Modelling Multimodal Financial Data with Inductive Logic Programming and Ontologies

Can Erten

Doctor of Philosophy

University of York

Computer Science

September 2025

Abstract

This thesis describes an attempt to implement automated, machine learning-based algorithms for processing financial reports with the aim of extracting relevant, actionable information. The purpose of the research is to apply hybrid machine learning to the financial domain using both structured and unstructured data.

The ontology of Security Exchange Commission (SEC) financial reports provides a suitable model for the representation of structural data, such as the company management hierarchy, the main shareholders and the relationships between individuals in these roles and the companies in which they play a role. Using SEC financial reports, I built an ontology by extracting and transforming the filings into a structured graph of companies, people and their relations.

Using the ontology that I built, I show how a hybrid learner can be used to find strategies and rules using Inductive Logic Programming (ILP) and traditional machine learning algorithms. This research shows that it is possible to increase the probability of selecting cointegrated pairs of stocks compared to random selection. This has the potential to enhance the decision-making process for choosing the right pairs of stocks while saving on computation time and resources.

To address scalability when learning from large ontologies, I developed P-CONNER, a concurrent Description Logic learner that accelerates the learning process using data-parallel algorithms on both the CPU and GPU. I conducted experiments on benchmark tests and demonstrated that P-CONNER can learn from large ontologies much faster than traditional learners.

Overall, the techniques I introduced in this thesis help automate the construction of a financial ontology from company reports. Combining symbolic and sub-symbolic machine learning for a finance pair trading strategy demonstrates statistically significant improvement by utilising an ontology structure with interpretable rules. This is facilitated by the hardware-independent concurrent Description Logic learner, which enables scalable learning from ontologies.

Contents

List of Figures

List of Tables

To my family.

Acknowledgements

I have been very fortunate to have the support of my supervisor, Dr. Dimitar Kazakov, who has provided me with invaluable guidance throughout my PhD. Dimitar's expertise in multiple areas of machine learning and his innovative ideas have been instrumental in driving my research.

I would like to acknowledge the time and effort of several students at the University of York who used my data and learner system in their work, especially Tianda, Neel, and Roscoe.

I would like to thank my father-in-law, Philip, for taking the time to review my thesis and providing valuable feedback.

Finally, I thank my wife, Harriet, for everything.

Declaration

I declare that this thesis is a presentation of original work and I am the sole author. This work has not previously been presented for a degree or other qualification at this University or elsewhere. All sources are acknowledged as references.

Except where stated, all of the work contained in this thesis represents the original contribution of the author.

Chapter 6 describes my own implementation of a learner in description logic, P-CONNER. This contribution extends the work of CONNER [1].

Parts of the research described in this thesis have previously been published in the following peer-reviewed papers and edited proceedings:

- Unsupervised Learning of Functional Groups for Computational Chemistry [2] paper relates to Chapter 2, where I explore the background of Inductive Logic Programming and symbolic machine learning. It also provides context for Chapter 6 where I use concurrent computation in description logic.
- Lecture Notes in Computer Science Inductive Logic Programming 29th International Conference [3] edited volume helped me to write Chapter 2 where Inductive Logic Programming, knowledge representation and description logic is presented.
- Pair Trading with an Ontology of SEC Financial Reports [4] paper is used in Chapter 3 for the ontology construction methodology from SEC reports and Chapter 4 for the evaluation of company pairs for pair trading strategy.
- Ontology Graph Embeddings and ILP for Financial Forecasting [5] paper relates to Chapter 5 where ontology graph embeddings, clustering and Inductive Logic Programming are combined for financial forecasting rules. It also motivated Chapter 6 for the need for scalable learning from ontologies.

Chapter 1

Introduction

This thesis describes an attempt to implement automated, machine learning-based procedures for the processing of financial reports with the aim of extracting relevant, actionable information.

1.1 Context, Motivation and Problem Statement

Financial forecasting and algorithmic trading rely on identifying actionable market signals to create strategies. Pair trading is a market-neutral strategy that relies on identifying two highly correlated or cointegrated assets. This is achieved by tracking the price spread between two assets, where traders can take opposite long and short positions to profit from the expected convergence of their prices, regardless of broader market trends and it is a highly effective strategy. By maintaining a proportion of each company in a portfolio, analysts and traders can make a profit from the mean reversion regardless of broader market movements.

The primary challenge in pair trading is the computational complexity in identifying cointegrated pairs. The number required to do the statistical tests grows quadratically with the number of pairs of companies analysed. For instance, for a pool of 10,000 companies, nearly 50,000,000 pairwise comparisons are required. Filtering these company pairs to reduce the search space is a critical problem for traders and also, in a generalised problem space, researchers, as the problem exist in other domains as well. I have chosen finance domain because the data is widely available and mostly free and accessible.

Historically, the analysts using traditional machine learning approaches to pair trading have relied on purely quantitative data to find price correlations to filter the pairs. Numerical tests like the Engle-Granger can only do the comparisons between pairs, but the complexity is still quadratic without any prior filtering. Additionally, these sub-symbolic models are black boxes that lack transparency where they fail to provide readable justifications for trades. Further, the models do not incorporate qualitative domain knowledge, such as corporate structures or shared assets.

To address these limitations, this thesis adopts a hybrid machine learning approach that combines symbolic machine learning, Inductive Logic Programming (ILP) with sub-symbolic machine learning, graph embeddings and k-means clustering. The rationale for selecting ILP is the explainability and ability for relational reasoning. ILP can consume structured data to hypothesise why a pair of stocks might move together based on topological connectivity. It can generate explicit, human-readable rules rather than feature vectors of numbers.

