10-K Filing Embeddings

EC 2355: Unleashing Novel Data at Scale

Hamzeh Hamdan

May 1, 2024

Abstract

This project serves two purposes: (1) to accurately embed 10-K filing text, and (2) to make the code accessible as a tool for research. Sample uses include clustering companies, analyzing trends within subsections of 10-K filings (e.g. business description, risk factors), and creating segments within markets or industries. Using text data from the SEC's EDGAR API and the API from sec-api.io, I tokenized and embedded data using SEC BERT Shape, a variant of BERT trained on 260,773 10-K filings from 1993 to 2019.[8, 7, 5] These embeddings can then be clustered and/or visualized through dimensionality reduction techniques. Results were validated through cosine similarity and proximity validation accuracy. Initial testing with 23 companies across 7 industries revealed promising results; of the 23 potential pairings of companies within the same industry, 11 were accurately classified as similar through a cosine similarity above 0.4. Of the remaining 230 potential pairings across industries, only 3 were wrongly classified as similar through a cosine similarity above 0.4. This yields a coverage of 48% and accuracy of 79%. Adjusting the cosine similarity threshold can inversely affect coverage and directly affect accuracy.

Note: The code used in this project is in a public repository on my GitHub.

Contents

1 Introduction

While the capabilities of natural language processing are growing in their ability to model the complexities of verbal communication and text, the use of textual analysis in empirical finance is limited, and typically only occurs in studying the "sentiment" of a text. This does not utilize the rich, qualitative insights that are found in textual data, such as that found in the SEC 10-K filings. These filings, which publicly traded companies file annually, offer comprehensive details about a company's business model, risk factors, and other factors vital to their operation.

Recent advanced in natural language processing have shown great potential in extracting and utilizing the content of financial texts. For instance, Yang, Zhang, and Fan illustrate the use of text embeddings, using financial domain-specific lexicons to enhance stock volatility predictions.[10] Additionally, a study by Adosoglou et al. shows that neural network embeddings can capture semantic changes in 10-K filings, and can be used in a trading strategy that earns up to 10% in yearly risk-adjusted abnormal returns.[1] Lastly, Tingyue Gan at UC Berkeley found that GICS industry classifications can be improved using 10-K filings data using text classification and clustering.[4]

While there have been a very recent increase in investigations of financial text analyses, there remains a large barrier due to the challenges of extracting, processing, and analyzing 10-K filings. This project addresses these challenges by creating an open-source Python class that extracts the data from the SEC's API, creates embeddings, clusters, cosine similarity heatmaps, and proximity validation accuracy reports. Drawing inspiration from the aforementioned works, this project develops a methodology that employs SEC BERT Shape, a variant of BERT pretrained on 260,773 10-K filings from 1993 to 2019.[5] The primary aim of this project is to provide a tool that accurately embeds 10-K filings text for a set of companies. Users would simply need a list of company tickers, a list of the sections of the 10-K filing they'd like to include, and their sec-api.io API key to run the embedding and clustering algorithms. As for validation, cosine similarity heatmaps are also available and, if labels are provided, proximity validation accuracy.

2 Background

2.1 Significance of 10-K Filings

Every year, public companies submit their annual 10-K filing report to the SEC. This report includes a comprehensive overview of the company's operations and financial health, and provides insight that exceed, in depth, traditional numerical metrics. However, these filings are often 80-150 pages long, making them very inefficient and challenging to manually analyze.

A 10-K filing report typically includes information on the business description, risk factors, cybersecurity, properties, legal proceedings, management's discussion and analysis of financial condition and results of operation, quantitative and qualitative disclosures about market risk, executive officers and corporate governance, and executive compensation. As such, an accurate embedding algorithm for subsections of 10-K filings could allow for easier analysis of similarities among companies within any subset of these factors.

2.2 Challenges of Using 10-K Filings

Despite the richness of data provided in 10-K filings, their analysis has typically been constrained to qualitative metrics and basic textual analysis. The challenges of using 10-K filings can be divided into two sections: data retrieval and data comprehension.

While the SEC has publicized 10-K filings through the EDGAR Database and its corresponding API, the interface has a very large learning curve and is inaccessible. Often, it'll falsely note that

the user is surpassing the 10 requests / second threshold and will temporarily suspend users. Even more problematic, the lack of standardization amongst companies' reporting processes make the database useful for retrieving a company's 10-K filing, but poses significant challenges for text extraction. This can be solved using sec-api.io's API, which extracts the text from the URLs of 10-K filings found by the EDGAR Database.

