industry	Industry code	Counts
Mining	B	1067
Manufacturing	D	6238
Transportation, Communications, Electric, Gas, Sanitary Services	E	1516
Retail Trade	G	1438
Finance, Insurance, And Real Estate	H	2263
Services	I	2367
TOTAL	-	15430

The dataset used here comprises 10-K/Q SEC filings for 2010 Q1 through 2019 Q3 (39 quarters) for select S&P 500 companies, and the labels are the industry codes. The dataset contains six labels and does not comprise firms in category A (agriculture) or category J (public administration). The table above shows the counts in each industry category. There is considerable class imbalance. We fit the model with multimodal AutoGluon, which also ensembles BERT models. The code required is very simple.

```
predictor = TabularPredictor(label=TARGET_COL).fit(train_data=train_df,  hyperparameters=
    'multimodal', excluded_model_types=['FASTAI'])
performance = predictor.evaluate(test_df)
```

The `hyperparameters='multimodal'` option is used to invoke the multimodal classifier with BERT embeddings. In order to use this option it is necessary to have a machine with a GPU. For this example, we used a `p2.xlarge` machine instance with 1 GPU, 4 CPUs, and 61 GB of RAM. This base level machine can be upgraded of course. The results for the test data show: accuracy = 0.91, balanced accuracy = 0.91, MCC = 0.819, ROC AUC = 0.963, f1-score = 0.902, precision = 0.892, recall = 0.912. We reran the models without BERT embeddings and achieved lower accuracy but better run times—the entire training was completed in one hour. Results are shown in Table 3 and are of high accuracy.

[6] https://www.naics.com/search/

Table 3: Multiclass classification leader board of models used for text-based industry classification. The classes are shown in Table 2.

ID	Model	Score Test	Score Val	Pred Time Test	Pred Time Val	Fit Time
0	LightGBM	0.8234	0.9466	0.60	0.27	305.57
1	CatBoost	0.8169	0.9459	0.98	0.77	578.97
2	XGBoost	0.8149	0.9474	4.49	1.59	1533.93
3	WeightedEnsemble_L1	0.8053	0.9489	136.98	56.84	8257.86
4	LightGBMXT	0.8028	0.9459	0.55	0.32	224.87

8 Pre-Trained Financial Transformers

Text and image data has to be represented as a vector, matrix, or more general tensor in order for it to be used in machine learning. Recent advances in transformer technology (Vaswani et al., 2017; Devlin et al., 2019) have enabled highly contextual vector representations that have improved the accuracy of supervised learning on textual data. The machine learning solution in the previous section may take advantage of vector representations of many types, e.g., Mikolov et al. (2013), Mikolov et al. (2017), including those generated by transformers.

In this section, we present financial transformers that are trained from scratch on general text from Wikipedia used in the RoBERTa model of Liu et al. (2019) and also ten years of SEC 10-K/Q filings for S&P 100 companies. These "pre-trained" models may be trained on only SEC text or a combination of text from the Wikipedia corpus plus SEC text. SageMaker JumpStart deployed four models that we trained. The RoBERTa-SEC-Base and RoBERTa-SEC-Large models are both trained only on SEC text. The other two models, RoBERTa-SEC-Wiki-Base and RoBERTa-SEC-Wiki Large are trained on SEC filings and Wikipedia. It is simplest to think of transformers as endpoints to which text can be submitted and a vector representation of the text is returned. In the transformers described below, the vector sizes are 768 (standard size) and 1024 (large). (Technical details: The data type used is float16, batch size is 2048, with a learning rate of 0.000625), with 250,000 training steps. Optimization is undertaken using the `neslamb` algorithm Zheng et al. (2020), with a warm up ratio of 125%. Maximum input sequence length is 512. Training was performed on a Horovod distributed training platform with 8 slots per instance, 8 instances, and a vocabulary of size 32,000.) Horovod is an open-source distributed deep learning training framework developed by Uber that makes scaling AI models across multiple GPUs and machines fast and easy, supporting TensorFlow, Keras, PyTorch, and MXNet with minimal code changes. Full details are provided in Das et al. (2021).

8.1 Using the Pre-trained Financial Transformer

In order to use one of the pre-trained transformers, it needs to be deployed to a SageMaker JumpStart endpoint. We have used the model RoBERTa-SEC-Wiki-Large, which is a fine-tuned model based on RoBERTa (Robustly Optimized BERT Approach), a powerful language model by Meta AI that significantly improves upon Google's BERT by using dynamic masking, larger batches, more training data, and longer training, leading to better performance in tasks like text classification, sentiment analysis, and question answering. We call the deployed transformer to get fixed-length text embeddings and then use the embeddings to undertake transfer learning by passing them through neural net layers, which are trained on specific data. Das et al. (2021) shows that these supervised learning based on these pre-trained transformers achieves better classification accuracy than vanilla transformer models such as BERT and RoBERTa. When applied to the well known Financial Phrasebank dataset, test accuracy from these domain-specific pre-trained models is 95.6%, about 3-4% higher than that achieved from standard transformers.

The sample dataset on text is from cryptocurrency news with binary labels for competitive news about two cryptocurrencies (label=1) and non-competitive news (label=0). The goal is to classify sentences or articles about two cryptocurrencies into the statement being competitive about the two cryptocurrencies versus the news being non-competitive. The dataset has 459 observations (rows). The binary classification model comprises a fully-connected neural network of two hidden layers with 128 neurons in each layer.

Even though only 10 epochs of training are performed on a very small dataset, we obtain 86% accuracy on the test data.

In sum, this section shows that (i) pre-training transformers on financial text improves the classification accuracy for financial NLP use cases; (ii) whereas pre-training the model can be expensive because of the extensive data and computation requirements, using the model for transfer learning is very easy and inexpensive in SageMaker JumpStart where the pre-trained model may be deployed to an endpoint and then called to obtain embeddings, used for downstream classification.

In the next section, we present a comprehensive solution that showcases how multimodal ML may be used for a large use case that showcases how unimodal ML is improved upon by multimodal ML.

9 Multimodal ML for Corporate Credit Ratings

The traditional credit rating process begins with a classifier that uses several balance-sheet and income statement features for a company. Classification may be binary, i.e., whether the company is above or below investment grade in credit quality. Classification may also be multi-category, i.e., to determine the precise credit rating of a firm, ranging from AAA through BBB (investment grade) and BB through CCC (below investment grade) to D (default). This process has traditionally been inherently unimodal and uses tabular data.

What if we enhanced the data to be multimodal by adding in the text of the Management Discussion and Analysis (MD&A) section of the 10-K or 10-Q SEC filing, which contains forward looking statements about the company's business conditions? Will we attain an improved classifier using multimodal ML? This is explored in Nguyen et al. (2021). The dataset (https://www.kaggle.com/datasets/agewerc/corporate-credit-rating) used for their model comprises the following tabular data:

- Liquidity Ratios: current ratio, quick ratio, cash ratio, and days of sales outstanding,
- Profitability Ratios: gross profit margin, operating profit margin, pretax profit margin, net profit margin, effective tax rate, return on assets, return on equity, return on capital employed, and EBIT per revenue,
- Debt Ratios: debt ratio and debt equity ratio,
- Operating Performance Ratios: asset turnover, fixed asset turnover, company equity multiplier, enterprise value multiple, and payable turnover, and
- Cash Flow Ratios: operating cash flow per share, free cash flow per share, cash per share, operating cash flow to sales ratio, and free cash flow to operating cash flow ratio.