1.2 Core Concepts

Stock markets provide a way for companies to fund their operations and for investors to own part of a business with the prospect of receiving dividends or capitalising on the growth of a business.

Stock market regulators The US Securities and Exchange Commission (SEC) plays an important role in aiming to rectify the asymmetry in access to information about individual businesses. One of their functions is to request that mandatory reports on insider trading be made public within clearly defined time limits.

The SEC reports are an example of this type of information and are regularly used by traders in their decision-making processes. However, the length and complexity of these reports make digesting the information in them far from trivial.

Machine Learning can be used to automate the analysis of textual and numerical data. In the context of analysing SEC reports, this includes the sections on insider trading, new

appointments as an officer or director and other changes in the share ownership.

Ontologies The term is closely related to *knowledge graphs*, they both represent entities and their relationships as graphs of facts. However, an ontology additionally provides a formal vocabulary and semantics (classes, properties, constraints, and inference rules) that support logical reasoning.

Ontologies support machine learning as background knowledge: they constrain hypotheses, supply features and type information, and allow for consistency checking. Learning methods can produce ontology-aligned outputs, such as the inference of class memberships, property assertions, refined class hierarchies and learning relational rules that can be exported as RDF/OWL or as Horn clauses.

Using ontologies in the context of SEC report analysis and stock market trading can provide a suitable model for the representation of structural data, such as the company management hierarchy, the main shareholders and the relationships between individuals in these roles and the companies in which they play a role.

1.3 Thesis Aims and Research Questions

The overall aim of this thesis is to develop and evaluate hybrid machine learning frameworks that automatically extract, structure and utilise data from SEC filings to improve financial forecasting, specifically pair trading strategies.

To achieve this aim, this thesis formulates and addresses three research questions:

Research Question 1: Is it possible to develop an automated procedure to extract structured information from the SEC reports that can be represented as an ontology?

I hypothesise that semi-structured XBRL formatting of SEC Forms 3,4 and 5 can be systematically parsed to accurately map the relationships between the companies and corporate insiders, such as the directors and other share-holding employees.

This question is addressed in Chapter 3, where I detail the development of a custom web crawler and extraction methodology to transform raw regulatory filings into a semantically

rich ontology, providing the primary dataset for this research.

Research Question 2: To what extent is the information captured by an ontology extracted from SEC reports relevant to stock market traders?

I hypothesise that relationship connections in the graph of the generated ontology, for example two or more companies sharing a director or senior officer, serve a strong statistically significant signal that their stock prices are cointegrated.

This question is addressed in Chapter 4, where I show that the ontology can be used to find pairs of companies that are suited to a popular trading strategy known as pair trading. I demonstrate that filtering pairs based on ontological links significantly increases the probability of finding cointegrated stocks compared to random selection. This is achieved without negatively impacting the survival time for a defined time frame.

Research Question 3: Is it possible to use a machine learning algorithm to learn from data represented as an ontology in order to search through a range of automatically formulated hypotheses and test their relevance to a financial forecasting-related application?

I hypothesise that by combining the structural relations of the ontology with sub-symbolic graph embeddings, ILP learners can automatically discover interpretable trading rules for pair selection.

This research question is first addressed in Chapter 5, where I demonstrate how a learner in first-order logic can be used to represent data in an ontology-equivalent format from which I was able to learn information potentially useful for trading. Chapter 6 describes my own contribution to the development and implementation of a learner in description logic known as P-CONNER, novel hardware-independent concurrent ILP, description logic learner, as well as how P-CONNER can be directly applied to an ontology extracted from SEC reports.

1.4 Thesis Structure

Apart from this introductory chapter, the roadmap is in Figure 1.1 and it consists of

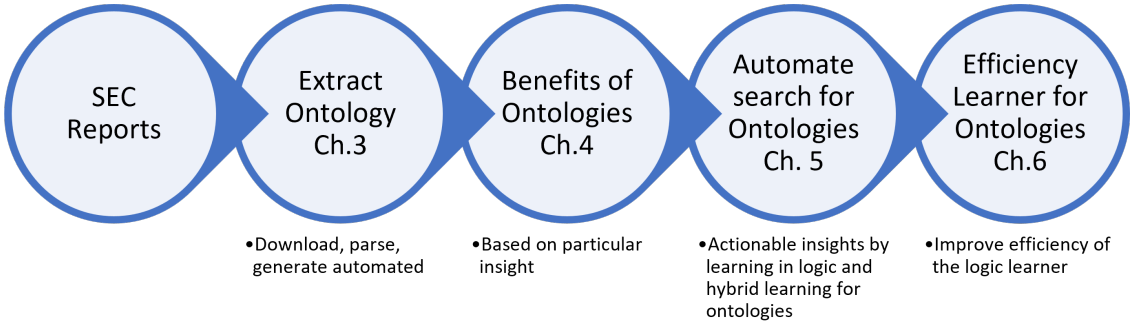

Figure 1.1: Structure and Roadmap of Thesis

- a literature review (Chapter 2),
- a chapter on extracting ontologies from SEC reports (Chapter 3) ,
- a chapter detailing pair trading using the SEC financial reports ontology (Chapter 4)
- and a chapter on learning from SEC reports using Inductive Logic Programming (Chapter 5).
- Chapter 6 describes the concurrent ontology ILP learner that I built.
- Chapter 7 summarises the main findings and sets guidance for potential future work.

1.5 Knowledge Contributions

The goal of the research is to apply hybrid learning to the financial domain using both structured and unstructured data. The first contribution would be the ontology that is built from financial reports that companies are obliged to provide. Secondly, the ontology can be used to design and define new trading strategies (Chapter 4 includes proof to test this financial ontology for trading strategies). Finally, the use of Inductive Logic Programming (ILP) is novel for hypotheses and strategies in an automated way, as shown in Chapter 5 with Progol.