The primary challenge in using textual data from the 10-K filings lies in the interpretation of nuanced information embedded in the text; the language used in the filings is often specialized and can vary between companies and industries.

3 Methods

3.1 Data Collection

The data collection process is split into two parts: the SEC's EDGAR API for retreiving 10-K filing links for each company and the sec-api.io API for retrieving the text by subsection.

Using the SEC's EDGAR API, we check to make sure that inputted tickers in the list match those in the EDGAR database, discarding those that do not. For the remaining tickers, we find their CIK numbers, their submission metadata with all the forms they have submitted, and the accession number of their latest 10-K filing data. We use this to form a unique link for each 10-K filing. Using the sec-api.io API, we can retrieve text for subsections of the 10-K filings using the links retrieved above. These are returned as a Pandas DataFrame and can be saved as CSV files.

3.2 Preprocessing, Tokenizing, and Embedding with SEC BERT Shape

When preprocessing the text retrieved by the API, we decode HTML entities, remove URLs, convert the text to lowercase, replace newline characters with space, and remove extra spaces from the text. As an additional requirement for SEC BERT Shape, numbers must also be reformatted to [X,XXX.XX] format, where the numbers are replaced by X and there are square brackets around the number. We do this by temporarily using Spacy tokenization, checking each token using a REGEX pattern, converting the tokens if they are detected, then joining the tokens into a processed text.

As for the tokenization, we use the AutoTokenizer from SEC BERT Shape. We note that SEC BERT Shape only has a 512 token window, so we chuck and tokenize the text under the following conditions: 1) each chuck must end on a full sentence, 2) each chuck must have an overlap of 2 sentences, if possible, to maintain context. The tokens are then combined into one vector. Lastly, we encode the tokens, as SEC BERT Shape handle numerical encoded embeddings, and not the text we've been dealing with.

We embed using SEC BERT Shape on the vector of tokens retrieved. This is returned as a Pandas DataFrame as well, where each subsection of the 10-K filing included by the user has its own tokenization and embedding; this is done so that users do not have to run the code repeatedly for similar efforts (i.e. you can load the tokens from a model that analyzed sections '1', '1A', and '2' to analyze sections '1' and '1A', though that would require some changes in the code by the user as the loading function is not supported in the code yet).

3.3 Clustering with K-Means and HBDSCAN

Clustering on the embeddings is supported using both KMeans and HDBSCAN. KMeans clustering is a popular centroid-based clustering algorithm that is widely used due to its simplicity, efficiency, and ease of implementation; however, it also assumes spherical clusters, which might not be a valid assumption in these embeddings. HDBSCAN is a density-based algorithm that is

flexible to irregular cluster geometries; however, it is very sensitive to its parameters and is harder to interpret. While the library does not currently support the elbow method or silhouette scores, the clustering data is provided in the returned Pandas DataFrame so these analyses can be made easily.

3.4 Embedding and Cluster Visualization with t-SNE and UMAP

Due to the large dimensionality of the embeddings, techniques like t-SNE and UMAP are very helpful in visualizing the embeddings and/or clusters. Users can choose which of the two, or both, dimensionality reduction techniques they'd like to include in the visualization, and can save the resulting image to a file path.

3.5 Validation Using Cosine Similarity Heatmaps and Proximity Validation Accuracy

In order to validate results, two appraoches were taken: cosine similarity heatmaps and proximity validation accuracy (if labels were provided).

Cosine similarity is a metric used to measure how similar two documents are regardless of their size; it measures the cosine of the angle between the embedding vectors of two companies. Taking on a support of -1 to 1, values closer to 1 have greater similarity. A heatmap of known companies' cosine similarity can be a great source of validation; a great test metric might be to select a few industries that are dissimilar, and a few companies that are similar within each industry. Running a cosine similarity heatmap analysis should reveal larger values within companies in the same industry. To help with visualization, the method also allows users to filter for values that are strictly above a threshold; we found 0.4 and 0.5 to be great places to start.

If ground labels, such as industry or sector classifications, are provided, proximity validation accuracy can also be used. Proximity validation accuracy evaluates embeddings by checking how well they match the expected cosine similarities according to their labels. The proximity threshold can be adjusted by the user. This is useful when users know which companies should be similar or different as it allows them to to measure how well the embeddings and clustering processes are performing relative to these expectations.