The rating distribution described above mimics the proportions usually found in the US for corporations. The categories are six-fold, obtained by collapsing some of the smaller categories into one. This results in six classes with some grouped as follows: {AAA, AA}, A, BBB, BB, B, {CCC, CC, C, D}, with frequency counts equal to 96, 398, 671, 490, 302, and 72, respectively. The data comprises quarterly observations of ratings for each ticker and the 10-K/Q MD&A is added from the corresponding quarter. In addition, the industry code is also included as a feature, as it captures industry effects that differentiate companies in their credit quality. Using the JumpStart API for NLP scoring (https://sagemaker-jumpstart-industry-pack.readthedocs.io/en/latest/smjsindustry.nlp_scorer.html), additional columns are added for text features such as positivity, negativity, polarity, certainty, uncertainty, fraud, litigiousness, risk, safety, readability, and sentiment. The completes the feature set.

9.1 Multimodal Model Training

Several ML models are trained on the dataset such as kNN, random forest, extra (randomized) trees, boosted decision trees, neural networks, etc. A rough summary of the results from Nguyen et al. (2021) shows that for binary classification between investment grade and below investment grade classes, the error is approximately 16% on tabular data (accuracy is around 83-84%), but the addition of text from the SEC filings helps reduce the overall classification error to around 12%, i.e., a 25% reduction in the error rate. When ticker symbols are included, we see an additional 1% improvement in accuracy. Further improvement (around 1%) is seen with stack ensembling of these models and repeated k-fold bagging. Broadly speaking, inclusion of text improves credit rating classification.

In multi-category classification, the accuracy from tabular data is 58%, which increases to 64% when text is included. An additional 2% improvement in accuracy comes from including tickers, with improvement all the way to 70% accuracy when deep ensembling and bagging are applied. It is important to note that when the classifier makes an error it is usually only incorrect by one rating

category and if errors in an adjacent rating class are not counted as errors, then the classification accuracy is 95%.

Before describing the training and deployment of the model, it is useful to briefly consider how multimodal ML is implemented using stack ensembling, which is used for classification. Assume we have tabular and text data. Erickson et al. (2020) provides a good description of the stack ensembling approach shown in Figure 8 (and implemented in the AutoGluon library):

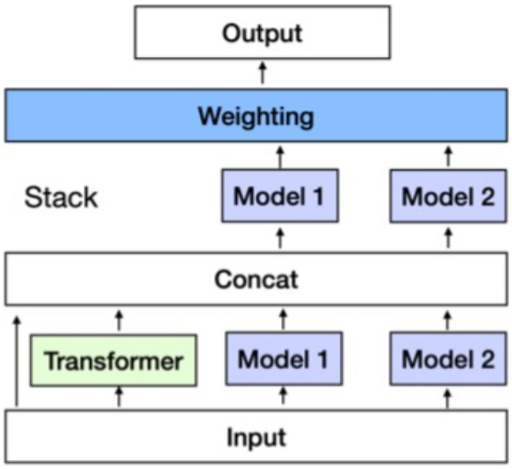

Figure 8: Diagram showing the layout of the training model with stack ensembling.

The input to the model is both text and tabular. The text passes through a transformer to create an embedding and the tabular data goes through several several base ML models, such as: kNN, Random Forest (Entr, Gini), Extra Trees (Entr, Gini), Boosted decision trees (CatBoost, XGBoost, LightGBM, LightGBM-XT, LightGBM-Large), FastAI NN, MXNet NN, etc. The original tabular data is then concatenated to the output of the transformer and the outputs of the various tabular models to achieve a fused representation of text and tabular data. This representation is then passed through the collection of ML models to generate outputs that are weighted using feed-forward neural network layers to eventually generate the output layer which is then used to obtain model predictions. The entire "stack" is trained at once and all parameters are fitted together. We can have multiple levels of this configuration to create a bigger "stack ensemble." These stack ensembles have been shown to be highly accurate for multimodal machine learning (Shi et al., 2021; Tang et al., 2024). We show its application below.

9.2 Training in SageMaker JumpStart

The infrastructure for this paper has been deployed in SageMaker JumpStart and the blog describing this application is here: https://aws.amazon.com/blogs/machine-learning/use-sec-text-for-ratings-classification-using-multimodal-ml-in-amazon-sagemaker-jumpstart/.

The dataset of the paper has been replaced with a dataset using features used by Altman (1968) in his seminal paper. Text for this solution is the MD&A section from SEC filings. The dataset has the following features (the financials are relative to normalizing total asset value to be equal to 100): text of the MD&A section, industry code, total assets, current liabilities, total liabilities, retained earnings, current assets, net sales, EBIT (earnings before interest and taxes), equity trading volume, and credit rating.

The dataset has 3286 rows. As noted, the financial data is normalized to have total assets at 100, and industry code is added (2 digit SIC codes). The last column contains the rating category. The data is converted into the famous five Altman ratios and the final feature set has the following features:

- A: EBIT/TotalAssets
- B: NetSales/TotalAssets
- C: MktValueEquity/TotalLiabs

- D: (CurrentAssets - CurrentLiabs)/TotalAssets
- E: RetainedEarnings/TotalAssets

The MD&A column is then processed using the NLP scoring API from Section 5 using code that calls the API to add columns to the dataframe with textual scores.

The multimodal train-test dataframes become the data source for multimodal ML of credit ratings. This is undertaken using a single API call to invoke the AutoGluon library, see https://github.com/autogluon/autogluon. The usage is extremely simple: the target column of the dataframe (in our case is the `Rating` column) and is specified as such in the API call. The remaining columns of the multimodal dataset will be used as features. These are automatically processed irrespective of modality (tabular, text, or images) and submitted for machine learning. The user does not have to process the text columns to make embeddings, or the categorical columns via one-hot encoding, thereby eliminating several steps. The system decides how to merge the representations of text and tabular data as well in an optimal manner. The API below will run several different ML models and ensemble them to give the best model automatically.

The API runs several models and takes a total of 15 minutes approximately. It generates several outputs, one of which is a leader board shown in Table 4, offering a comparison across all the models:

Table 4: Leaderboard for the corporate credit rating classification example. Values shown are for binary classification into investment grade rating or below investment grade rating.

Model	Validation Accuracy	Inference Time (s)	Fitting Time (s)
LightGBM	0.848	0.0744	18.979
WeightedEnsemble_L2	0.848	0.0748	19.971
LightGBMLarge	0.835	0.0597	71.022
XGBoost	0.835	0.0935	15.732
LightGBMXT	0.747	0.0593	12.937
CatBoost	0.671	0.6908	23.268
NeuralNetMXNet	0.608	0.0177	6.739
KNeighborsDist	0.494	0.1045	2.669
NeuralNetFastAI	0.468	8.0954	49.353
RandomForestGini	0.430	0.1068	7.336
KNeighborsUnif	0.430	0.1071	2.712
RandomForestEntr	0.430	0.1085	7.209
ExtraTreesGini	0.380	0.1076	7.232
ExtraTreesEntr	0.354	0.1075	7.630

We see that it is important to run several models for comparison, as various use cases will show differential performance across supervised learning models. Taking the best model, the system generates the confusion matrix, which is displayed in Figure 9 for the test dataset:

Even when the model predicts the rating incorrectly, it is interesting that it is usually off only by one rating level, suggesting that it is highly accurate given the categories in this classification problem are ordinal and not cardinal.

