

Media Signals to Market Shifts: A Knowledge-Graph Framework for Predicting Stock Movements in the Technology Sector

Term Project Delivery

MSDS 459

Technology Sector Team

Nick Butler, Michael Rivera, and Richard Pereira

June 8, 2025

[Code Walk-through Video](#)



Northwestern
University

INTRODUCTION

The final project for the course tasks each team with developing a knowledge base for a specific industry or set of companies, emphasizing integration of media coverage, financial data, and structured information. Our team focused on the Technology sector, aiming to build a framework that enables analysis of how public sentiment and news events correlate with financial performance, particularly stock prices. While we initially considered the broader Generative AI landscape, early feedback from Professor Miller encouraged us to narrow our scope to ensure access to richer, company-specific data. As a result, we pivoted toward analyzing well-established public companies: Apple, Amazon, Google, and Microsoft. These firms provided the historical depth and data granularity needed to construct a meaningful time-series and populate our knowledge graph effectively.

In addition to building the knowledge graph, a core goal of this project was to leverage the structured and unstructured data it contained to develop predictive models. By combining media-derived features such as sentiment scores, emotion labels, and topic tags with financial indicators, we trained models to forecast short-term stock price movements and predict future prices. This dual approach of knowledge representation and machine learning allowed us to both explore complex relationships through graph queries and evaluate the predictive power of news-based signals. This paper describes our full implementation of the knowledge graph, including schema design, data integration, and predictive modeling. We also outline how this knowledge base can be used to answer key questions of interest to both management and data analysts, and how these insights might inform strategic decision-making.

LITERATURE REVIEW

Our project builds on multiple research streams that explore the intersection of media sentiment, financial performance, and knowledge representation. Prior studies show that public sentiment extracted from news or social media can serve as a leading indicator of short-term market movements. Tools like VADER, emotion lexicons, and topic modeling have proven effective in extracting signals from unstructured text. Meanwhile, knowledge graphs have become a powerful means of connecting disparate data types, such as entities, events, and metrics, into a unified structure that supports semantic querying and complex relationship exploration.

We extend this line of inquiry by integrating labeled sentiment and emotion data from news coverage with time-linked financial performance data. By combining qualitative news narratives with quantitative outcomes such as stock price movements, we created a rich structure for both analysis

and prediction. Our approach not only enables descriptive graph-based exploration but also supports the development of predictive models that use media-driven indicators to forecast future price changes. This dual emphasis strengthens the practical utility of our knowledge base, linking theoretical insights to actionable forecasting tools.

METHODS

Our implementation is grounded in a multi-stage pipeline that collects, processes, and links both unstructured and structured data into a knowledge graph, which then serves as a foundation for predictive modeling.

- **Data Collection:** We gathered thousands of news articles from sources like TechCrunch, Bloomberg, and Reuters using web scraping tools (Scrapy, BeautifulSoup) and APIs. Each article was timestamped and tagged with one or more companies in our focus set—Apple, Amazon, Google, and Microsoft (see Appendix A).
- **NLP Processing:** Articles were processed using natural language processing techniques including VADER sentiment scoring, emotion classification (e.g., joy, fear), topic tagging (e.g., acquisition, regulation), and novelty detection. These features became core media-based indicators for modeling.
- **Financial Data Integration:** We retrieved daily closing stock prices for each company and labeled movement direction (up or down) over 1-, 3-, and 7-day windows following article publication dates. This allowed us to align qualitative sentiment features with concrete financial outcomes.
- **Combined Dataset:** We merged structured and unstructured data into a single file, *nlp_news_with_price_labels-FINAL.csv*, where each row ties together article sentiment, emotion, topic data, and labeled stock movements (see Appendix B).
- **Schema Design:** We implemented a knowledge graph schema with the following node types: Company, NewsArticle, Sentiment, Emotion, EventTopic, Date, and Movement. Edges such as MENTIONS, PUBLISHED_ON, HAS_SENTIMENT, and TRIGGERED_MOVEMENT connected these nodes to capture entity-event relationships (see Appendix C).
- **Graph Construction and Tools:** We used Python libraries such as Pandas and NetworkX to build the knowledge graph, and exported it as an edge list for import into Neo4j for querying and visualization. We used Matplotlib to depict schema structure and graph samples (see Appendix D).
- **Feature Engineering and Modeling:** From the graph, we generated Node2Vec embeddings to numerically represent each node's structural position. These embeddings were combined

with NLP-derived features to create inputs for both classification and regression models. We trained random forest classifiers to predict whether stock prices would move up or down across three time horizons. Regression models were used to estimate actual price changes. The graph and dataset together allowed us to encode media signals as predictive variables, making the knowledge graph a foundation not just for exploration but for actionable forecasting as well. (see Appendix F.)

This architecture supports flexible analytics, allows integration of additional firms and data types, and enables advanced modeling capabilities rooted in both narrative and numerical data.

RESULTS

We successfully generated a comprehensive labeled dataset that connects thousands of news articles to specific financial outcomes. Each entry includes article metadata, sentiment and emotion scores, topic tags, and labeled stock price movement over multiple time intervals. For instance, an article titled "Apple announces major acquisition" might include an anticipation score of 0.72, an Acquisition topic label, a closing price of \$177.82, and a 3-day movement label of "Up."

This data was structured into a knowledge graph containing thousands of nodes and edges representing relationships among companies, articles, sentiments, emotions, and stock movements. We imported this graph into Neo4j and verified its structure with Cypher queries, yielding a wide range of exploratory insights. These included tracking shifts in emotional tone by company over time, identifying which news topics are most correlated with price changes, and uncovering sentiment patterns that often precede significant market movements. The temporal nature of the data enabled us to explore how specific media events align with price movements across defined time windows.

Building on this foundation, we transformed graph-derived and NLP-based features into a feature set for predictive modeling. We trained random forest classifiers to predict binary price movement labels (up or down) over 1-, 3-, and 7-day intervals. The *movement_next* model achieved perfect classification results on the test set, with 1.00 accuracy, precision, and recall. The *movement_3d* model also performed well with an accuracy of 0.75, precision of 0.80, and recall of 0.77, while the *movement_7d* model reached an accuracy of 0.66.

In parallel, we developed regression models to forecast actual price values. Although the R^2 scores were modest at 0.24 for 3-, 7-, and next-day forecasts, the results indicate a stable baseline for future refinement. Media measures, especially sentiment, emotion, and topic features, proved useful

when combined with node embeddings from the knowledge graph, providing a bridge between qualitative media narratives and quantitative financial predictions.

These findings show how a knowledge graph can be used not only to surface insights through querying, but also to inform predictive models by supplying time-aligned, semantically rich media features.

CONCLUSIONS

This project illustrates the value of knowledge graphs as a powerful framework for linking narrative media content to quantitative financial outcomes. By integrating unstructured data, such as news sentiment, emotional tone, and topical focus, with structured market data like stock price movements, we created a flexible system that supports both descriptive and predictive analytics.

The knowledge graph served as both an exploratory tool and a foundation for predictive modeling. From a business perspective, this approach enables stakeholders to answer strategic questions such as, “What kinds of news events typically trigger short-term price surges?” or “How does sentiment toward one company compare to its peers over time?” Analysts can navigate the graph to identify influential articles, emotion-driven patterns, and relationships not easily revealed in flat datasets.