4 Results

4.1 Testing with 23 S&P 100 Companies

The pipeline was initially run on 23 companies from the S&P 100 using the business description and risk factors sections of their most recent 10-K filings; the sector breakdown was as follows: 3 Communication Services, 3 Energy, 4 Financials, 2 Consumer Discretionary, 5 Information Technology, 3 Health Care, and 3 Utilities. Note that Alphabet Inc. (GOOG) is noted as a Communication Services company but was grouped with Information Technology due to its similarities to other companies (Apple, Microsoft).[2, 6]

When testing with K-Means clustering, I used 9 clusters. An intuitive choice might have been 7, as that is the number of GICS sectors represented in the sample of companies above; however, those results did not yield as much interpretability in the clusters.

Looking at the embeddings in Figures 1 and 2, we find that the Energy companies (CVX, XDM, COP) make up a cluster. Similarly, we see 3 of the 5 Information Technology companies (MSFT, NVDA, GOOG) in the same cluster; All of the Financials were also in the same cluster (MS, BAC, WFC, GS). Some counter-intuitive results included Apple (AAPL) and Eli Lilly and Co (LLY)

being in the same cluster, as well as Pfizer (PFE), Starbucks (SBUX), and Disney (DIS). We also note that UMAP typically visualized clustered data closer to each other.

Figure 1: t-SNE Visualization of Embeddings.

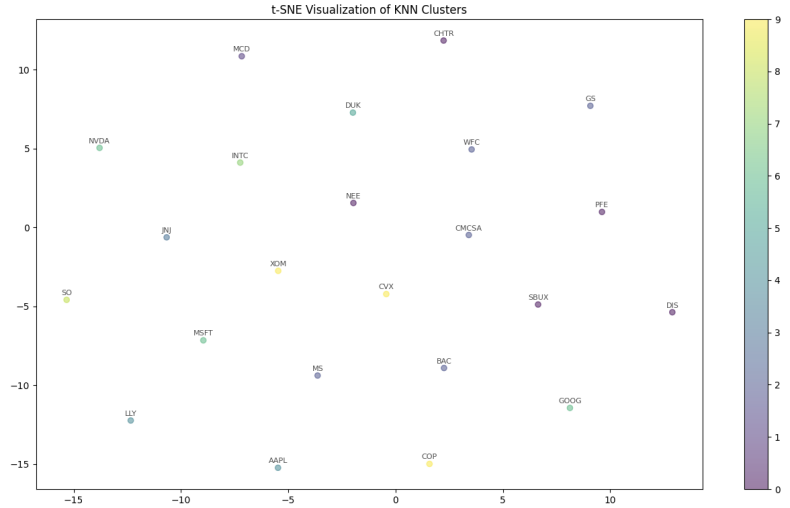
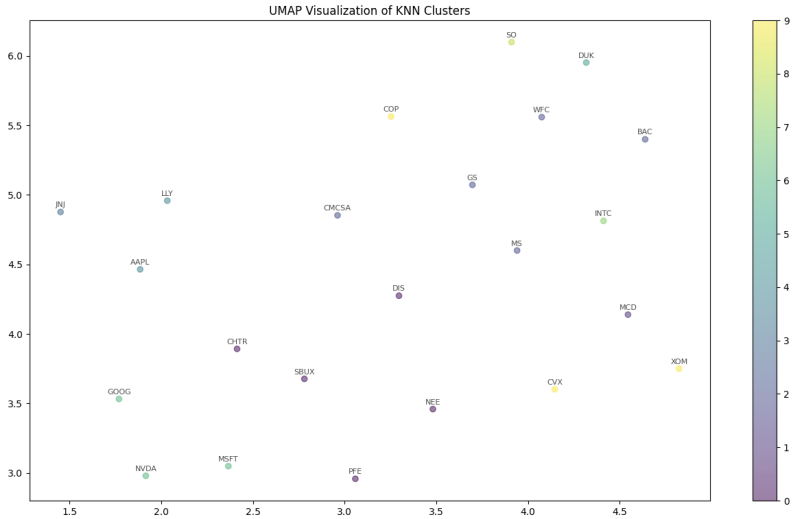
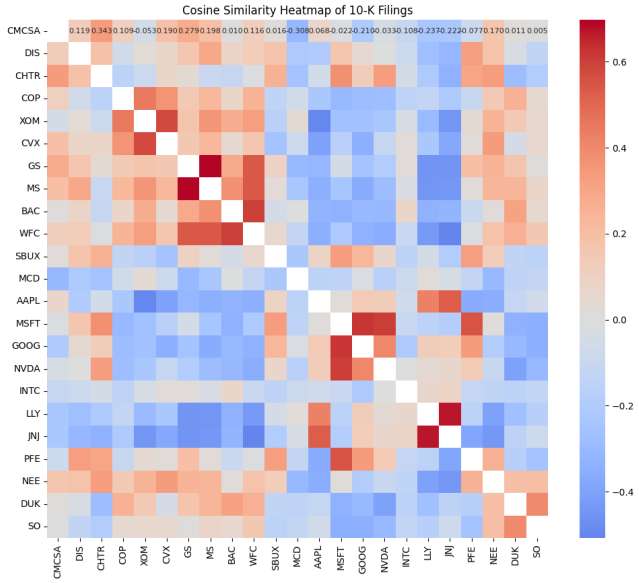