AutoGluon will also generate feature importance using the column permutation technique. In this approach each feature column in the dataframe is shuffled, the model is re-estimated, and the resultant drop in accuracy is recorded. Important features will evidence a greater drop in accuracy of the model than irrelevant features. Table 5 shows the top features determined by this method.

Table 5: Feature importance for the corporate credit rating classification example.

Feature	importance	std_dev	p_value
polarity	0.5522	0.0382	0.0008
MDNA	0.0808	0.0101	0.0026
C	0.0438	0.0058	0.0029
sentiment	0.0404	0.0267	0.0601
fraud	0.0101	0.0101	0.1127
risk	0.0101	0.0000	0.5000
E	0.0034	0.0058	0.2113

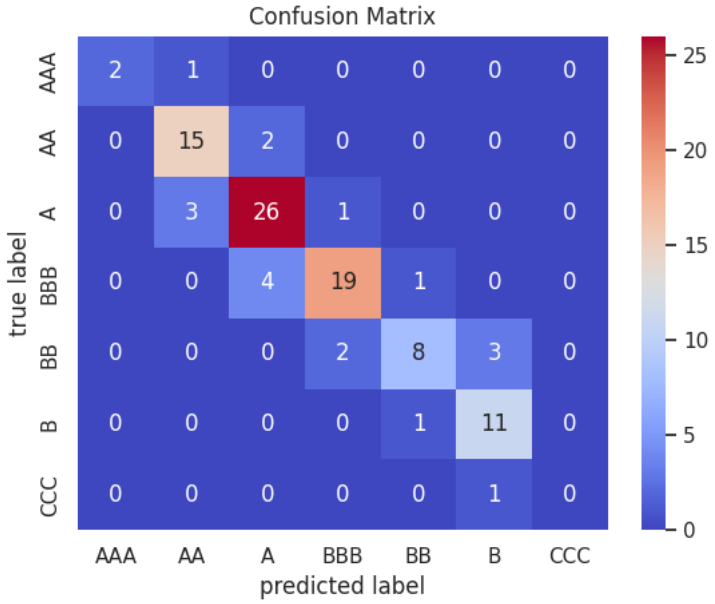

Figure 9: Confusion matrix for multiclass corporate rating classification.

We see that polarity is the most important feature. Polarity is the difference in positive and negative word counts, normalized by the total of positive and negative word counts. The text column of the MD&A section is the second important feature. Altman's C ratio, i.e., the ratio of equity market value to liabilities is third. It makes sense that this metric for leverage is an important one in discriminating across firms on credit quality. The next important features are the textual scores for sentiment, fraud, and risk, all of which should matter in comparing firms' credit ratings. The last important features is Altman's ratio E, i.e., the ratio of retained earnings to total assets. It is notable that when the additional modality of text is added to Altman's Z-score model, we see that it matters as much if not more than traditional tabular features from the balance-sheet and income-statements.

9.3 Deployment of the Model to an Endpoint

Once the model is built, it is useful to deploy it in production mode. In SageMaker JumpStart this can be done by pushing the trained model to an endpoint using very little code.

This endpoint is then called using code that can pass features for a single out-of-sample instance to the endpoint for classification. Instead of handling a single request each time, a batch of examples can also be submitted. Full details with code are available from the hosted solution here: https://aws.amazon.com/blogs/machine-learning/use-sec-text-for-ratings-classification-using-multimodal-ml-in-amazon-sagemaker-jumpstart/.

Credit scoring is a classic example where multimodal machine learning can greatly improve on unimodal (tabular) ML. An important extension is to use large language models (LLMs) to use the predicted rating and the text features of the model as context to generate an explanation of the model's predictions.

10 Using Multimodal Data in Graph Machine Learning

Data from different modalities can also include different data structures. In this section we explore how data structured as a graph (network) may be used to improve tabular credit scoring models. This section is related to Section 9 where we explored credit scoring with multimodal data (tabular plus text). Here, we convert text data about companies into a graph, where companies with similar forward-looking statements in their SEC filings are linked in a network where each node is a firm. This additional linking of companies on the graph is useful for estimating credit quality of firms because it captures credit contagion linkages, which are not possible to bring into the analysis using

tabular data alone. Full details of this application are published in **?**, and the code has been released here: https://codeocean.com/capsule/5230264/tree/v2.

In order to construct the network of companies for credit assessment, we turn to the text in SEC 10-K/Q filings, in particular the management discussion section (MD&A), which contains forward-looking statements. The text in this section for each company is converted into an embedding using an embedding model. Then, the pairwise cosine similarity of all companies is computed and a link in the network is created if two companies have cosine similarity greater than 0.5 (this is a hyperparameter that may be adjusted). This network construction approach is a simple and effective way to build a network of companies that contains information about companies that have related risks that they may pass on to each other if any of them experiences dropping credit quality.

The dataset comprises numerical and categorical features for each company such as the current liabilities, retained earnings, industry code, etc., as well as the corporate network based on the SEC filings text. Graph neural networks (GNNs) are ideal for fitting an ML model to this data. GNNs utilize all the constructed information above to learn a hidden representation (embedding) for each company, using all the tabular information about the company and that of its neighbors in the network. The hidden representation is then used as input for a linear classification layer to determine whether the company has an investment-grade or below investment-grade rating. The SageMaker JumpStart application takes care of (i) constructing the corporate network and (ii) fitting the GNN. This particular application uses the GraphSAGE framework for machine learning, see Hamilton (2020). For comparison, the `node2vec` approach of Grover and Leskovec (2016) is also implemented and gives slightly inferior results.

For purposes of comparison, we may examine the models that only use tabular data versus models that also include the corporate graph. This is presented in Das et al. (2023). The accuracy for binary classification (investment grade vs non-investment grade) from purely tabular models is around 73% whereas the combination of tabular and graph data models results in around 82% accuracy. In fact, ensembling tabular and GNN models gives the best results. Details are available in the application on SageMaker JumpStart and also in the related GitHub repository: https://github.com/awslabs/sagemaker-graph-based-credit-scoring. The blog post (Das et al., 2022) for this solution also provides many additional details. Constructing corporate networks in this manner using text and combining it with graph machine learning offers many other promising applications such as assessing supply chain risk, banking contagion risk (Das et al., 2021), industry risk analysis, systemic risk (Das, 2016), etc. Overall, this illustrates that using data in alternative data structures such as graphs can enhance machine learning models.

11 Explaining Decisions by Multimodal ML Models

Explaining the decisions of machine learning models in terms of feature importance is now well established. Techniques such as LIME (Local Interpretable Model-Agnostic Explanations) Ribeiro et al. (2016), SHAP (Shapley Additive Explanations) Lundberg and Lee (2017), integrated gradients Sundararajan et al. (2017), feature permutation Breiman (2001), have all been applied quite successfully to tabular data. Many of the techniques applied to text have been borrowed from the image classification field (Shrikumar et al., 2019; Ribeiro et al., 2016).