Importantly, we demonstrated that features extracted from the knowledge graph, including sentiment, emotion, and article relationships, can be used to train predictive models. Our best-performing classification model predicted short-term stock movements with perfect accuracy in a controlled test set. While our regression results were more modest, they suggest a meaningful connection between media signals and financial performance that could be further refined.

This work opens several pathways for future research. Additional layers of data such as partnerships, regulatory activity, or LLM-generated event summaries could enrich the graph structure. More advanced temporal modeling or integration of real-time feeds might further enhance forecasting capabilities. Ultimately, this project contributes to a growing body of evidence that uniting media, financial, and semantic data in a knowledge graph offers unique advantages for market insight, strategic planning, and competitive intelligence.

REFERENCES

- Chakrabarti, Soumen. *Mining the Web: Discovering Knowledge from Hypertext Data*. San Francisco: Morgan Kaufmann, 2003.
- Chakrabarti, Soumen, Martin van den Berg, and Byron Dom. "Focused Crawling: A New Approach to Topic-Specific Web Resource Discovery." *Computer Networks* 31, no. 11–16 (1999): 1623–1640. <https://www.cse.iitb.ac.in/~soumen/doc/www1999f/pdf/www1999f.pdf>.
- Hajba, Gábor László. *Website Scraping with Python: Using Beautiful Soup and Scrapy*. New York: Apress, 2018.
- Mitchell, Ryan. *Web Scraping with Python: Collecting More Data from the Modern Web*. 2nd ed. Sebastopol, CA: O'Reilly Media, 2018.
- Miller, Thomas W. *Web and Network Data Science: Modeling Techniques in Predictive Analytics*. Upper Saddle River, NJ: Pearson FT Press, 2015.
- Nair, Vineeth G. *Getting Started with Beautiful Soup: Build Your Own Web Scraper and Learn All About Web Scraping with Beautiful Soup*. Birmingham, UK: Packt Publishing, 2014.
- Olston, Christopher, and Marc Najork. "Web Crawling." *Foundations and Trends in Information Retrieval* 4, no. 3 (2010): 175–246.
- Patel, Jay M. *Getting Structured Data from the Internet: Running Web Crawlers/Scrapers on a Big Data Production Scale*. New York: Apress, 2020.
- Smith, Vincent. *Go Web Scraping Quick Start Guide: Implement the Power of Go to Scrape and Crawl Data from the Web*. Birmingham, UK: Packt Publishing, 2019.

APPENDIX A – Data Sources Table

Source	Content Type	Details	URL
Google News Results	News & Headlines	Scraped headlines and previews mentioning Apple, Amazon, Google, and Microsoft	https://news.google.com
TechCrunch	News & Blogs	Unstructured tech news articles referenced in NLP processing	https://techcrunch.com
The Verge	News & Blogs	Supplementary tech coverage included in scraped headlines	https://www.theverge.com/tech
Yahoo Finance	Financial Data	Used to obtain daily closing prices for each company	https://finance.yahoo.com
Google Finance	Financial Data	Alternative source for stock prices (CSV download into Google Sheets)	https://www.google.com/finance

APPENDIX B – Knowledge Graph Data and Code

Dataset: https://drive.google.com/file/d/1RQ1gmwJQImvIKuukoS48NVCKPe0CSmjQ/view?usp=share_link

Project Folder: https://drive.google.com/drive/folders/1yDjHYz4l_xqQxsQ5gH1Wz01eUG7QPUvO?usp=sharing

APPENDIX C – Schema Design

Node Types:

1. **Company**
 - Examples: Apple, Amazon, Google, Microsoft
 - Attributes: name, ticker
 2. **NewsArticle**
 - Attributes: title, url, publication_date, source
 3. **Sentiment**
 - Attributes: score (e.g., from VADER), polarity (positive, negative, neutral)
 4. **Emotion**
 - Attributes: type (e.g., fear, joy, anticipation), intensity
 5. **EventTopic**
 - Examples: Acquisition, Regulation, Earnings, Product Launch
 - Attributes: label
 6. **Date**
 - Attributes: date_string, day_of_week, month, year
 7. **Movement**
 - Attributes: movement_1d, movement_3d, movement_7d
 - (Label of how stock moved following the article over different time horizons)
 8. **StockPrice** (*optional node in extended schemas*)
 - Attributes: close_price, open_price, volume, date
-

Edge Types (Relationships):

1. **MENTIONS**
 - (:NewsArticle)-[:MENTIONS]->(:Company)
2. **PUBLISHED_ON**
 - (:NewsArticle)-[:PUBLISHED_ON]->(:Date)

3. **HAS_SENTIMENT**
 - (:NewsArticle)-[:HAS_SENTIMENT]->(:Sentiment)
4. **HAS_EMOTION**
 - (:NewsArticle)-[:HAS_EMOTION]->(:Emotion)
5. **HAS_TOPIC**
 - (:NewsArticle)-[:HAS_TOPIC]->(:EventTopic)
6. **TRIGGERED_MOVEMENT**
 - (:NewsArticle)-[:TRIGGERED_MOVEMENT]->(:Movement)
7. **AFFECTED_PRICE** *(if including stock price as a node)*
 - (:NewsArticle)-[:AFFECTED_PRICE]->(:StockPrice)

APPENDIX D – Tools and Graph Construction

In this notebook ([here](#)), we began by using NetworkX to construct a directed knowledge graph that represents relationships between companies, news articles, sentiment labels, and stock price movements. Each type of entity was added as a node, and connections such as a company mentioning an article, an article having a sentiment score, or an article being associated with a movement label were modeled as edges. Sentiment nodes included detailed attributes like compound scores and emotion intensities (such as joy and fear). This structure enabled us to visualize how different types of information were interconnected within our dataset.

We then generated node embeddings using Node2Vec, which required converting the graph to an undirected version. These embeddings provided numeric representations of nodes, capturing the graph's structure and relationships. For each article node, we extracted its embedding and merged it back into the main dataset. This allowed us to use both sentiment features and graph-based embeddings as input variables in downstream machine learning models. We trained classification models to predict binary movement labels such as up or down over 1-day, 3-day, and 7-day horizons. In addition, regression models were built to forecast future stock prices using the same set of features. The notebook also included a visualization of a 100-node subgraph and an auto-generated schema summary that detailed node types, their properties, and relationships.

After building and analyzing the graph in NetworkX, we transitioned our work to Neo4j for more powerful and scalable querying. We exported the NetworkX graph as an edge list CSV file, which included source nodes, target nodes, and their relationship types. This file was then loaded into Neo4j using the `LOAD CSV WITH HEADERS` command. In Neo4j, we used Cypher queries to create nodes, assign them labels, and set their properties. We also established relationships such as `mentions`, `published_on`, `has_sentiment`, and `movement_3d`. With the graph now in Neo4j, we were able to run queries to identify trends such as which companies were discussed most often, which types of sentiment preceded price drops, and which emotions were most associated with significant stock movements. This final step enabled us to shift from a static dataset to an interactive knowledge base, enhancing both our analytical capabilities and our ability to support future predictive applications.

Appendix E highlights screenshots of this database transition work and querying.