The ontology created offers significant signals and strategies for pair trading that were not previously possible using handcrafted rules. To explore the relationship between these companies, forecast, and build trading strategies, ILP is used to find strategies and rules that were not initially obvious. This work also combines the node2vec embedding and

clustering to extend the background information, i.e the ontology to drive more complicated rules for making decisions for the investor.

Building a concurrent learner on description logic that can operate on ontologies to learn these rules and develop trading strategies for pair trading would be the next step. Firstly, the ILP I used requires its data to be converted to Progol backgrounds, where I could work directly with a large ontology instead.

The main contributions of this thesis, in terms of novelty and significance, are, in order of importance:

1. While ILP has been successfully applied in various domains, the application to financial forecasting has been limited by the difficulty of representing complex market data. Combination of symbolic and sub-symbolic machine learning, Inductive Logic Programming (ILP) and embeddings for a finance pair trading strategy: The combination of symbolic and sub-symbolic methods to generate and discover interpretable trading rules from the ontology that I created as data is very powerful. This system explores new rules to successfully build different trading strategies using ontology. By combining structural relations from the ontology with numerical data on cointegration and graph embeddings, marrying hybrid machine learning to reveal hidden information represents a strong and important discovery that demonstrates statistical significance for the trading strategy.

2. Utilising the Ontology for building Pair Trading Strategies: I demonstrate that Ontology can be effectively used to design and define trading strategies, specifically for pair trading. This research shows that, by leveraging the relationships identified within the ontology, it is possible to increase the probability of selecting cointegrated pairs of stocks compared to random selection. This can potentially enhance the decision-making process for choosing the right pairs of stocks while saving on compute time and resources compared to other quantitative approaches.

3. Ontology Construction from financial reports: Another novel contribution is the automated construction of a financial ontology from the structured and semi-structured XBRL SEC report documents. In addition, the time series information combined with numerical cointegration information makes the data very rich for analysis. This effectively makes it possible to query the reports and derive information such as the network of connected individuals and companies by adhering to a uniform and semantically rich representation, which would not be possible from an xml document.

The ontology dataset allows for capturing the relationships between companies and their major shareholders, offering a structured foundation for analysis. It can also be leveraged for other applications, including trading strategy development.

4. A hardware-independent concurrent Description Logic learner enabling scalable learning from ontologies: I designed and built P-CONNER, a concurrent Description Logic learner that accelerates learning using data-parallel algorithms on both CPU and GPU. I conducted experiments on benchmark tests and showed that it could learn from large ontologies much faster than traditional learners.

Chapter 2

Background

This chapter is structured to build the big picture of the research domain. Section 2.1 introduces the U.S. Securities and Exchange Commission reports, setting out the details of the text corpus. Section 2.2 reviews the financial concepts that are used in this thesis, such as, pair trading strategies and time-series cointegration tests required to execute the experiments, Section 2.3 discusses data modelling, contrasting relational databases with graph based data structures necessary for this research. Finally, Section 2.4 provides a review of Machine Learning (ML) application in finance, recognising the limitations of the existing sub-symbolic approaches and introducing the ILP and Description Logic (DL) ontologies for automated rule discovery.

This chapter discusses financial reports and their use, specifically focussing on Securities and Exchange Commission (SEC) reports, which are mandatory filings for companies in the United States in structured and semi-structured XBRL format. It also covers a number of concepts in financial forecasting and pair trading strategies, along with numerical methodologies for time series similarities. In order to understand the data contained in the reports and the data I produce from these reports, I also discuss the data and data models, such as relational and document databases; more importantly, I focus on graph data models to form the knowledge graph and ontologies. I discuss machine learning in some detail to show how symbolic and sub-symbolic techniques can be combined in this field, as in the case of ILP and node embeddings. Lastly, I discuss knowledge representation and reasoning, focussing on the structure, semantics, and inference capabilities of ontologies.

Specifically, the SEC filings provide the raw data on the corporate insider relationships, which the ontology transforms into a structured, searchable graph network. The ILP algo-

rithm then navigates this network to discover logical, interpretable rules, such as companies sharing directories and the time series data of the companies. These concepts all form part of my research and contribute to forming the 'big picture', with SEC reports as the primary data source, ontologies as a means of knowledge representation, and machine learning techniques, particularly ILP, for extracting valuable insights for financial forecasting and trading strategies.

2.1 Securities and Exchange Commission Reports

SEC is an independent agency of the United States federal government. It exists to protect investors, enforce the law against market manipulation, and facilitate capital formation [1]. During the Great Depression, the U.S. Congress passed the Securities Exchange Act of 1934, which created the SEC.

They require companies to provide several different forms of reports, primarily concerning company statements both annually and quarterly. There are also reports that indicate an official statement and a change of structure. Some types of reports are particularly interesting here, as they serve as mandatory notices of corporate insider trading, more specifically, employee trades (buys or sells) amounting to more than 10% of the company's shares. Forms 3, 4, and 5 provide the framework for reporting such insider trading information, including the title of the individual and their directorship or other senior office that they hold. Making this information public provides reassurance that all such transactions are 'above board'. This is important, actionable information that investors use to gauge the insiders' perception of their company's prospects, as whether they hold on to their stock, buy more, or sell a significant proportion of their shares is indicative of their expectations for future profits.

SEC reports were chosen because they are a highly reliable source of insider trading data which companies are legally required to publish regularly. As a text corpus, these filings are large, semi structured documents formatted in XBRL which I will mention in more detail in Section 2.1.1.