Figure 2: UMAP Visualization of Embeddings.



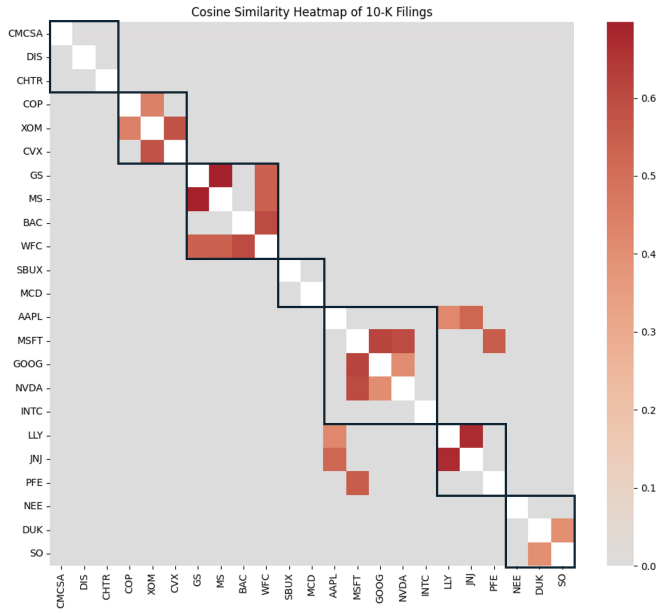
Using the validation techniques above, we did observe some promising results. As seen in Figure 3, the cosine similarity between companies within the same industry for embeddings based on business description and risk factors were typically higher than those between companies from different sectors. Visually, this corresponds to the darkest reds tend to be along the diagonal; more concretely, fixing a set of rows or columns that align with a GICS sector, the darkest reds should be along the diagonal.

Figure 3: Cosine Similarity Heatmap.



Note that Figure 3 has a lot of noise with cosine similarities that are insignificant. Filtering for cosine similarity values above 0.4, we result with Figure 4; note that the black square outlines along the diagonal represent the cosine similarities of companies within the same GICS sector.

Figure 4: Cosine Similarity Heatmap, Score > 0.4.



Of the 23 potential pairings of companies within the same industry, 11 were accurately classified as similar through a cosine similarity above 0.4. Of the remaining 230 potential pairings across industries, only 3 were wrongly classified as similar through a cosine similarity above 0.4. This yields a coverage of 48% and accuracy of 79%.

Experimentation was done with a threshold of 0.5, which decreased coverage to 35% and increased accuracy to 80%. Note that "accuracy" here relies on the assumption that companies within the same industry share a nearly exclusive similarity in business description and risk factors.

Running the proximity validation accuracy on different threshold values, we found a decreasing increase in accuracy as threshold increased, with an accuracy of 0.85 with a threshold of 0.4.

4.2 Embedding S&P 100

The embeddings and K-Means clustering of the S&P 100 with 10 clusters (the number of GICS sectors represented by the S&P 100) are shown in Figures 5 and 6.[2, 6]

Figure 5: t-SNE Visualization of Embeddings.