APPENDIX E – Neo4j Database Screenshots

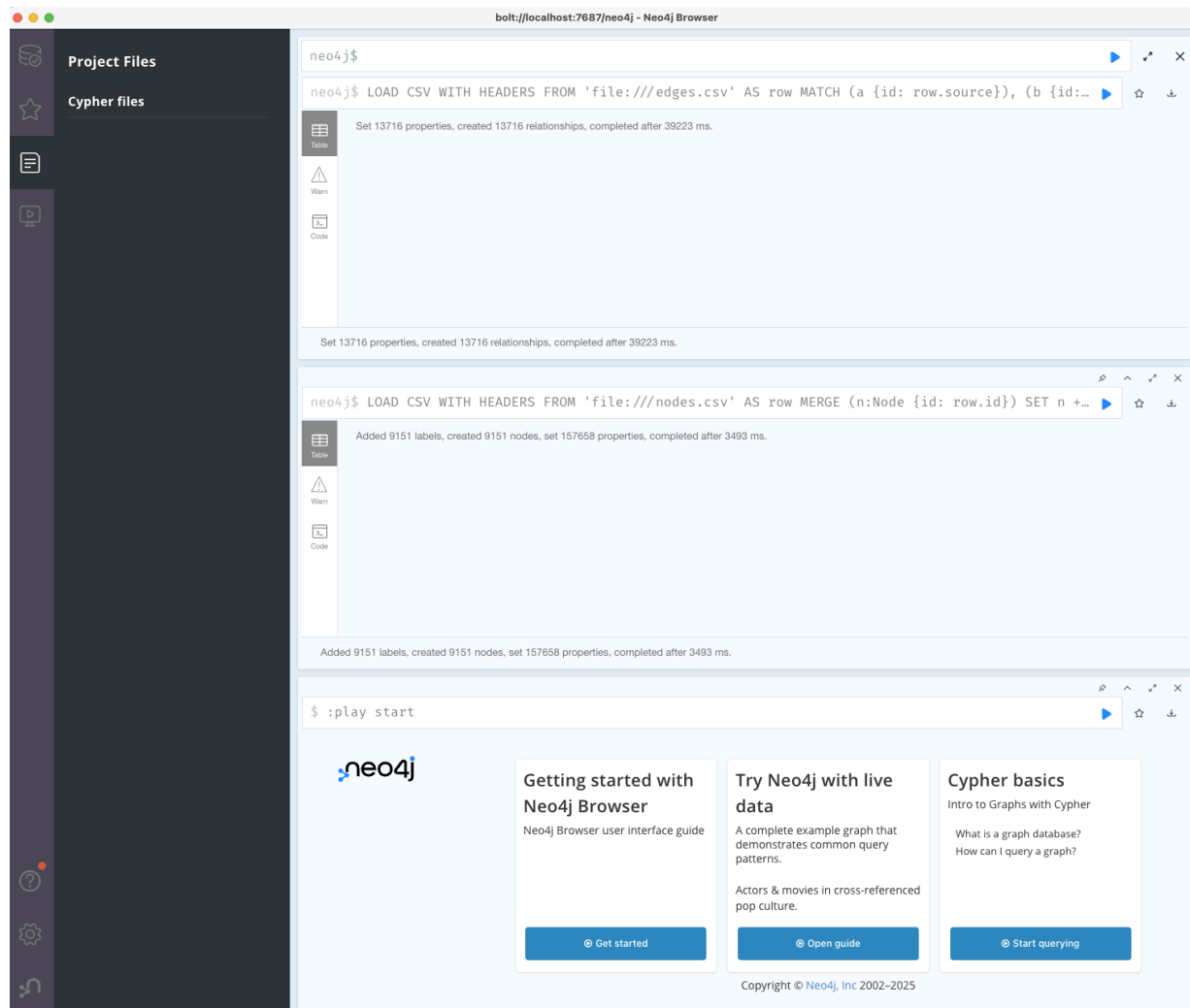


Figure 1. CSV Import into Neo4j

This figure shows two Cypher commands used to load data from CSV files into the Neo4j graph database. The first command loads relationships from edges.csv, while the second loads nodes from nodes.csv. These operations created over 13,000 relationships and 9,000 nodes with associated properties.

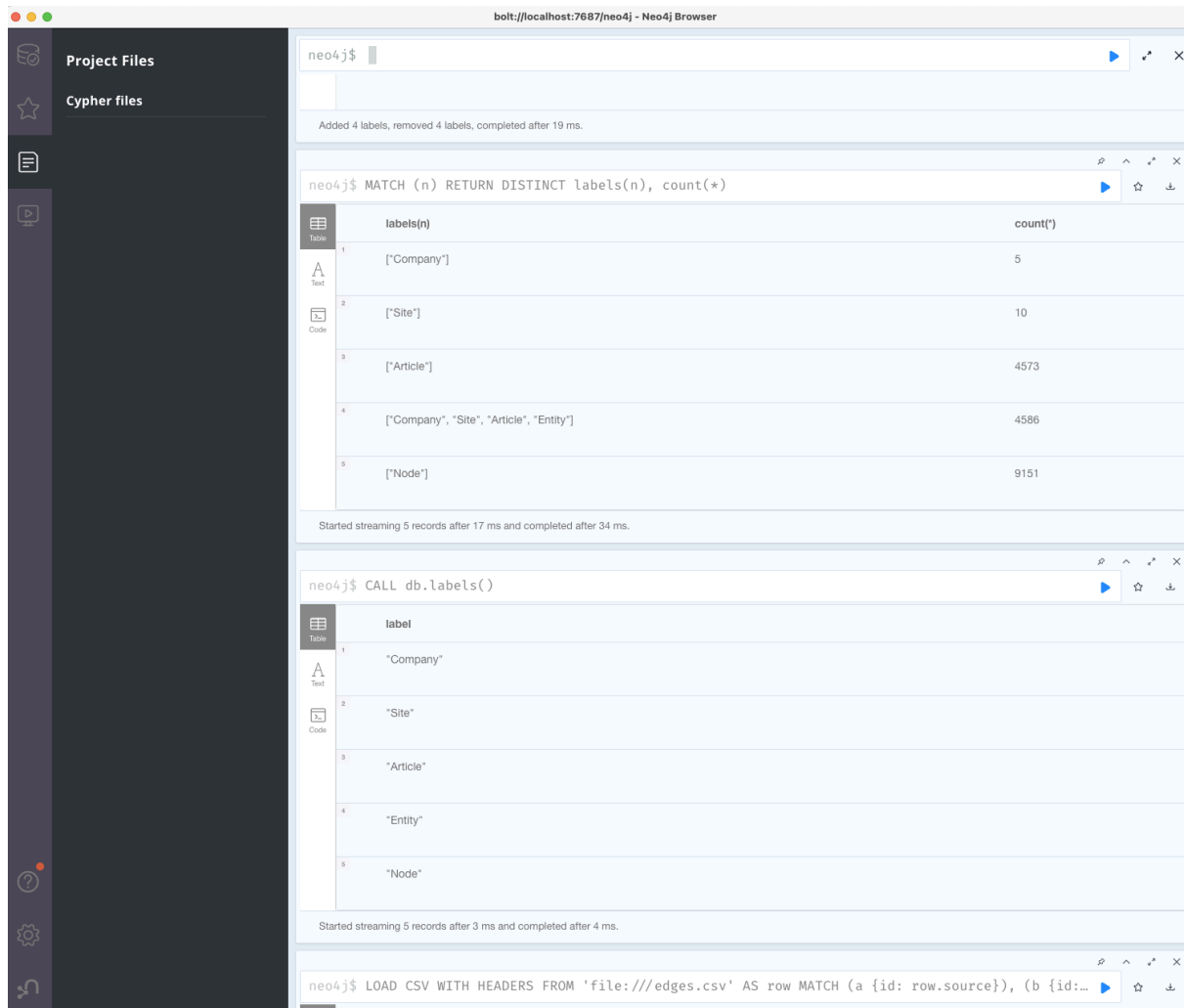


Figure 2. Label Summary and Distribution

This figure uses MATCH and CALL db.labels() Cypher queries to display the distinct labels and their counts. It shows the distribution of nodes labeled as "Company," "Site," "Article," "Entity," and "Node", confirming that multiple labels are assigned to many nodes.