Although all three forms mentioned use the same XML schema, their purposes are slightly different. The initial filing uses Form 3, and it must be completed within ten days of becoming an officer, director, or beneficial owner. Form 4 reports changes in ownership

[1] https://www.investor.gov/introduction-investing/investing-basics/role-sec

and is due within two business days. Form 5 is used as a means of deferred reporting for overdue reports of transactions that should have been reported earlier on Form 4.

Although the SEC does not have an Application Programming Interface (API), it provides an index file that allows one to web crawl all the company data based on company name, period, or report type. I built a web crawler for SEC forms 3–5 in order to download them, and I generated ontology graphs from the downloaded data to model the 'director' relationship.

Regarding the 'raw' data format, it is XML mixed with some custom headers known as XBRL (which stands for eXtensible Business Reporting Language), a global standard for exchanging business information, particularly financial data, for which a formal specification is provided by the XBRL International standards organisation; https://www.xbrl.org/.

2.1.1 eXtensible Business Reporting Language

XBRL is a format for exchanging business information. XBRL facilitates business reporting with the expression of semantics. Common usage in many countries that make use of XBRL includes regulators of stock exchanges and securities, banking regulators, business registrars, revenue reporting and tax-filing agencies, and national statistical agencies, e.g. the Securities and Exchange Commission (SEC), the United Kingdom's HM Revenue and Customs (HMRC), and Singapore's Accounting and Corporate Regulatory Authority (ACRA) and the Ministry of Corporate Affairs (MCA) of India.

It is based on XML and many companies and reports use the XML format; however, it also supports reports in JSON and CSV formats. XBRL is mainly used for the exchange of financial information, such as quarterly financial reports. The XBRL standard is developed and published by XBRL International, Inc. (XII). [6]

XBRL communications include metadata set out in taxonomies, which capture the definitions of individual reporting concepts, the relationships between concepts, and other semantic meanings.

2.1.1.1 XBRL Specification

In typical usage, XBRL consists of an XBRL instance containing the business facts being reported and a collection of taxonomies (called a Discoverable Taxonomy Set (DTS)), which define metadata about these facts, such as what the facts mean and how they relate to one another. XBRL adopts (makes use of) the standards common for all XML in addition to XML Schema, XLink, and XPointer. The XBRL instance begins with the <xbrl> root element. There may be more than one XBRL instance embedded in a larger XML document. An XBRL instance is also known as an XBRL file. The XBRL instance itself holds the following information:

Business Facts These can be divided into two categories:

- Items are facts that hold a single value. They are represented by a single XML element with the value as its content.
- Tuples are facts that hold multiple values. They are represented by a single XML element containing nested Items or Tuples.

Context In the design of XBRL, all 'Item' facts must be assigned a context. Contexts define the entity, e.g. company or individual, to which the fact applies, the period of time the fact is relevant, and an optional scenario.

Date and time information appearing in the period element must conform to the date standard. Scenarios provide further contextual information about the facts, such as whether the business values reported are actual, projected, or budgeted.

Units These define the units used by numeric or fractional facts within the document, such as USD and shares. XBRL allows more complex units to be defined if necessary. Facts of a monetary nature must use a unit conforming to monetary standards.

Footnotes These utilise XLink to associate one or more facts with certain content. References to XBRL taxonomies, typically through schema references. It is also possible to link directly to a linkbase.

Here follows an example of a company's XBRL file :

```
 <SEC-DOCUMENT>0000950138-17-000142.txt : 20170209
<SEC-HEADER>0000950138-17-000142.hdr.sgml : 20170209
<ACCEPTANCE-DATETIME>20170209153216
...
ISSUER:

 COMPANY DATA:
  COMPANY CONFORMED NAME:   XX
  ,,,

REPORTING-OWNER:

 OWNER DATA:
  COMPANY CONFORMED NAME:   YY

 FILING VALUES:
  FORM TYPE:  4
</SEC-HEADER>
<DOCUMENT>
<TYPE>4
<SEQUENCE>1
<FILENAME>edgar.xml
<DESCRIPTION>PRIMARY DOCUMENT
<TEXT>
<XML>
<?xml version="1.0"?>
<ownershipDocument>
    <schemaVersion>X0306</schemaVersion>

    <documentType>4</documentType>

    <periodOfReport>2017-02-07</periodOfReport>

    <issuer>
        <issuerCik>0000032604</issuerCik>
        <issuerName>EMERSON ELECTRIC CO</issuerName>
        <issuerTradingSymbol>EMR</issuerTradingSymbol>
    </issuer>
```

```
    <reportingOwner>
        <reportingOwnerId>
            <rptOwnerCik>SS</rptOwnerCik>
            <rptOwnerName>PP</rptOwnerName>
        </reportingOwnerId>
,,,
        </reportingOwnerAddress>
        <reportingOwnerRelationship>
            <isDirector>1</isDirector>
            <isOfficer>0</isOfficer>
            <isTenPercentOwner>0</isTenPercentOwner>
            <isOther>0</isOther>
        </reportingOwnerRelationship>
    </reportingOwner>
    </,,,
```

2.1.1.2 SEC Filings

Within SEC reports, there are several different 'forms' available that companies need to provide information on at various intervals. In order to create my financial ontology, I used forms 3, 4 and 5 that mainly deal with the beneficial ownership of securities [7]. In this report, every employee or director who buys or sells any assets of the company needs to be reported. It also reveals an interesting aspect that is not otherwise available; that is, an employee or director may be present in one or more companies. When I combine this data and build relationships between the company and the employees. One can see that the network is connected. The network is derived from the same individuals that work in multiple companies. The network is represented with companies and individuals as nodes connected by edges.