However, these techniques become more difficult to apply to other data modalities. Deep text classifiers pose interesting problems in explaining the decisions made by machine learning classifiers as the number of features is large. If word-level feature importance is the focus, then the number of features will be the size of the vocabulary used, which will be in the tens of thousands. Further, only highlighting the important words in the document that were critical in the decision made by a ML classifier does not promote human understanding of the decision well. In Zafar et al. (2021) the authors find that the best granularity for textual feature importance is at the sentence level, better than at the word level or paragraph level.

The presentation of explanations can also take many forms. Taking the credit rating application we have seen already in this article, the algorithm (an adapted version of Shapley values developed in the paper Zafar et al. (2021)) determines which sentences from the SEC filing are important in driving the model decision. A brief sketch of this approach is that the tokenized text $\mathbf{t} = [t_1, ..., t_T]$ can be partitioned into non-overlapping meta-tokens $\mathbf{m} = [m_1, ..., m_M]$, where each meta-token comprises contiguous tokens. There are two simple ways in which this meta-partitioning is exploited: (i)

Compute Shapley values $\phi(t_i)$ for each token t_i and then aggregate these values for each meta-token m_j. In the examples shown below, each meta-token is a sentence. Hence, attribution is achieved at the sentence level. This is the approach followed here in the examples here. (ii) Another approach is to form the Shapley coalitions at the meta-token level and Shapley values are computed by dropping entire meta-tokens at a time. An example of the top few sentences with corresponding Shapley values is shown in Figure 10.

	Sentences	Scores
0	MANAGEMENT'S DISCUSSION AND ANALYSIS OF FINANCIAL CONDITION AND RESULTS OF OPERATIONS Management's Discussion and Analysis is the Company's analysis of its financial performance and of significant trends that may affect future performance.	0.802230
1	Our success has enabled us to focus on growth in 2014, which we intend to deliver through investments in our legacy assets, continued success in our development drilling and exploration programs, continued ramp up in our unconventional plays and additional project startups, such as the recent startup at Siakap North-Petai and the anticipated startups in Canada, Malaysia and the United Kingdom in 2014.	0.751796
2	As a result, we expect to deliver 3 to 5 percent volume growth in 2014.	0.843867
3	As a result, the discount of WTI to Brent decreased 48 percent in the first quarter of 2014, compared with the first quarter of 2013.	0.715378
4	Cash flow from operations of $6.3 billion, including a $0.6 billion working capital benefit and a $1.3 billion FCCL distribution.	0.891076
5	At that time, our terminal regasification capacity will be reduced from 0.9 billion cubic feet per day to 0.4 billion cubic feet per day, until July 1, 2016, at which time it will be reduced to zero.	0.725555
6	Earnings for the first quarter of 2014 included gains of $6 million after-tax, compared with a $270 million after-tax benefit associated with asset dispositions in the first quarter of 2013.	1.057426
7	- Higher operating expenses. -	0.772479
8	Earnings in the first quarter of 2014 benefitted from lower production taxes, mainly as a result of higher 2014 capital spending, lower crude oil production volumes and lower crude oil prices.	0.992302
9	Lower 48 and Latin America As of March 31, 2014, Lower 48 and Latin America contributed 30 percent of our worldwide liquids production and 38 percent of our natural gas production.	0.791164
10	For information about guarantees, see Note 9-Guarantees, in the Notes to Consolidated Financial Statements, which is incorporated herein by reference.	0.700656
11	In February 2014, we repaid notes at maturity totaling $400 million.	1.182341
12	- Exploration and appraisal drilling in deepwater Gulf of Mexico.	0.691558
13	For other examples of legislation or precursors for possible regulation and factors on which the ultimate impact on our financial performance will depend, see the "Climate Change" section in Management's Discussion and Analysis of Financial Condition and Results of Operations on pages 65-66 of our 2013 Annual Report on Form 10-K. CAUTIONARY STATEMENT FOR THE PURPOSES OF THE "SAFE HARBOR" PROVISIONS OF THE PRIVATE SECURITIES LITIGATION REFORM ACT OF 1995 This report includes forward-looking statements within the meaning of Section 27A of the Securities Act of 1933 and Section 21E of the Securities Exchange Act of 1934.	1.015347

Figure 10: Examples of explanatory sentences from the text with the corresponding Shapley values. Features are assumed to be at the sentence granularity.

Another interesting visualization is to show the entire document with each sentence color-coded in green if it increases the credit rating and red if it reduces the credit rating. One can hover over the sentences to see the text, as shown In Figure 11.

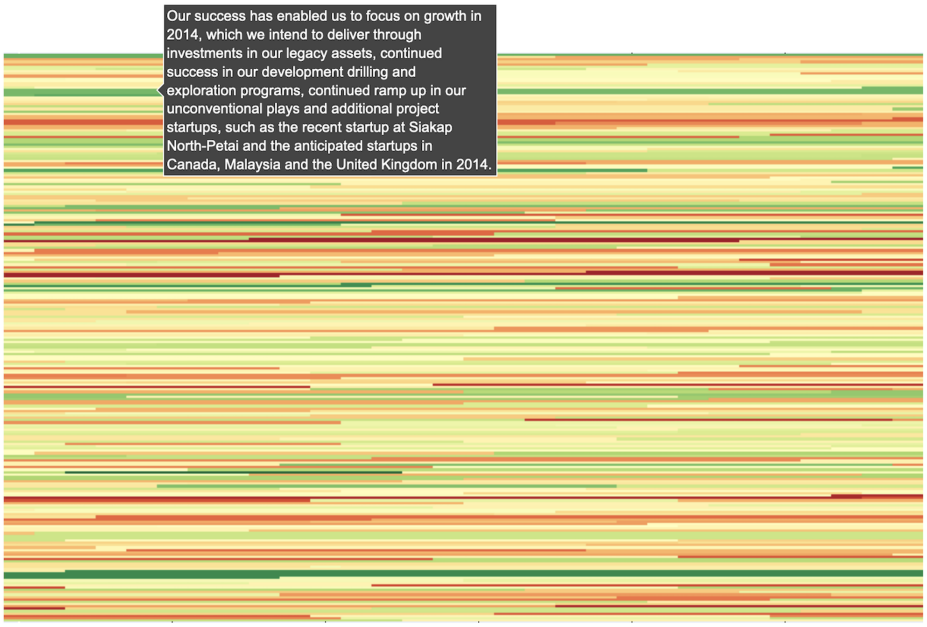

Figure 11: Visualizing sentence features in the corporate credit rating example using color coding and hover to see the features.

Experiments in the paper by Zafar et al. (2021) show that when human subjects were asked to classify companies by credit rating with the assistance of word-based explanations versus sentence-based explanations, they performed better with the sentence-based format. Therefore, this work suggests that token-based explainability methods may not be ideal, and modelers may wish to use higher granularity text explanations (sentences or paragraphs).

12 Domain Adaptation of LLMs for Generative AI Financial Applications

Large language models (LLMs), also known as "foundation models" are huge neural networks that can perform a variety of language tasks such as question and answering, text completion, zero-shot/few-shot classification, text summarization, code generation, named entity recognition (NER), grammar and spelling correction, sentence / sentiment classification, a chatbot and conversational AI, tweet generation, translation, intent classification, etc. These are all applications areas in finance.

LLMs are used to generate text but the downside is that they hallucinate and generate text that is totally made up, or reasonably accurate but still containing simple errors. In order to address the hallucination problem and also specialize the LLM, users can fine-tune the LLM with text of their own. This process is known as "domain adaptation" of the LLM. Domain adaptation results in much more accurate results for use cases and applications that are related to the documents used for fine-tuning.