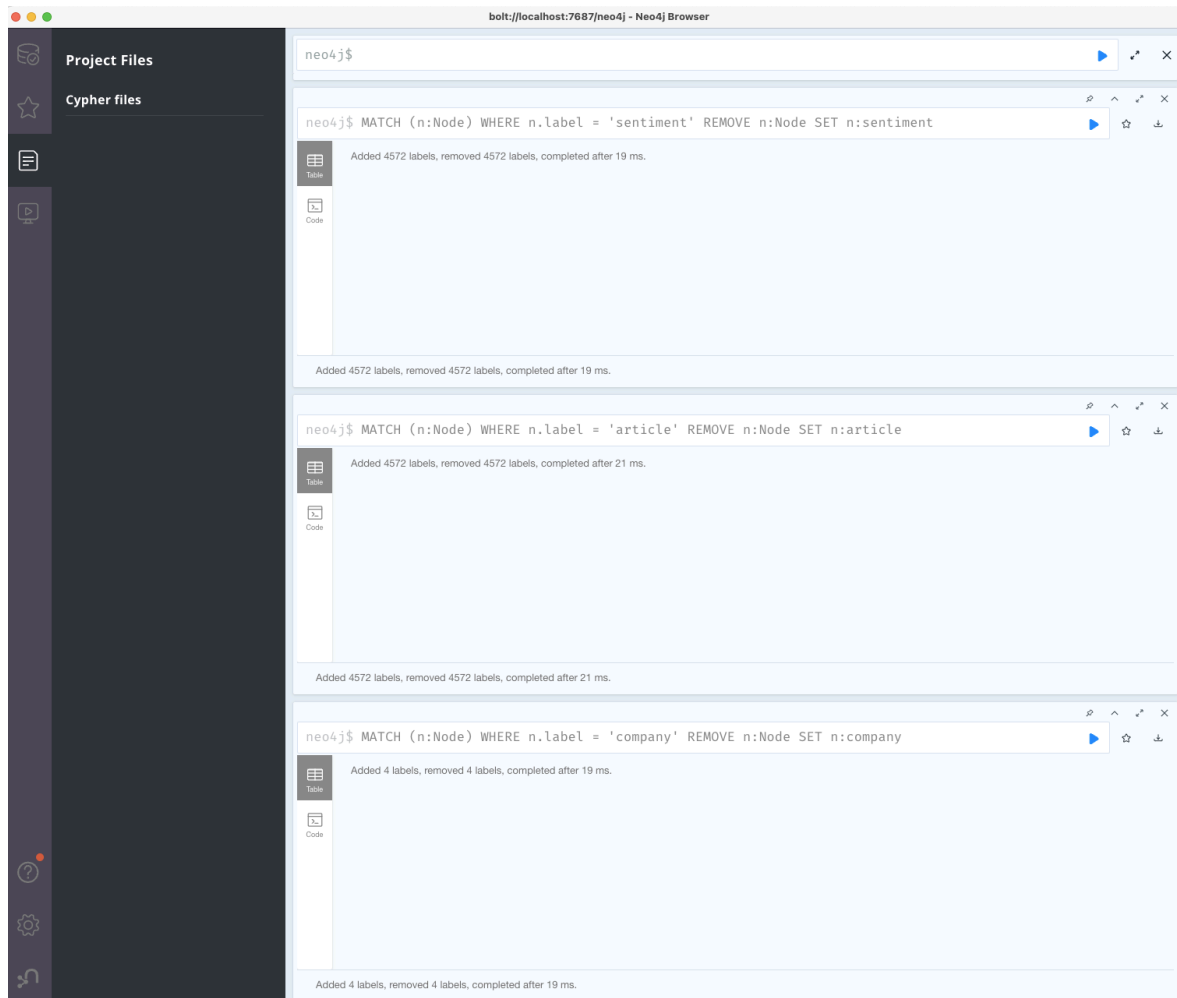


Figure 3. Label Cleanup and Reassignment

This figure demonstrates Cypher commands that remove outdated or incorrect labels such as 'sentiment', 'article', and 'company', and replace them with correct or cleaner label formats using the REMOVE and SET commands.