See Figure 2.1 for a PDF formatted and rendered example of Form 4. As described before, all submissions are in XBRL format, but companies optionally also provide formatted printable formats for their filings.

FORM 4

☐ Check this box if no longer subject to Section 16. Form 4 or Form 5 obligations may continue. *See* Instruction 1(b).

☐ Check this box to indicate that a transaction was made pursuant to a contract, instruction or written plan that is intended to satisfy the affirmative defense conditions of Rule 10b5-1(c). See Instruction 10.

OMB APPROVAL
OMB Number: 3235-0287
Estimated average burden hours per response... 0.5

UNITED STATES SECURITIES AND EXCHANGE COMMISSION
Washington, D.C. 20549

STATEMENT OF CHANGES IN BENEFICIAL OWNERSHIP OF SECURITIES

Filed pursuant to Section 16(a) of the Securities Exchange Act of 1934 or Section 30(h) of the Investment Company Act of 1940

1. Name and Address of Reporting Person *	2. Issuer Name **and** Ticker or Trading Symbol	5. Relationship of Reporting Person(s) to Issuer (Check all applicable)
KONDO CHRIS	**Apple Inc. [AAPL]**	____ Director ____ 10% Owner
(Last) (First) (Middle)	3. Date of Earliest Transaction (MM/DD/YYYY)	_X_ Officer (give title below) ____ Other (specify below)
ONE APPLE PARK WAY	**5/12/2025**	**Principal Accounting Officer**
(Street)	4. If Amendment, Date Original Filed (MM/DD/YYYY)	6. Individual or Joint/Group Filing (Check Applicable Line)
CUPERTINO, CA 95014		_X_ Form filed by One Reporting Person
(City) (State) (Zip)		___ Form filed by More than One Reporting Person

Table I - Non-Derivative Securities Acquired, Disposed of, or Beneficially Owned

1.Title of Security (Instr. 3)	2. Trans. Date	2A. Deemed Execution Date, if any	3. Trans. Code (Instr. 8)		4. Securities Acquired (A) or Disposed of (D) (Instr. 3, 4 and 5)			5. Amount of Securities Beneficially Owned Following Reported Transaction(s) (Instr. 3 and 4)	6. Ownership Form: Direct (D) or Indirect (I) (Instr. 4)	7. Nature of Indirect Beneficial Ownership (Instr. 4)
			Code	V	Amount	(A) or (D)	Price			
Common Stock	**5/12/2025**		**S**		**4,486**	**D**	**$208.1933**	**15,533**	**D**	

Table II - Derivative Securities Beneficially Owned (*e.g.*, puts, calls, warrants, options, convertible securities)

1. Title of Derivate Security (Instr. 3)	2. Conversion or Exercise Price of Derivative Security	3. Trans. Date	3A. Deemed Execution Date, if any	4. Trans. Code (Instr. 8)		5. Number of Derivative Securities Acquired (A) or Disposed of (D) (Instr. 3, 4 and 5)		6. Date Exercisable and Expiration Date		7. Title and Amount of Securities Underlying Derivative Security (Instr. 3 and 4)		8. Price of Derivative Security (Instr. 5)	9. Number of derivative Securities Beneficially Owned Following Reported Transaction(s) (Instr. 4)	10. Ownership Form of Derivative Security: Direct (D) or Indirect (I) (Instr. 4)	11. Nature of Indirect Beneficial Ownership (Instr. 4)
				Code	V	(A)	(D)	Date Exercisable	Expiration Date	Title	Amount or Number of Shares				

Explanation of Responses:

Reporting Owners

Reporting Owner Name / Address	Relationships			
	Director	10% Owner	Officer	Other
KONDO CHRIS **ONE APPLE PARK WAY** **CUPERTINO, CA 95014**			**Principal Accounting Officer**	

Signatures

/s/ Sam Whittington, Attorney-in-Fact for Chris Kondo — **5/14/2025**

** Signature of Reporting Person — Date

Reminder: Report on a separate line for each class of securities beneficially owned directly or indirectly.

* If the form is filed by more than one reporting person, *see* Instruction 4(b)(v).

** Intentional misstatements or omissions of facts constitute Federal Criminal Violations. *See* 18 U.S.C. 1001 and 15 U.S.C. 78ff(a).

Note: File three copies of this Form, one of which must be manually signed. If space is insufficient, *see* Instruction 6 for procedure.

Persons who respond to the collection of information contained in this form are not required to respond unless the form displays a currently valid OMB control number.

Figure 2.1: SEC Report Rendered as Form

2.2 Financial Forecasting

Financial forecasting is a broad discipline that includes the task of predicting future movements in the price and volume of trade of company stocks and financial products. This information can then be incorporated into profitable trading strategies. One such strategy that does not depend on economic cycles and current market trends (bullish or bearish) is pair trading. Economic cycles are defined as the long-term effects of the overall economy between the periods of growth and recession. The current market trends describe the direction of the stock market at a given time, such as rising or falling prices.

Maintaining this optimal proportion of each company as their prices fluctuate leads to selling the one on the rise and buying the one that is going down in relative terms. As the long-term average is expected to remain steady, this indicates that a price trend reversal will happen over time, resulting in opposite trades being made. An investor using such a strategy is guaranteed to make a profit with each trade, provided that the important property of a stationary long-term average continues to hold.

SEC reports provide insights and valuable information regarding the company's financial activities, as well as individuals' decisions to buy or sell their assets. It can also be used for financial forecasting, predicting future movements in price and trading volumes of company stocks and financial products. It can provide information about the company's financial health, performance, and market conditions to help make investment decisions. It is also an interesting resource for academic research to analyse the behaviours of companies and markets. One can build a system of indicators or signals to drive the trading strategy (algorithmic trading) in addition to their existing models. Credit agencies also utilise the SEC reports to assess the company's performance before granting credit approval. Companies also look for their competitors' reports to get insights into their close rivals, which may affect their decision to compete or to enter another market. It can also be used for monitoring irregular accounting and fraud.