We may use the text from SEC filings (extracted using the APIs discussed in Section 3) for domain-adapted fine-tuning. SageMaker JumpStart has solutions that implement fine-tuning of LLMs. See this blog post (Huang et al., 2023) where an example is provided end to end. For example, as shown in the blog post, when the GPT-J-6B LLM (Wang and Komatsuzaki, 2021) is used, there is a marked difference in the quality of responses before and after fine-tuning the LLM. Before fine-tuning, the answer to the following question – "What drives sales growth at Amazon?" is "Amazon is the world's largest online retailer. It is also the world's largest online marketplace. It is also the world". With fine tuning, the response gets better: "Sales growth at Amazon is driven primarily by increased customer usage, including increased selection, lower prices, and increased convenience, and increased sales by other sellers on our websites."

The substantive improvement in fact comes from what we might call "few-shot fine-tuning", i.e., fine-tuning using a small number of documents. In the example referenced above, fine-tuning is undertaken using the 10-Ks for two recent years only for a single ticker (AMZN in this case). In this way, the GPT-J-6B model is specialized to context related to Amazon's financials. Here is another example from the same fine-tuned model for a text completion task (different from the Q&A task presented earlier). In this example, we ask the LLM to complete a sentence, and use the prompt "This year we particularly focus on ..." – we see that without fine-tuning the LLM returns a repetitive incorrect answer, whereas with fine-tuning it delivers a comprehensive and correct response. The comparison is shown in Figure 12.

It is remarkable that simple few-shot fine-tuning can markedly improve performance. The total time for fine-tuning ranges from 15-30 minutes, which is parsimonious. In the financial setting it suggests that fine-tuning a separate model for each ticker may be predicated. SageMaker JumpStart also provides several other LLM model notebooks that may be used for text generation, see https://github.com/aws/amazon-sagemaker-examples/tree/main/introduction_to_amazon_algorithms/jumpstart-foundation-models.

13 Extensions

The SEC extractor is only the initial part of a TabText pipeline. The SDK runs on SageMaker and has the infrastructure to support many other features of interest for empirical and modeling work in finance using multimodal data. These additional features cover the gamut of data engineering to machine learning and finally, model interpretability for models on multimodal data. The following topics and extensions are not discussed in the article but are summarized here for consideration as additional features, applications, and use cases for multimodal ML.

1. Additional SEC forms. Whereas the initial SEC extractor in Section 3 contains the more widely downloaded forms, additional forms may be extracted. The shortlist includes (a) S-3

Figure 12: Comparison of the completion (first column) response from the model without fine-tuning (second column) and with fine-tuning using SEC 10-K filings (third column).

This year we particularly focus on	This year we particularly focus on: The role of the health care system in the prevention of obesity and diabetes The role of the health care system in the prevention of obesity and diabetes The role of the health care system in the prevention of obesity and diabetes The role of the health care system in the prevention of obesity and diabetes The role of the health care system in the prevention of obesity and diabetes The role of the health care system in the prevention of obesity and diabetes The role of the health care system in the prevention of obesity and diabetes	This year we particularly focus on: -Continued international expansion, including the development of newbusinesses in existing international market segments, as well as thedevelopment of new international market segments; -Expanding our customer base, including increasing sales to existingcustomers and attracting new customers; -Developing new business opportunities and entering into commercialagreements, including through licensing and other arrangements, with othercompanies, or through the acquisition of other companies; -Developing technology that provides for enhanced customer service andefficiently delivers our products and services; -Developing new product and service offerings and incorporating these intoexisting and new product and service offerings; -Developing efficient and effective systems for supporting our business;and -Other business and strategic initiatives. We believe that offering low prices to our customers is fundamental to ourfuture success. One way we offer lower prices is through free-shipping offersthat result in a net cost to us in delivering products, and through membershipin Amazon Prime, which provides free-shipping on millions of items andaccess to movies and other content. We also offer other promotions that enableus to turn a lower net cost of sales.We have organized our operations into two principal segments: North Americaand International. See Item 8 of Part II, ""Financial Statements andSupplementary Data-Note 12-Segment Information."" See Item 7 of Part II,""Management's Discussion and Analysis of Financial Condition and Results ofOperations-Results of Operations-Supplemental Information"" for supplementalinformation about our net sales.ConsumersWe serve consumers through our retail websites and physical stores and focuson selection, price, and convenience.

filings (simplified filing for securities registration), (b) 13D (filed when an entity acquires more than 5% stake in any class of company shares), (c) DEC 14A (Proxy statement), (d) N-1A filings (Registration statement), and (e) Form 435 (Mutual Fund Prospectus). These are all valuable sources of information for financial decisions that we did not discuss in this article.

2. Sectioning. Within these forms and the ones currently available in the SageMaker JumpStart system (10-K/Q, 8-K, 497, 497K, S-3 ASR), there may be sections that need to be parsed out. Whereas the plain text of the extraction enables easy parsing out of specific sections with some additional code, we will prioritize some of the more salient ones of interest. For example, the MD&A section in the 10-K/Q filings has already been added as a separate column of TabText, described in Section 3.3. For 10-Ks, 10-Qs, and 8-Ks, SageMaker JumpStart also provides a workflow to break out the documents by section, as presented in this blog: https://aws.amazon.com/blogs/machine-learning/create-a-dashboard-with-sec-text-for-financial-nlp-in-amazon-sagemaker-jumpstart/.

3. Searching. Adding an elastic search layer to a collection of indexed SEC filings in `s3` cache enables gathering a subset of documents that pertain to a specific area of interest or a concept, e.g., green energy. Portfolio managers may be interested in finding and summarizing the

MD&A sections of firms that mention this concept. With the advent of large language models (LLMs) and vector databases, this has become a reachable objective in the near term.