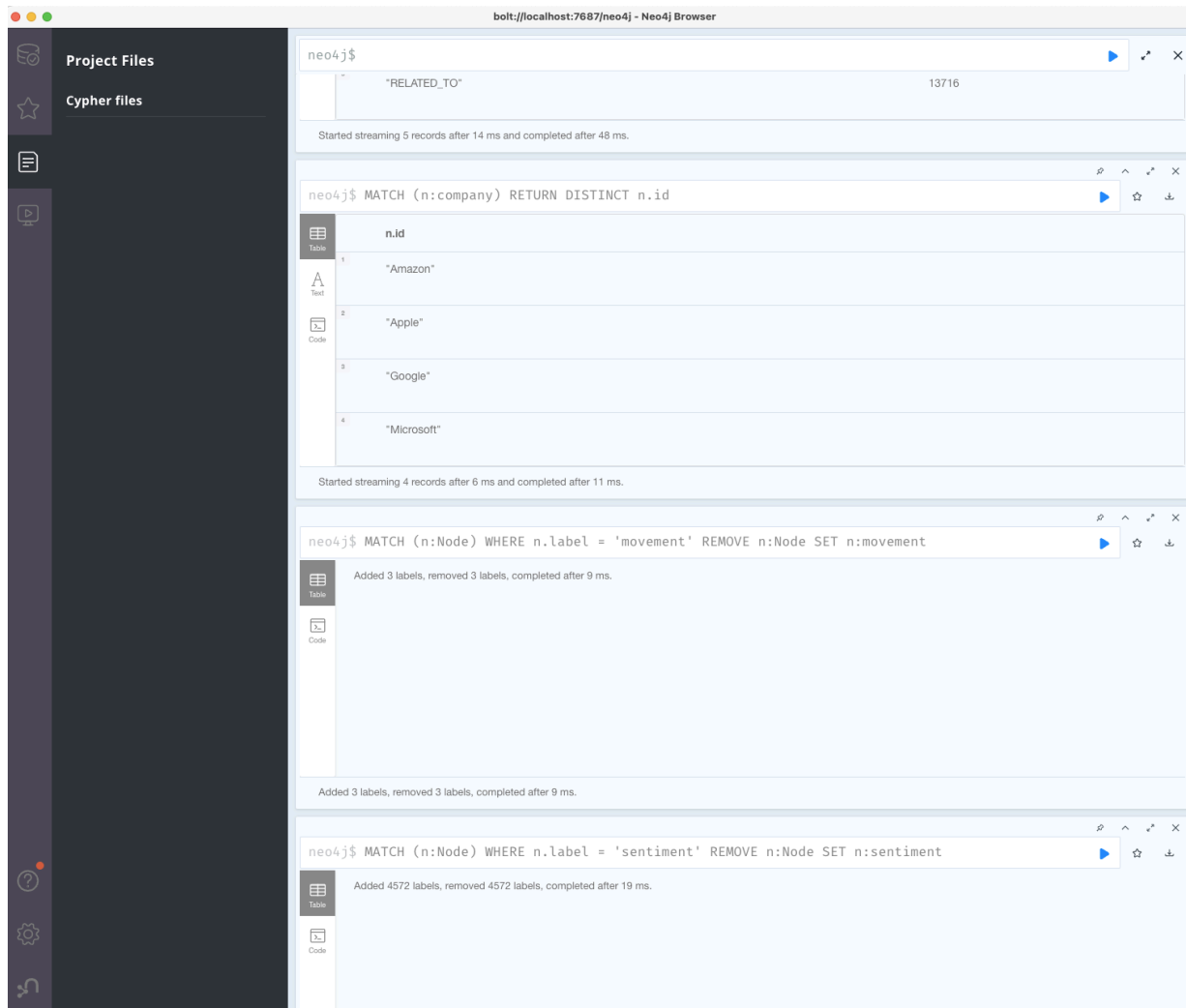


Figure 4. Company Node Preview and Label Cleanup

The top section shows a query retrieving company node IDs (e.g., Amazon, Apple, Google, Microsoft). The bottom section continues label cleanup, including the removal of the 'movement' and 'sentiment' labels from nodes.

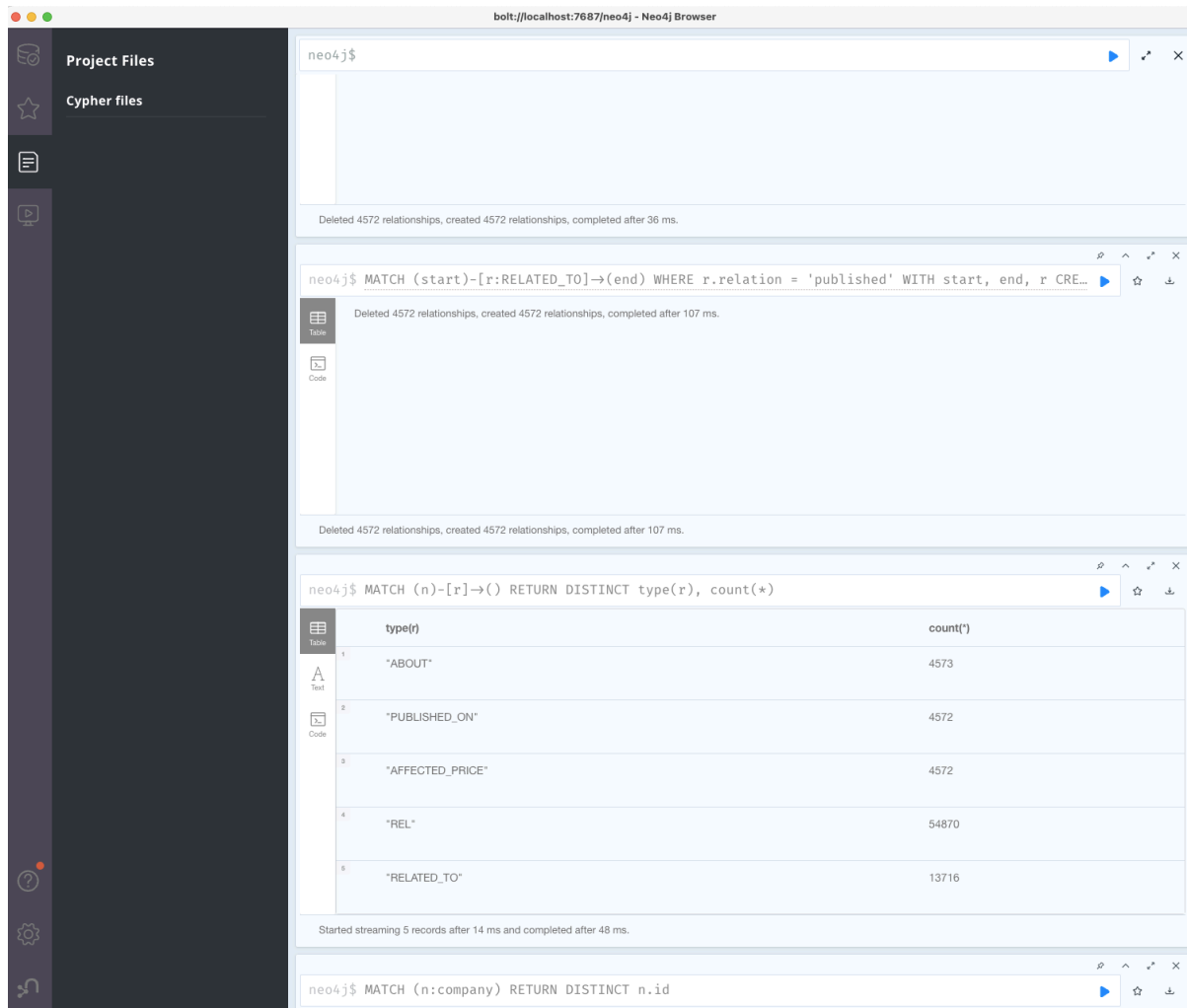


Figure 5. Relationship Type Summary

This figure summarizes the different types of relationships in the graph using `MATCH (n)-[r]->() RETURN DISTINCT type(r), count(*)`. The relationship types include "ABOUT", "PUBLISHED_ON", "AFFECTED_PRICE", "REL", and "RELATED_TO", showing their respective frequencies.