For all those reasons, SEC reports may also be toned down by the company itself, mainly not to provide too much insider information and may also present a slightly better picture than exists. This makes analysing the reports more challenging, as they are not objective or independent of the company's conflicting interests. However, it is still a valuable source of information, as it is a legal requirement to publish correct information, especially in accounting, financial reporting, and trading activities.

These diverse applications underscore the significance of SEC reports in financial forecasting and decision-making across various sectors of the financial industry.

2.2.1 Financial Markets

Financial markets [8] are interesting to many people as a medium to drive money and assets. However, they are not predictable by nature and are heavily dependent on national economies and the behavioural economics of the people trading in them. Since the early days of its existence, it has been full of stories in which people become really wealthy, often becoming very poor by losing all their money. Bubbles, since the early days, have been challenging for individuals and economies, such as the Dutch tulip bubble (Tulip Mania, 1634–1637) [9], a brief period in the Dutch Golden Age when contract prices for some tulip bulbs reached extraordinarily high levels and then collapsed dramatically.

In one form or another, whether with money or tulips, stock markets have existed since the beginning of the 17^{th} century; for example, the Amsterdam Stock Exchange opened in 1602. Over the years, the financial markets have increased levels of control and oversight and added more regulations to protect people, companies and governments from instability. There are more financial tools [10] available than ever before, such as micro-trading, options, futures and complex instruments indexed to currency, some of which also provide instant access to different markets.

There are several stock markets managed by different stock exchanges that group different indexes or companies. There is a stock market for houses, technology companies, commodities, etc. There are regulatory bodies, such as the Securities and Exchange Commission (SEC) in the US, that oversee the markets and enforce rules.

In this research, I am focusing on the SEC reports of a period for which the time series data is freely available. It captures a wide range of companies in different sectors, including both large companies and small companies that share common directors and executives. Some companies share the same individuals as their executives. I wanted to understand if there could be a pairs trading strategy where the stocks move together and predict market behaviour using symbolic AI, mainly inductive logic programming.

Predicting the stock market is a complex task that we often cannot comprehend, as it is dependent on millions of related or sometimes unrelated events and unknown factors. In this research, I am making some assumptions about how the market behaves with insider

trading.

1. Markets are not just random events.
2. Past performance with the same management team, may reflect future performance.
3. Managers have a huge influence for the company that they operate.

2.2.2 Pair Trading

Pair trading is a market neutral trading strategy enabling traders to profit from any market conditions [11]. It is a strategy that can be seen as statistical arbitrage and convergence trading. The strategy involves capturing two correlated stocks; when the correlation weakens, one goes higher, and the other goes down. The trader would be short on the higher stock and long on the lowering stock, which provides the arbitrage window until they get closer to their long-term average prices.

Pairs of companies are considered to be good for pair trading if they are cointegrated. Cointegrated pairs are those whose weighted average is stationary for some values of the weights. This has been tested with a statistical test. The level of confidence for that test is the coefficient of cointegration. There are also other techniques in time series analysis, such as Ornstein-Uhlenbeck models [12], autoregressive moving average models, and error correction models.

It is possible to model the behaviour of the stock pairs, although it is difficult to choose the right pair [13]. Traditionally, pair trading has been modelled using purely quantitative statistical tests and sub-symbolic machine learning algorithms such as Neural Networks [14] or Support Vector Machines [15] both of which rely on numerical price movement data only. This research fundamentally differs by transforming qualitative SEC insider trading reports into a structural financial ontology using ILP to discover interpretable trading rules. Then one can find the cointegration coefficient between two stocks and monitor them for changes. For instance, for a pool of only 5,000 companies, there exist 12,497,500 possible unique pairs. This number increases significantly, and the complexity rises when you increase the number of companies. For instance, for 10,000 companies, we are looking at 49,995,000 pairs. The complexity is quadratic, $O(n^2)$, which is a non-trivial amount of computation.

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \tag{2.1}$$

For a pair of companies :

$$\binom{n}{2} = \frac{n \times (n-1)}{2} \tag{2.2}$$

Correlated assets are the assets that tend to move together with statistical significance. There are various reasons for that, but the key one is that each company or government does not exist in an isolated world. They work together or compete with similar products or economies. Some assets are correlated because they are in the same sector or work with the same manufacturers; if a manufacturer is having problems, it will affect these competing companies in the same way.

2.2.3 Time Series

In order to build trading strategies and test if the right pairs have been selected, I also gathered time series [16] information. This information has historically been available for free; however, any real-time information is a specialised and commercial service.

2.2.3.1 Cointegration

Cointegration is used for modelling the long term relationship [17] of a time series. It can be applied to financial time series. If two or more time series are individually integrated, but some linear combination of them has a lower order of integration, then the series are said to be cointegrated.

When two or more time series are cointegrated, it means that, despite being non-stationary individually, they move together over time in such a way that their linear combination results in a stationary series. This property is particularly useful in financial markets, where it can indicate that the prices of certain assets are linked in the long term, allowing for the development of trading strategies that exploit these relationships. By detecting cointegration, analysts can better understand the underlying dynamics between the time series, making it a valuable tool for forecasting and risk management.

Y_t and X_t are the time series being analysed. If x_t and y_t both have an order of integration

d=1 and are cointegrated, then a linear combination of them must be stationary for some value of β and u_t. The mathematical representation of cointegration can be expressed as:

$$Y_t - \beta X_t = u_t$$

where β is the cointegration coefficient.