4. Summarization. Since SEC filings are long text documents, summarization of these documents is an important aid to researchers and analysts. We discussed this in Section 4. There are two potential uses of summaries. (a) Submission of a document to a classifier may lead to shortlisting a subset of documents for further attention and a quick pass through the documents may be facilitated with summaries. (b) Summaries of very long documents may be useful for classification algorithms, in improving signal to noise and also in reductions of computation time.
5. Scoring. The vast majority of the academic literature in finance has been focused on scoring documents for positivity, negativity, litigiousness, etc., see an excellent recent review by Loughran and McDonald (2020). Other scores are undertaken for readability, sentiment, etc. Adding numeric scores to long text and summaries is the first step in developing the TabText dataframe. The SDK on SageMaker will be able to support this at scale. Scoring is done using word lists and simple algorithms. An example is shown in Section 7.2.
6. Lexicons. NLP scoring has been undertaken using word lists (lexicons) developed in Loughran and Mcdonald (2011), which are hand-curated and require manual effort. We extend these using an automatic lexicon generator using language models for concepts other than the limited number currently available. For example, fraud/negligence lexicons and ESG lexicons. The word lists used in Section 7.2 were generated using our lexicon generator. The algorithm for this has been published in Das et al. (2022).
7. Document analysis. Features may be generated from more than just lexicons. Cases of particular interest are features (additional columns of TabText) that comprise (a) key sentences for a concept and surrounding sentences, (b) percentage similarity across documents or to a baseline document, (c) differences in text between quarterly filings for the same ticker or between tickers, (d) removal of boiler plate text, etc. Again, these use cases may be facilitated with LLMs.
8. Machine learning. It is possible to extend tabular ML models to multimodal data using TabText. In order to fit ML models the TabText dataframe can be extended to representations of text and tabular data using an array of approaches. This offers a rich menu of representations and machine learning models. Combined representations using standard approaches such as count vectorizers, TF-IDF vectors, transformer based representations (BERT, RoBERTa, etc.) combined with tabular data are then fed into ML pipelines. Section 7 above shows various examples of ML applications on TabText. These applications will soon experience interactions with multimodal LLMs (e.g., Huang et al. (2023)).
9. Long Documents. BERT models use relatively small documents comprising a small number of tokens. Regulatory filings are huge documents, well exceeding the maximum sequence length for tokens in BERT models, and needs an adaptation of classic BERT models. This is an interesting area of research fostering the use of SEC data. Long document classification has been attracting its own research interest in the legal area (Lee and Hsiang, 2019; Wan et al., 2019), academia (Liu et al., 2018), all of which have a rich supply of such text. Pappagari et al. (2019) find that ASR transcripts of call center conversations provides another source of study, in which individual transcripts can become exceedingly long as a conversation prolongs. A recent attempt at long document classification is the Big Bird paper, see Zaheer et al. (2020), and more recently Ding et al. (2020). With LLMs now being able to ingest much longer token sequences, and undertake zero-shot and few-shot classification tasks, the maximum sequence length constraint on transformer models is being relaxed and will open up much easier long document analysis (Chung et al., 2025).
10. Pre-training and fine-tuning financial language models, which was discussed in Section 8. One use of the SEC filings is in pre-training language models. Using the SEC 10-K and 10-Q filings for a large number of tickers combined with other common text will support pre-training finance-tilted language models. Filings have been shown to improve the informativeness of markets, see Gao and Huang (2020). To train these models with more than 100 million parameters, a number of GPUs are required which may be enabled on SageMaker. Ideally, p3dn.24xlarge instances, where each instance has 92 vCPUs and 8 Nvidia V100 Tensor core GPUs, with a distributed deep learning training framework to run the pre-training job using all GPUs. New chips keep emerging and will make multimodal

model building even easier as time passes. Pre-training models on proprietary text offers the possibility of a wide range of financial language models. Fine-tuning LLMs for finance is now the latest frontier (Wu et al., 2023) and such usage is widespread.

11. Explaining financial NLP models. The goal is to provide interpretability for text classification/regression models at various levels of granularity: individual words, phrases, and sentences. Further develop these methods to provide interpretability for text+tabular data. Interpretability for text data is an under-explored domain and poses a unique set of challenges. Well-adopted interpretability techniques in tabular domain (e.g., as LIME and SHAP) do not provide satisfactory performance on text data as they are designed to provide interpretability at the level of individual features. We have seen in Section 11 how sentence-based explanations offer better interpretability of ML decisions on text. We have also seen in Section 9 how feature importance across text and tabular data is achieved. Going forward, expect to see generative AI used to write out natural language explanations that are much more interpretable than ranked Shapley values and feature importance.

The potential for building end to end multimodal data pipelines in finance is huge. Getting and transforming SEC data for multimodal machine learning is just the starting point. Expect to see new architectures across more modalities integrated with LLMs, bringing machine financial decisions closer to human decision making.

References

Altman, E. I. (1968). Financial Ratios, Discriminant Analysis and the Prediction of Corporate Bankruptcy. *The Journal of Finance 23*(4), 589–609.

Antweiler, W. and M. Z. Frank (2004). Is All That Talk Just Noise? The Information Content of Internet Stock Message Boards. *The Journal of Finance 59*(3), 1259–1294.

Araci, D. (2019, August). FinBERT: Financial Sentiment Analysis with Pre-trained Language Models. *arXiv:1908.10063 [cs]*. arXiv: 1908.10063.

Autor, D., D. Cho, L. D. Crane, M. Goldar, B. Lutz, J. Montes, W. B. Peterman, D. Ratner, D. Villar, and A. Yildirmaz (2022, May). The $800 Billion Paycheck Protection Program: Where Did the Money Go and Why Did It Go There? *Journal of Economic Perspectives 36*(2), 55–80.

Bach, M. P., Z. Krstic, S. Seljan, and L. Turulja (2019). Text Mining for Big Data Analysis in Financial Sector: A Literature Review. *Sustainability 11*(5), 1–27.

Balyuk, T., N. Prabhala, and M. Puri (2020, November). Indirect Costs of Government Aid and Intermediary Supply Effects: Lessons from the Paycheck Protection Program. *NBER Working Paper No. w28114*.

Bodnaruk, A., T. Loughran, and B. McDonald (2015, August). Using 10-K Text to Gauge Financial Constraints. *Journal of Financial and Quantitative Analysis 50*(4), 623–646.

Bonsall, S. B., A. J. Leone, B. P. Miller, and K. Rennekamp (2017, April). A plain English measure of financial reporting readability. *Journal of Accounting and Economics 63*(2), 329–357.

Breiman, L. (2001, October). Random Forests. *Machine Learning 45*(1), 5–32.

Brown, S. and J. W. Tucker (2011). Large-Sample Evidence on Firms' Year-over-Year MD&A Modifications. *Journal of Accounting Research 49*(2), 309–346.

Brown, T. B., B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei (2020, July). Language Models are Few-Shot Learners. Number: arXiv:2005.14165 arXiv:2005.14165 [cs].

Bushee, B. J., I. D. Gow, and D. J. Taylor (2018). Linguistic Complexity in Firm Disclosures: Obfuscation or Information? *Journal of Accounting Research 56*(1), 85–121. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/1475-679X.12179.

Calomiris, C. W., H. Mamaysky, and R. Yang (2020, June). Measuring the Cost of Regulation: A Text-Based Approach. SSRN Scholarly Paper ID 3550922, Social Science Research Network, Rochester, NY.

Chebonenko, T., L. Gu, and D. Muravyev (2018, March). Text Sentiment's Ability to Capture Information: Evidence from Earnings Calls. SSRN Scholarly Paper ID 2352524, Social Science Research Network, Rochester, NY.

Chen, T., M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang (2015, December). MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems. arXiv:1512.01274 [cs].

Child, R., S. Gray, A. Radford, and I. Sutskever (2019, April). Generating Long Sequences with Sparse Transformers. *arXiv:1904.10509 [cs, stat]*. arXiv: 1904.10509.

Chung, Y., G. T. Kakkar, Y. Gan, B. Milne, and F. Ozcan (2025, April). Is Long Context All You Need? Leveraging LLM's Extended Context for NL2SQL. *Proceedings of the VLDB Endowment 18*(8), 2735–2747. arXiv:2501.12372 [cs].

Cohen, L., C. Malloy, and Q. Nguyen (2020). Lazy Prices. *The Journal of Finance 75*(3), 1371–1415. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/jofi.12885.

Cong, L. W., T. Liang, B. Yang, and X. Zhang (2019, November). Analyzing Textual Information at Scale. SSRN Scholarly Paper ID 3449822, Social Science Research Network, Rochester, NY.

Cong, L. W., T. Liang, and X. Zhang (2019, September). Textual Factors: A Scalable, Interpretable, and Data-driven Approach to Analyzing Unstructured Information. SSRN Scholarly Paper ID 3307057, Social Science Research Network, Rochester, NY.