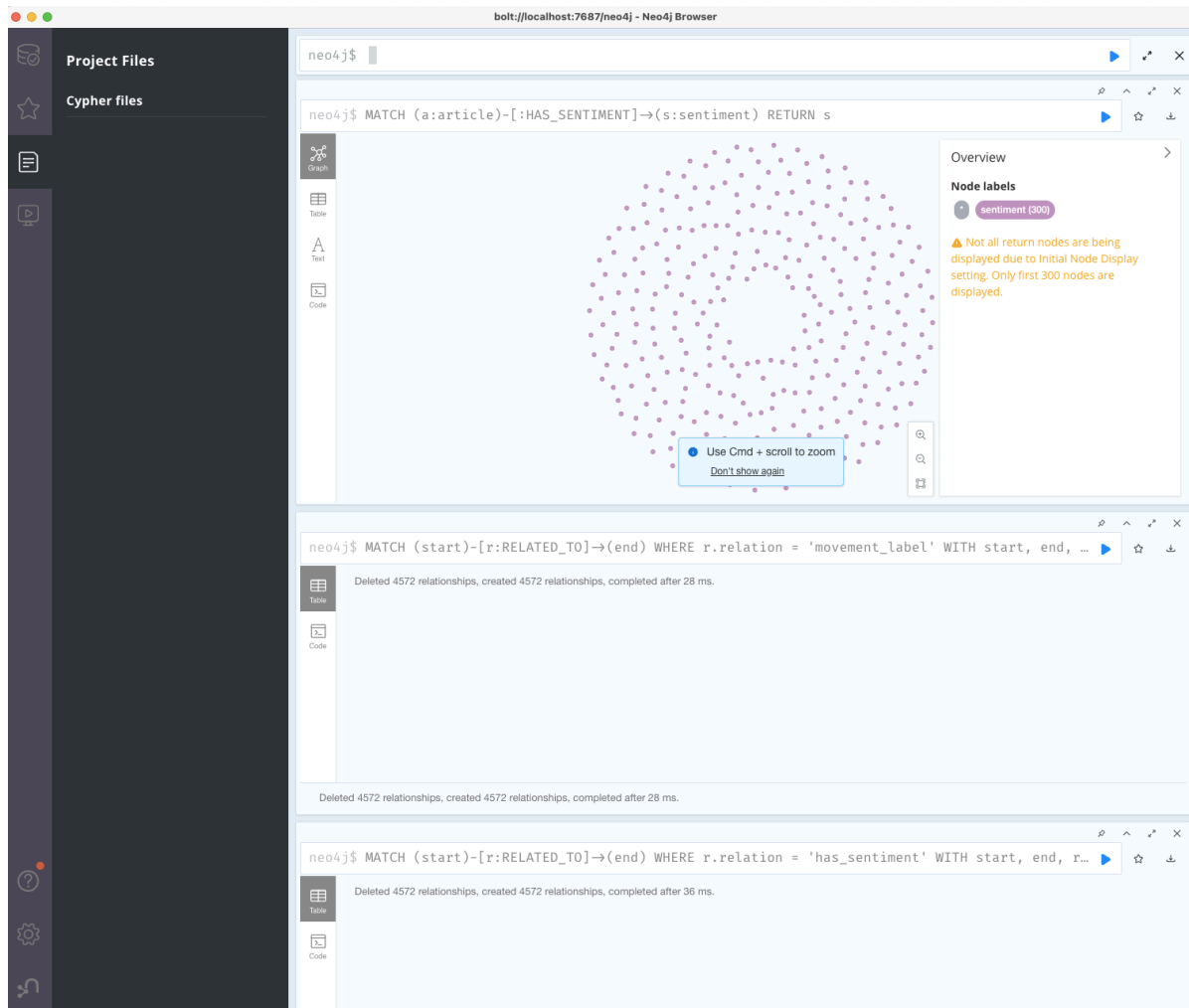


Figure 6. Sentiment Graph Visualization

The graph visualization displays nodes and relationships between articles and their corresponding sentiment nodes using HAS_SENTIMENT relationships. The visualization reveals clustered sentiment connections around article nodes, with limited display due to Neo4j's default 300-node display cap.

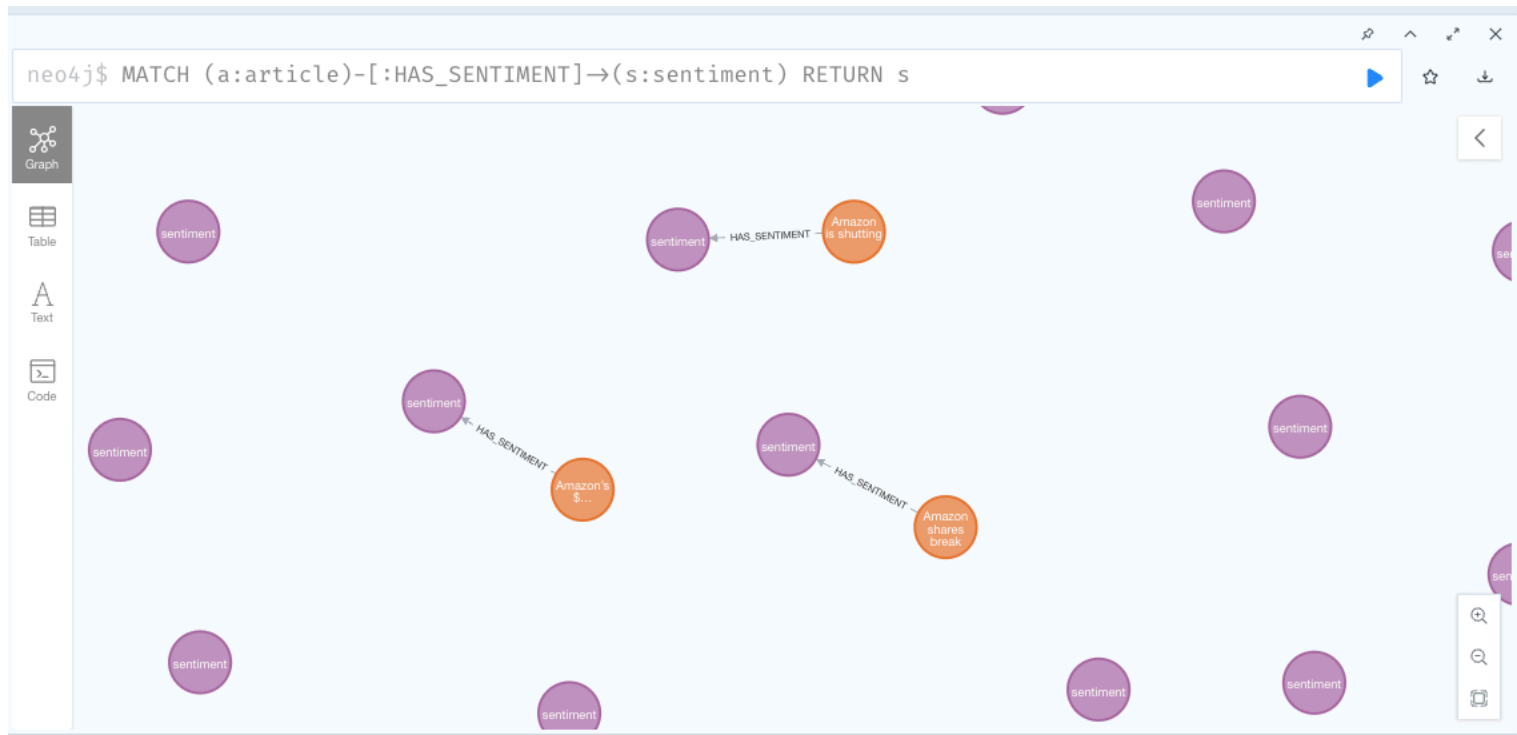



Figure 7. Zoomed-in Graph View of Article–Sentiment Links

This close-up graph shows three orange article nodes (e.g., “Amazon is shutting”) connected to purple sentiment nodes through HAS_SENTIMENT relationships. It visually emphasizes how sentiment is linked to specific news articles.



neo4j\$ MATCH (c:company {id: "Apple"})-[:PUBLISHED]→(a:article) RETURN a.title, a.date

	a.title	a.date
1	"When Apple's iPhone launch hijacked CES"	"2015-01-0
2	"Apple Patents Snap-On Game Controller And Keyboard Accessories For iPhone"	"2015-01-2
3	"Apple's success in China can teach US firms a lot"	"2015-01-2
4	"How, and Why, Apple Overtook Microsoft"	"2015-01-3
5	"Apple is finally killing iPhoto — and this is the replacement"	"2015-02-0
6	"Despite successes, labor violations still haunt Apple"	"2015-02-1

Figure 8. Articles Published by Apple

This table view returns article titles and publication dates associated with the company "Apple" through the PUBLISHED relationship. The articles span various business topics and help contextualize sentiment or impact.