Cointegration is used to find relationships and is considered a multi-variate time series model. It is used for the analysis of at least two series.

Let Pt and Qt be random walks [18]. The linear combination of Pt - cQt [1] may not be a random walk. If that is true, then Pt - cQt is forecastable and Pt and Qt are said to be cointegrated. For example, a dog on a leash, let's assume Pt is the owner, and Qt is the dog, both series may look like a random walk once plotted [19]. However, the difference which is the distance between them, will look mean reverting. If the dog goes far ahead, it gets pulled back, or the owner may catch up. If the dog falls behind, the owner may wait, or it gets pulled. As a result, Pt and Qt are cointegrated. It can be seen that there are different series types that are cointegrated. From the industry point of view, this can be corn and sugar, bitcoin and ethereum. It can also be seen on rival companies, like Pepsi and Coca-Cola, Apple and Microsoft. Most of the time, they might fall apart and not follow each other due to different strategies, or different management. However, what if there is a relationship with the companies that have the same executives or co-managing a company with another company? Surprisingly, managing multiple companies for the top executives is not uncommon, from large corporations (Starbucks shares Satya with Microsoft), or companies that are not well known.

Spread represents the deviation from the long-term equilibrium relationship established by cointegration. The mathematical representation of spread is:

$$\text{Spread} = Y_t - \beta X_t$$

where β is the cointegration coefficient.

Using historical spread data (entry strategy), if the spread is greater than the mean, the trader can execute a short position by selling the overvalued stock and buying the undervalued stock. This is to profit from the expected reversion of the spread back to the

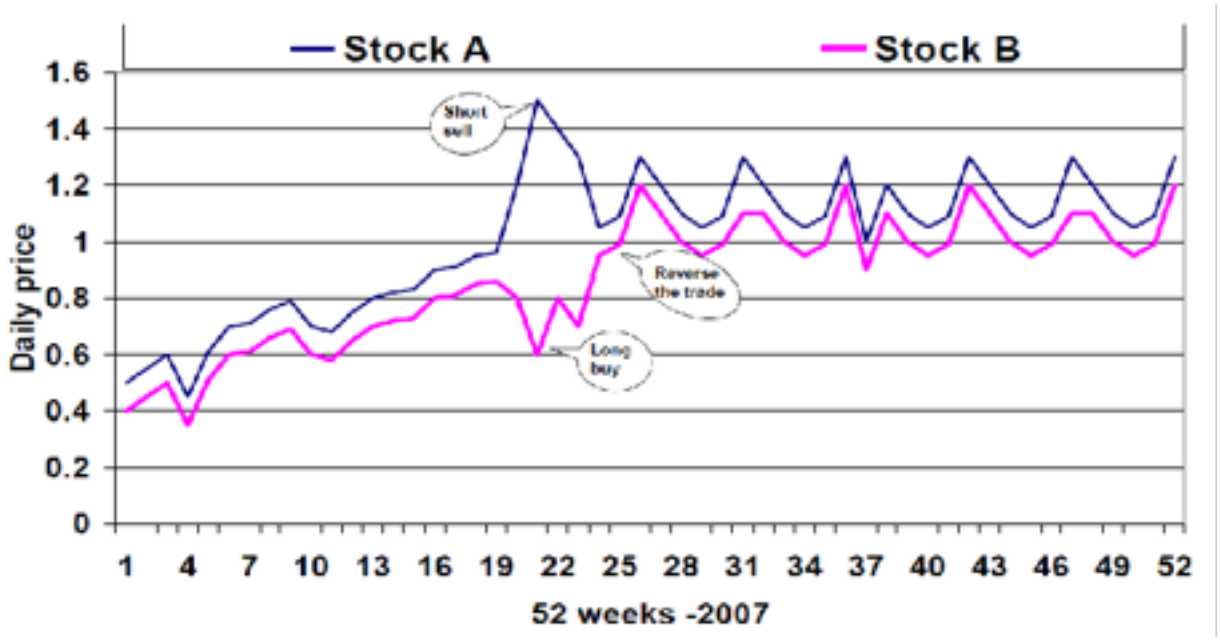

Figure 2.2: Cointegrated Stocks

mean.

I will set out below some of the examples of short sell, long buy and closing the position, which are also illustrated in Figure 2.2.

Example: Sell 100 shares of AlphaTech (AT) at £105 each, receiving £10,500. Buy 100 shares of BetaTech (BT) at £50 each, spending £5,000.

Or, if the spread is less than the mean, the trader can execute a long position by buying the undervalued stock and selling the overvalued stock. This is to profit from the expected reversion of the spread back to the mean.

Example: Buy 100 shares of AlphaTech (AT) at £100 each, spending £10,000. Sell 100 shares of BetaTech (BT) at £50 each, receiving £5,000.

When the spread is close to the mean (exit strategy), the trader can close the position by buying back the overvalued stock and selling the undervalued stock. This is to profit from the expected reversion of the spread back to the mean.

Example: Buy back the 100 shares of AlphaTech (AT) at £100 each, spending £10,000. Sell the 100 shares of BetaTech (BT) at £50 each, receiving £5,000.

Historically, this strategy was not widely known but has now been implemented in a number

of trading tools. The suitability for pair trading of two companies is determined through a numerical test of *cointegration* [20]. As the test is computationally costly, and the number of tests needed grows as $O(n^2)$ with the number of companies in hand, there is a lot to be gained if the number of candidate pairs could be reduced.

In order to be considered cointegrated, the pairs had to pass both the Engle-Granger [21] and Johansen's [22] cointegration tests.

More details on cointegration and application can be found in Chapter 4 .