Costa, L. d. F. (2021, November). Further Generalizations of the Jaccard Index. arXiv:2110.09619 [cs].

Coval, J. D. and T. Shumway (2001). Is Sound Just Noise? *The Journal of Finance 56*(5), 1887–1910.

Das, S., S. Adeshina, P. Yang, and X. Huang (2022, May). Build a corporate credit ratings classifier using graph machine learning in Amazon SageMaker JumpStart | Artificial Intelligence. Section: Amazon SageMaker: https://aws.amazon.com/blogs/machine-learning/build-a-corporate-credit-ratings-classifier-using-graph-machine-learning-in-amazon-sagemaker-jumpstart/.

Das, S., C. Goggins, J. He, G. Karypis, S. Krishnamurthy, M. Mahajan, N. Prabhala, D. Slack, R. van Dusen, S. Yue, S. Zha, and S. Zheng (2021, July). Context, Language Modeling, and Multimodal Data in Finance. *The Journal of Financial Data Science 3*(3), 52–66.

Das, S., X. Huang, S. Adeshina, P. Yang, and L. Bachega (2023, October). Credit Risk Modeling with Graph Machine Learning. *INFORMS Journal on Data Science 2*(2), 197–217.

Das, S. R. (2014). Text and Context: Language Analytics in Finance. *Foundations and Trends® in Finance 8*(3), 145–261.

Das, S. R. (2016, March). Matrix Metrics: Network-Based Systemic Risk Scoring. *The Journal of Alternative Investments 18*(4), 33–51.

Das, S. R. and M. Y. Chen (2007). Yahoo! For Amazon: Sentiment Extraction from Small Talk on the Web. *Management Science 53*(9), 1375–1388.

Das, S. R., M. Donini, M. B. Zafar, J. He, and K. Kenthapadi (2022, November). FinLex: An effective use of word embeddings for financial lexicon generation. *The Journal of Finance and Data Science 8*, 1–11.

Das, S. R., K. J. Mitchener, and A. Vossmeyer (2021). Bank Regulation, Network Topology, and Systemic Risk: Evidence from the Great Depression. *Journal of Money, Credit and Banking n/a*(n/a). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/jmcb.12871.

Desola, V., K. Hanna, and P. Nonis (2020). FinBERT: pre-trained model on SEC filings for financial natural language tasks. *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20) Special Track on AI in FinTech*.

Devlin, J., M.-W. Chang, K. Lee, and K. Toutanova (2019, May). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805 [cs]*. arXiv: 1810.04805 version: 2.

Ding, M., C. Zhou, H. Yang, and J. Tang (2020). CogLTX: Applying BERT to Long Texts. *Advances in Neural Information Processing Systems 33*.

Erickson, N., J. Mueller, A. Shirkov, H. Zhang, P. Larroy, M. Li, and A. Smola (2020, March). AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data. arXiv:2003.06505 [stat].

Ertugrul, M., J. Lei, J. Qiu, and C. Wan (2017, April). Annual Report Readability, Tone Ambiguity, and the Cost of Borrowing. *Journal of Financial and Quantitative Analysis 52*(2), 811–836.

Gao, M. and J. Huang (2020, April). Informing the Market: The Effect of Modern Information Technologies on Information Production. *The Review of Financial Studies 33*(4), 1367–1411.

Gentzkow, M., B. Kelly, and M. Taddy (2019, September). Text as Data. *Journal of Economic Literature 57*(3), 535–574.

Grover, A. and J. Leskovec (2016, August). node2vec: Scalable Feature Learning for Networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, New York, NY, USA, pp. 855–864. Association for Computing Machinery.

Hafez, P., M. Kangrga, J. Guerrero-Colon, F. Gomez, and R. Matas (2020). Capturing Alpha From Your Own Digital Content. Library Catalog: www.ravenpack.com.

Hafez, P., R. Matas, F. Gomez, M. Kangrga, B. Skorodumov, and A. Liu (2020). RavenPack News Sentiment Data Outperforms During Coronavirus Crisis. Library Catalog: www.ravenpack.com.

Hamilton, W. L. (2020, September). *Graph Representation Learning*. San Rafael, California: Morgan & Claypool.

Hoberg, G. and G. Phillips (2016, October). Text-Based Network Industries and Endogenous Product Differentiation. *Journal of Political Economy 124*(5), 1423–1465.

Huang, S., L. Dong, W. Wang, Y. Hao, S. Singhal, S. Ma, T. Lv, L. Cui, O. K. Mohammed, Q. Liu, K. Aggarwal, Z. Chi, J. Bjorck, V. Chaudhary, S. Som, X. Song, and F. Wei (2023, February). Language Is Not All You Need: Aligning Perception with Language Models. arXiv:2302.14045 [cs].

Huang, X., A. Khetan, M. Cvitkovic, and Z. Karnin (2020, December). TabTransformer: Tabular Data Modeling Using Contextual Embeddings. arXiv:2012.06678 [cs].

Huang, X., A. K. Lokanatha, M. Karp, and S. Das (2023, April). Domain-adaptation Fine-tuning of Foundation Models in Amazon SageMaker JumpStart on Financial data | Artificial Intelligence. Section: Amazon SageMaker: https://aws.amazon.com/blogs/machine-learning/domain-adaptation-fine-tuning-of-foundation-models-in-amazon-sagemaker-jumpstart-on-financial-data/.

Jaccard, P. (1901). Distribution de la flore alpine dans le Bassin des Dranses et dans quelques régions voisines. *Bulletin de la Société Vaudoise Des Sciences Naturelles 37*, 241–272. Medium: text/html,application/pdf,text/html.

Jegadeesh, N. and D. Wu (2013, December). Word power: A new approach for content analysis. *Journal of Financial Economics 110*(3), 712–729.

Joulin, A., E. Grave, P. Bojanowski, M. Douze, H. Jégou, and T. Mikolov (2016, December). FastText.zip: Compressing text classification models. *arXiv:1612.03651 [cs]*. arXiv: 1612.03651.

Kearney, C. and S. Liu (2014, May). Textual sentiment in finance: A survey of methods and models. *International Review of Financial Analysis 33*, 171–185.

Lee, J.-S. and J. Hsiang (2019, June). PatentBERT: Patent Classification with Fine-Tuning a pre-trained BERT Model. *arXiv:1906.02124 [cs, stat]*. arXiv: 1906.02124.

Levy, A. and K. Kumar (2021, February). An Overview of Modeling Credit Portfolios. Technical report, Moody's Analytics.

Li, F. (2010a). The Information Content of Forward-Looking Statements in Corporate Filings—A Naïve Bayesian Machine Learning Approach. *Journal of Accounting Research 48*(5), 1049–1102. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1475-679X.2010.00382.x.

Li, F. (2010b). Textual analysis of corporate disclosures : a survey of the literature. *Journal of accounting literature 29*, 143–165.

Li, J. and X. Zhao (2014). Complexity and Information Content of Financial Disclosures: Evidence from Evolution of Uncertainty Following 10-K Filings. *SSRN Electronic Journal*.

Liu, L., K. Liu, Z. Cong, J. Zhao, Y. Ji, and J. He (2018, August). Long Length Document Classification by Local Convolutional Feature Aggregation. *Algorithms 11*(8), 109. Number: 8.