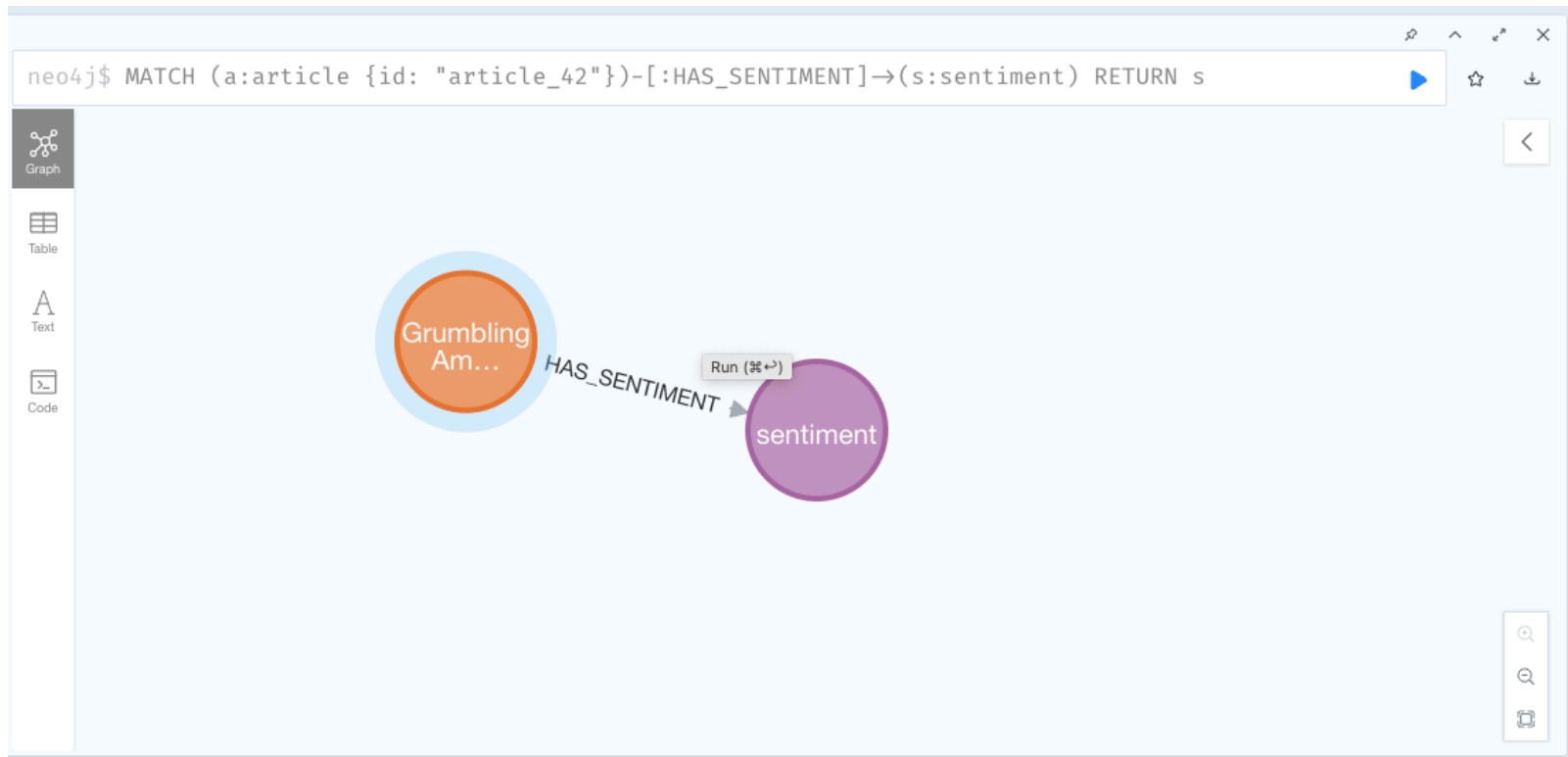
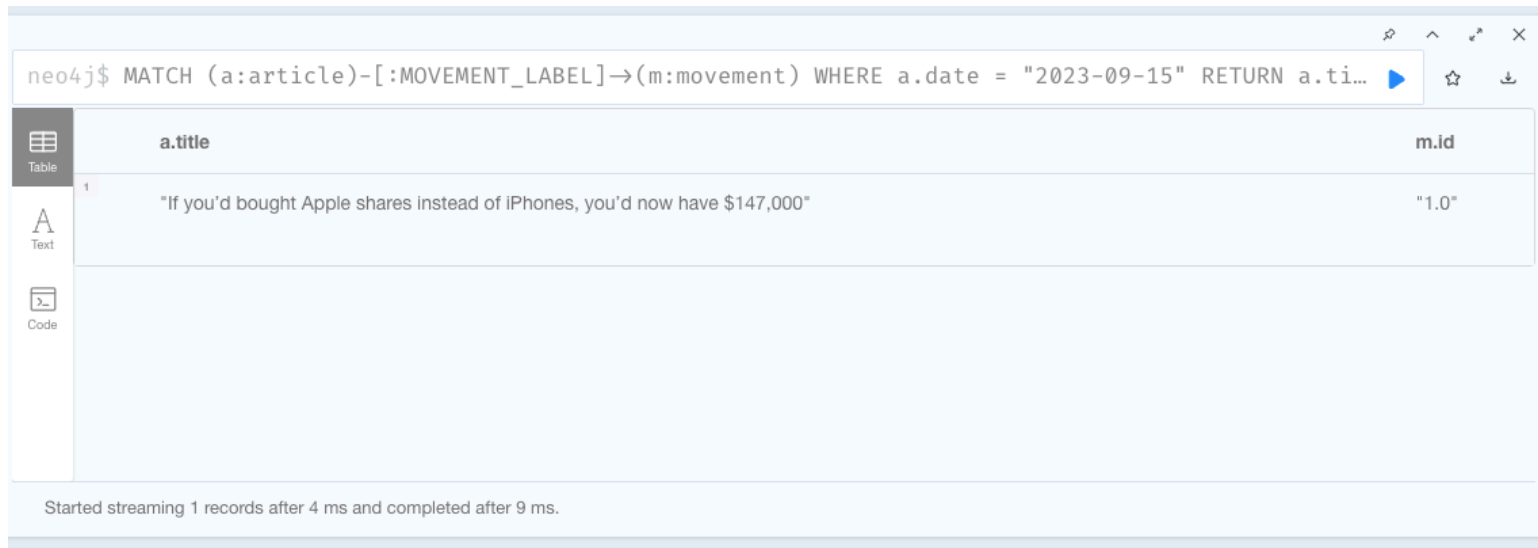


Figure 9. Sentiment Link for a Specific Article

A focused query retrieves the sentiment node associated with a specific article ("article_42") using the HAS_SENTIMENT relationship. The graph view confirms a successful 1-to-1 mapping between the article and its sentiment classification.