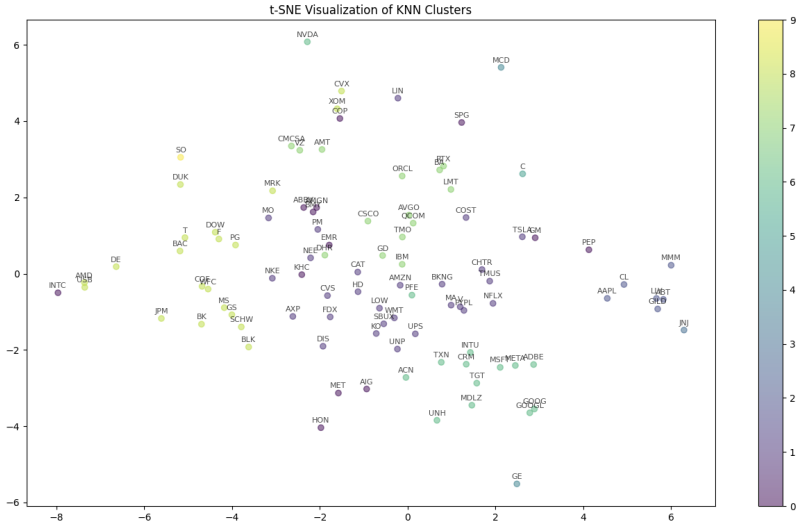
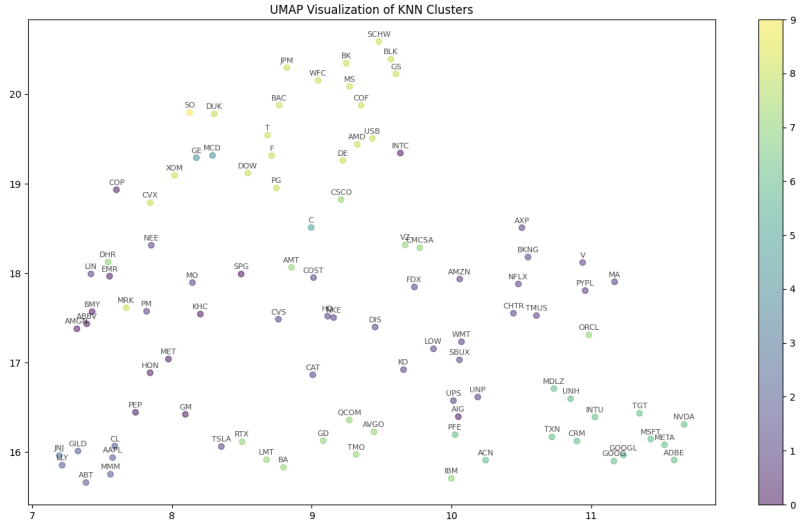


Figure 6: UMAP Visualization of Embeddings.



I would like to note here that testing has only occurred on the S&P 100 because of my familiarity with the companies. Diversified testing of smaller companies might be helpful as those companies might have different structures and information in their 10-K filings.

5 Discussion

5.1 More on Validation Efforts

While I have explored validation using cosine similarity heatmaps and proximity validation accuracy, there are some limitations to these techniques and further validation efforts would be helpful.

For the cosine similarity heatmaps, the intuitive visualization and direct quantitative comparisons allowed for a deeper understanding of the data structure. However, cosine similarity is sensitive to the scale and distribution of the data, and can break down in high-dimensional spaces; additionally, while the visualization is intuitive, interpreting a specific score between two companies is not feasible.

As for proximity validation accuracy, it provides a measure of how well the clusters formed align with pre-defined labels. Their effectiveness, though, is contingent on the accuracy of the labels.

Some potential methods of further validation, it might be beneficial to spend more time analyzing the embeddings through t-SNE and UMAP and trying to find intuitive explanations for the clusters. Early analyses showed no significant clustering with industries, though Utility companies and Financials tended to be in close proximity to other companies within its sector.

Another helpful validation method could be to check the nearest neighbor for a sample of points in the embeddings space and checking if it makes sense. One measure, though potentially not very accurate, would be to see how often the nearest neighbor is within the same label (often industry, but I would like to test this with other labels).

I can also use the embeddings as features in the machine learning model and compare the performance against a baseline without the embeddings. This could show if there exists some useful information about the companies in the model.

Lastly, stability testing would also be very important. I can assess the stability by introducing small variations in the data and observing how much the embeddings change.

5.2 Implications of Research and Use Cases

Based on current testing, it appears that the model is capable of detecting similarities between companies; we saw this in our initial test with 23 companies from the S&P 100, where companies within the same industry tended to have much higher cosine similarity than those that were not.

This could have great potential implications for research. Understanding similarities between companies, especially those that are unknown, could be very helpful for academics, investors, and financial experts alike. Additionally, this toolkit will become increasingly helpful as people start using 10-K data in financial analyses more often; there is already increasing literature on the power of the rich data within 10-K filings.