Liu, Y., M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov (2019, July). RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv:1907.11692 [cs]*. arXiv: 1907.11692.

Loughran, T. and B. Mcdonald (2011). When Is a Liability Not a Liability? Textual Analysis, Dictionaries, and 10-Ks. *The Journal of Finance 66*(1), 35–65. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1540-6261.2010.01625.x.

Loughran, T. and B. McDonald (2013). IPO first-day returns, offer price revisions, volatility, and form S-1 language. *Journal of Financial Economics 109*(2), 307–326.

Loughran, T. and B. McDonald (2014). Measuring Readability in Financial Disclosures. *The Journal of Finance 69*(4), 1643–1671.

Loughran, T. and B. McDonald (2015, January). The Use of Word Lists in Textual Analysis. *Journal of Behavioral Finance 16*(1), 1–11. _eprint: https://doi.org/10.1080/15427560.2015.1000335.

Loughran, T. and B. Mcdonald (2016). Textual Analysis in Accounting and Finance: A Survey. *Journal of Accounting Research 54*(4), 1187–1230. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/1475-679X.12123.

Loughran, T. and B. McDonald (2020, June). Textual Analysis in Finance. SSRN Scholarly Paper ID 3470272, Social Science Research Network, Rochester, NY.

Loughran, T., B. McDonald, and H. Yun (2009, May). A Wolf in Sheep's Clothing: The Use of Ethics-Related Terms in 10-K Reports. *Journal of Business Ethics 89*(1), 39–49.

Lundberg, S. M. and S.-I. Lee (2017, December). A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, Red Hook, NY, USA, pp. 4768–4777. Curran Associates Inc.

Mikolov, T., K. Chen, G. Corrado, and J. Dean (2013, September). Efficient Estimation of Word Representations in Vector Space. *arXiv:1301.3781 [cs]*. arXiv: 1301.3781.

Mikolov, T., E. Grave, P. Bojanowski, C. Puhrsch, and A. Joulin (2017, December). Advances in Pre-Training Distributed Word Representations. *arXiv:1712.09405 [cs]*. arXiv: 1712.09405.

Molnar, C., G. König, B. Bischl, and G. Casalicchio (2020, June). Model-agnostic Feature Importance and Effects with Dependent Features – A Conditional Subgroup Approach. *arXiv:2006.04628 [cs, stat]*. arXiv: 2006.04628.

Mundhenk, T. N., B. Y. Chen, and G. Friedland (2020, March). Efficient Saliency Maps for Explainable AI. *arXiv:1911.11293 [cs]*. arXiv: 1911.11293.

Nguyen, C. V., S. R. Das, J. He, S. Yue, V. Hanumaiah, X. Ragot, and L. Zhang (2021, July). Multimodal Machine Learning for Credit Modeling. In *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*, pp. 1754–1759.

OpenAI (2023, March). GPT-4 Technical Report. arXiv:2303.08774 [cs].

Pappagari, R., P. Żelasko, J. Villalba, Y. Carmiel, and N. Dehak (2019, October). Hierarchical Transformers for Long Document Classification. *arXiv:1910.10781 [cs, stat]*. arXiv: 1910.10781.

Press, O., N. A. Smith, and M. Lewis (2022, April). Train Short, Test Long: Attention with Linear Biases Enables Input Length Extrapolation. arXiv:2108.12409 [cs].

Price, S. M., J. S. Doran, D. R. Peterson, and B. A. Bliss (2012, April). Earnings conference calls and stock returns: The incremental informativeness of textual tone. *Journal of Banking & Finance 36*(4), 992–1011.

Ribeiro, M. T., S. Singh, and C. Guestrin (2016, August). "Why Should I Trust You?": Explaining the Predictions of Any Classifier. *arXiv:1602.04938 [cs, stat]*. arXiv: 1602.04938.

Routledge, B. R. (2019). Machine learning and asset allocation. *Financial Management 48*(4), 1069–1094. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/fima.12303.

Salton, G. and C. Buckley (1988, January). Term-weighting approaches in automatic text retrieval. *Information Processing & Management 24*(5), 513–523.

Shapley, L. S. (1952, January). A Value for n-Person Games. *RAND Corporation*.

Shi, X., J. Mueller, N. Erickson, M. Li, and A. J. Smola (2021, November). Benchmarking Multimodal AutoML for Tabular Data with Text Fields. arXiv:2111.02705 [cs].

Shrikumar, A., P. Greenside, and A. Kundaje (2019, October). Learning Important Features Through Propagating Activation Differences. arXiv:1704.02685 [cs].

Sundararajan, M., A. Taly, and Q. Yan (2017, June). Axiomatic Attribution for Deep Networks. *arXiv:1703.01365 [cs]*. arXiv: 1703.01365.

Tang, Z., H. Fang, S. Zhou, T. Yang, Z. Zhong, C. Hu, K. Kirchhoff, and G. Karypis (2024, October). AutoGluon-Multimodal (AutoMM): Supercharging Multimodal AutoML with Foundation Models. In *Proceedings of the Third International Conference on Automated Machine Learning*, pp. 15/1–35. PMLR.

Tetlock, P. C., M. Saar-Tsechansky, and S. Macskassy (2008). More Than Words: Quantifying Language to Measure Firms' Fundamentals. *The Journal of Finance 63*(3), 1437–1467. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1540-6261.2008.01362.x.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin (2017, December). Attention Is All You Need. *arXiv:1706.03762 [cs]*. arXiv: 1706.03762.

Wallace, E., J. Tuyls, J. Wang, S. Subramanian, M. Gardner, and S. Singh (2019, September). AllenNLP Interpret: A Framework for Explaining Predictions of NLP Models. *arXiv:1909.09251 [cs]*. arXiv: 1909.09251.

Wan, L., G. Papageorgiou, M. Seddon, and M. Bernardoni (2019, December). Long-length Legal Document Classification. *arXiv:1912.06905 [cs]*. arXiv: 1912.06905.

Wang, B. and A. Komatsuzaki (2021, May). GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://huggingface.co/EleutherAI/gpt-j-6b.

Wu, S., O. Irsoy, S. Lu, V. Dabravolski, M. Dredze, S. Gehrmann, P. Kambadur, D. Rosenberg, and G. Mann (2023, March). BloombergGPT: A Large Language Model for Finance. arXiv:2303.17564 [cs, q-fin].

Yang, Y., M. C. S. UY, and A. Huang (2020, July). FinBERT: A Pretrained Language Model for Financial Communications. *arXiv:2006.08097 [cs]*. arXiv: 2006.08097.

Zafar, M. B., P. Schmidt, M. Donini, C. Archambeau, F. Biessmann, S. R. Das, and K. Kenthapadi (2021, December). More Than Words: Towards Better Quality Interpretations of Text Classifiers. arXiv:2112.12444 [cs].

Zaheer, M., G. Guruganesh, A. Dubey, J. Ainslie, C. Alberti, S. Ontanon, P. Pham, A. Ravula, Q. Wang, L. Yang, and A. Ahmed (2020, July). Big Bird: Transformers for Longer Sequences. *arXiv:2007.14062 [cs, stat]*. arXiv: 2007.14062.

Zheng, S., H. Lin, S. Zha, and M. Li (2020, September). Accelerated Large Batch Optimization of BERT Pretraining in 54 minutes. *arXiv:2006.13484 [cs, stat]*. arXiv: 2006.13484.