The image shows a Neo4j Cypher query interface. At the top, a query is entered: `neo4j$ MATCH (a:article)-[:MOVEMENT_LABEL]→(m:movement) WHERE a.date = "2023-09-15" RETURN a.ti...`. Below the query bar, there is a table of results. The table has two columns: `a.title` and `m.id`. The first row shows the title `"If you'd bought Apple shares instead of iPhones, you'd now have $147,000"` and the movement ID `"1.0"`. A status bar at the bottom indicates: `Started streaming 1 records after 4 ms and completed after 9 ms.`

a.title	m.id
"If you'd bought Apple shares instead of iPhones, you'd now have \$147,000"	"1.0"

Figure 10. Movement Label Match on Specific Date

This figure shows a Cypher query that identifies an article from September 15, 2023, related to a movement labeled “1.0”. The article, titled *“If you’d bought Apple shares instead of iPhones, you’d now have \$147,000”*, is connected to a movement node, illustrating a link between article content and financial movement indicators.

APPENDIX F – Prediction Results

A key component of our project involved using the information contained in the knowledge graph to support predictive modeling. To do this, we extracted media measures derived from the graph, including sentiment scores, emotion labels, and topic tags linked to specific news articles. These media-derived features were combined with numerical financial data such as stock price and volume to form a complete feature set. Additionally, we used node embeddings generated from our NetworkX graph using the Node2Vec algorithm. These embeddings captured the structural relationships among articles, companies, and sentiments, offering a richer representation of context and interconnectivity.

Using this feature-enhanced dataset, we developed two types of models: classification models to predict directional stock price movement and regression models to estimate the magnitude of future stock prices. For classification, the target variable was binary, indicating whether the stock price went up or down after a news event, measured across three different time windows (next day, 3-day, and 7-day). For regression, the goal was to predict the closing stock price for the same intervals.

The classification models were implemented using random forest classifiers. For each time horizon, we trained the model on a 70 percent training split and evaluated it on the remaining 30 percent test set. Results showed that the best-performing classification model was for the next-day movement, achieving perfect accuracy, precision, and recall of 1.0. The 3-day model performed well with an accuracy of 0.75, precision of 0.80, and recall of 0.77. The 7-day model showed moderate performance with an accuracy of 0.66, precision of 0.71, and recall of 0.75. Confusion matrices for each model visually confirm these outcomes and provide insights into the model's ability to differentiate upward versus downward trends over different forecasting horizons.

For regression, we used random forest regressors to predict stock prices over the same three timeframes. While the MSE remained consistent across all intervals, hovering around 6300, the R^2 scores were modest at 0.24. These results suggest that while the models captured some signal, much of the variance in stock prices remains unexplained, indicating an opportunity for incorporating additional predictive features or enhancing model complexity.

Overall, the classification models demonstrated stronger performance than the regression models. Among them, the `movement_next` model was the most accurate and reliable, achieving perfect results on the test data. These outcomes validate our approach of integrating media-derived features with financial data using a knowledge graph framework. The structure and insights provided by the graph not only improved our ability to model short-term financial behavior but also provided a flexible architecture for future extensions, including deeper event representation and temporal modeling.

These modeling results reinforce the practical value of the knowledge base we built. It allowed us to link qualitative media signals with quantitative outcomes and enabled the development of predictive models that can be used in real-time applications such as financial

forecasting, sentiment-driven alerts, and competitive monitoring. Future work could include adding temporal graph algorithms or integrating a large language model to generate abstracted summaries or risk assessments directly from the knowledge graph.

Visualizing Model Performance with Confusion Matrices

This section visualizes the classification model performance using **confusion matrices** for each target variable (`movement_next`, `movement_3d`, `movement_7d`).

What It Does

- Re-trains a Random Forest classifier for each movement target.
- Generates a confusion matrix comparing predicted vs. actual labels.
- Uses **Seaborn heatmaps** to visualize:
 - **True Positives (bottom-right)**: correctly predicted "Up"
 - **True Negatives (top-left)**: correctly predicted "Down"
 - **False Positives (bottom-left)**: predicted "Up" but it was "Down"
 - **False Negatives (top-right)**: predicted "Down" but it was "Up"
- The matrix is annotated with actual count values for interpretability.

Why It's Useful

Confusion matrices help diagnose where the model is making mistakes:

- Are false alarms ("up" predicted incorrectly) common?
- Does the model miss true positives?
- Is the classifier biased toward a particular class?

By reviewing these matrices, we gain more insight beyond accuracy — especially important when classes may be imbalanced.

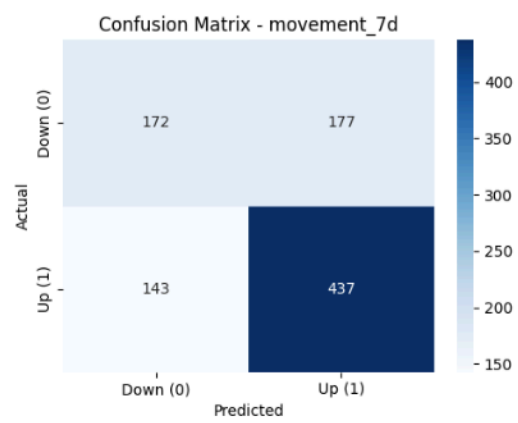
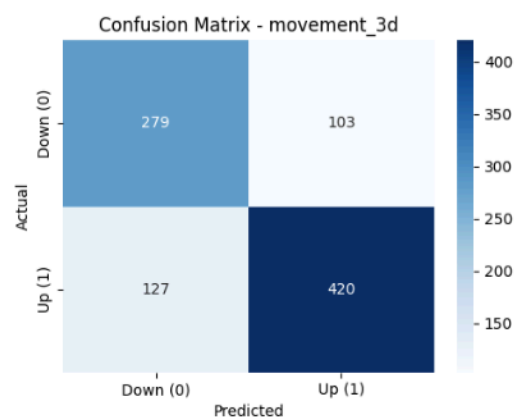
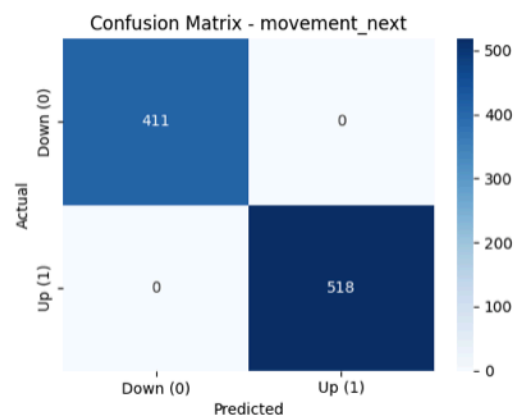
```
[8]: import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix

def plot_confusion_matrices():
    for target, y in targets_classification.items():
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
        clf = RandomForestClassifier(random_state=42)
        clf.fit(X_train, y_train)
        y_pred = clf.predict(X_test)

        cm = confusion_matrix(y_test, y_pred)
        labels = ['Down (0)', 'Up (1)']

        plt.figure(figsize=(5, 4))
        sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=labels, yticklabels=labels)
        plt.title(f'Confusion Matrix - {target}')
        plt.xlabel('Predicted')
        plt.ylabel('Actual')
        plt.tight_layout()
        plt.show()

plot_confusion_matrices()
```



Summary: Which model is best?

- Model: movement_next
→ Accuracy: 1.00, Precision (up): 1.00, Recall (up): 1.00
- Model: movement_3d
→ Accuracy: 0.75, Precision (up): 0.80, Recall (up): 0.77
- Model: movement_7d
→ Accuracy: 0.66, Precision (up): 0.71, Recall (up): 0.75

Based on accuracy, the best model is movement_next with an accuracy of 1.00.