It is important to note, as I did above, that additional validation is required to have a larger confidence in the embeddings. I mention below some next steps in which this can be improved.

5.3 Future Releases

The following outlines some plans for improving the current toolkit in future releases:

Improving Accuracy

To potentially improve the accuracy of the embeddings, I would like to explore using Claude to summarize the text before encoding and embedding it. Claude is known to have a larger window that can fit the 10-K data and provides rich summaries. This approach could also improve interpretability as users can feasibly read the summary for each company should they want to.

Improving Efficiency

In order to improve efficiency, I'd like to create a database with preset processed text for all 2023-2024 10-K filings. This would be quite a large database. However, retrieving the text from a

database, from the users perspective, is much faster, more feasible, and eliminates the operational cost of the sec-api.io API.

Improving Data Comprehension

To further enhance the toolkit's use in providing 10-K filing insights, developing some knowledge graph or employing GNNs across companies would help in mapping and analyzing the complex relationships between companies within and across filings. This would also help in visual representations of relationships among companies, sectors, and other sets of companies. Should I expand on this project with GNNs, I would likely partly validate the results by testing its predictive power about company performance, risk factors, and/or market movements based on the network's topology.

5.4 Package Usage

First, initialize tickers and sections as a list:

```
tickers = ['AAPL', 'MSFT', 'GOOG', 'FB'] # Tickers to include
sections = ['1', '1A'] # Sections to include in 10-K filings
```

Initialize model, retreive data, process data, tokenize data, and embedd data:

```
model = AnnualFilingCluster(os.getenv('EDGAR_HEADER')) # Initializing the model
text_data = model.retreiveTextData(tickers=tickers, sections=sections,
 ↪  years_back=1, sec_api_key=sec_api_key) # Retreive text data
processed_data = model.preprocessData(df=text_data, SECBERT=True) # Process data
tokenized_data = model.tokenizeData(processed_data=processed_data, SECBERT=True)
embedded_data = model.embedData(tokenized_data=tokenized_data) # Embed data
```

Cluster data and visualize the clusters:

```
clustered_data, reduced_embeddings =
 ↪  model.clusterData(embedded_data=embedded_data, model_type='KMeans',
 ↪  KMEANS_n_clusters=n_clusters, random_seed=109) # Cluster data
model.visualizeClusters(cluster_data=clustered_data,
 ↪  reduced_embeddings=reduced_embeddings,
 ↪  save_file_path='visualize_clusters.png') # Visualize clusters and save image
```

Plot cosine similarity heatmap and find proximity validation accuracy:

```
cosine_similarity_matrix =
 ↪  model.plotCosineSimilaryHeatmap(embeddings=reduced_embeddings,
 ↪  labels=tickers) # Plot cosine similarity heatmap
results = model.proximityValidation(embeddings=reduced_embeddings,
 ↪  labels=labels, threshold=threshold_value)['accuracy']
```

6 References

[1] George Adosoglou et al. In: *Expert Systems with Applications* 163 (Sept. 2020). URL: https://www.sciencedirect.com/science/article/abs/pii/S0957417420308186.

[2] Business Insider. Accessed: 30 Apr. 2024. URL: https://markets.businessinsider.com/index/components/s%5C&p_100.

[3] OpenAI. Accessed: 1 May 2024. URL: https://chat.openai.com.

[4] Tingyue Gan. UC Berkeley. Accessed: [Insert Access Date Here]. Apr. 2019. URL: https://cdar.berkeley.edu/sites/default/files/tgan190416risk.pdf.

[5] Hugging Face. Accessed: 30 Apr. 2024. URL: https://huggingface.co/nlpaueb/sec-bert-shape.

[6] MSCI. MSCI. Accessed: 30 Apr. 2024. URL: https://www.msci.com/our-solutions/indexes/gics.

[7] SEC EDGAR Filings API. Accessed: 30 Apr. 2024. URL: https://sec-api.io/.

[8] SEC Emblem. Nov. 2022. URL: https://www.sec.gov/edgar/sec-api-documentation.

[9] Saurabh Sehrawat. In: *SSRN* (Nov. 2019). URL: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3480902.

[10] Yi Yang et al. In: *INFORMS Journal on Computing* 34.1 (Jan. 2022), pp. 1–16. URL: https://dl.acm.org/doi/abs/10.1287/ijoc.2020.1046.