2.2.3.2 Programming Cointegration

There are several libraries implementing a cointegration test. The Augmented Engle-Granger (AEG) test from the `statsmodels` Python package [23] is a good implementation and can be can be applied to the time series data. In other words, given two series Pt and Qt, we will:

1. Regress Pt on Qt, and get slope c
2. Run Augmented Engle-Granger test on Pt - cQt to test for random walk.

The code snippet implementing this is:

```
from statsmodels.tsa.statools import coint
coint(P,Q)
```

The Augmented Dickey-Fuller test can be used for the same purpose:

```
from statsmodels.tsa.stattools import adfuller
```

Coint function returns a score and p-value of the cointegration test tuples. This tells us whether the spread between the two time series is stationary.

I require a 95% level of confidence in all experiments, i.e. a p-value ≤ 0.05 to assume the spread is stationary or, in other words, that a pair of stocks is cointegrated. Making the level of confidence threshold more stringent, e.g. $p \leq 0.01$ results in very few pairs passing the test, which is why $p \leq 0.05$ was chosen.

While looking for suitable candidates for pair trading, the simplest approach would be to enumerate all possible combinations of stocks or, if this proves too demanding, simply to produce random pairs and test them. For example, given one dataset of 502 companies, the total number of possible pairs is:

Combination(502,2) = 125,751 sets of data

This is a large number to process by a human analyst, and as it grows as $O(N^2)$, the challenge of applying the test to all pairs may become significant even when the process is automated.

To test for cointegration, I need a matched pair of time series showing the same time period, that is, start and end day, and the same number of days in between. I also considered other techniques in time-series analysis, such as Ornstein–Uhlenbeck models [12], [24], autoregressive–moving-average (ARMA) models [25], and error-correction models (ECM) [26].

Ornstein-Uhlenbeck (OU) models use a continuous-time stochastic process to mathematically describe mean-reverting behaviour of the time series. In the context of pair trading, the OU model captures the spread between two assets.

Autoregressive–moving-average (ARMA) models provide statistical description of a stochastic process. It combines autoregression with moving averages. ARMA models are effective for forecasting future trajectory of a spread. However, they require the time series to already be stationary.

Error-correction models (ECM) extend the analysis by estimating the specific speed at which a dependent variable returns to its equilibrium after a short-term market shock. ECM provides actionable insights, however similarly to ARMA models, it requires the cointegration relationship to be established in order to to be applied.

In this research, the primary objective is to perform binary classification, efficiently to determine whether a pair of stocks is cointegrated or not. This serves as the ground truth for my ILP learner. The AEG two-step method and Johansen's test satisfied this requirement.

2.3 Data and Data Models

Data and data models are one of the important decisions in software development. Each data model has different characteristics. Many people are familiar with the relational data model and document models; however as my research focusses on ontologies, I will focus more on the graph data models.

2.3.1 Relational Databases - Document Databases

Relational databases [27] are the most popular type of databases mainly due to their long history, successful applications, and easy-to-use and build applications. Relational databases consist of tables, rows, and columns, linking/related tables to model the data. The retrieval of data is very powerful and easy in relational databases, albeit they will require the data to be tuned.

Document databases have become popular this century, mainly by big tech companies needing to scale and store schema-free data from disparate sources. Document databases with a flexible structure allow storing any text-based document whether that is JSON, XML, or XBRL. There are specialised databases for each format, although JSON is more popular due to the ease of use with web applications and JavaScript. The querying of data is more challenging and often not standard, like relational databases, in document databases.

2.3.1.1 Query Language - SQL

Structured Query Language (SQL) is a standard language for storing, manipulating, and retrieving data in relational databases; some document databases also adopt SQL as a query language due to its popularity and years of knowledge that people have. It is a declarative language, meaning it is a set of instructions that the database system will execute.

2.3.2 Graph data models

Graphs contain two types, vertices and edges. Existing methods are good for tree-like structures with one-to-many relationships or in documents. Although trees are also graphs, not all graphs are trees. In order to represent graphs in the data, existing data models are often not sufficient. Modelling many-to-many relationships requires expensive join operations, which degrade computational performance.

Some graph structures include property lists, trees, social graphs, semantic web graphs, and tube networks. Known algorithms can operate on graphs, and query languages can abstract away the algorithm in the search process [28].

2.3.2.1 Property Graphs and Triples

When modelling graph data, there are two established patterns: the triple model which is used in semantic knowledge graphs and the property graph model.

In the triple model [29], the triplet notation is standard way of representing knowledge graph and ontologies. The data is formed by a three-part statement, (Subject, Predicate, Object). Subject represents a unique node in the graph. Predicate defines the relationship. Object can be another node, or primitive literal value. The triple examples (John, works, full-time), the subject is a node in the graph, the object is a primitive value as string.

The property graph model [30] is another way to structure the graph. This model allows nodes and edges to act as a data container. Each node is defined with a unique identifier and it lists incoming and outgoing edges and a collection of properties. While property graphs are more efficient for storing and querying complex, nested properties, they do not provide a standard data model or query language. For this research, property graphs are less suited for DL reasoning and ILP requirements.

2.4 Machine Learning

Machine Learning (ML) is a discipline that develops algorithms that improves with experience, which usually comes in the form of data from the given domain. It is a field of study that gives computers the ability to learn without being explicitly programmed. More

formally, as defined by Tom Mitchell in his book "Machine Learning" [31]. "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P"

This definition encapsulates the key aspects of machine learning. Task (T): The specific problem or application the system is designed to address. Performance measure (P): A quantifiable metric used to evaluate how well the system performs the task. Experience (E): The data or interactions from which the system learns.

Machine learning algorithms improve their performance on a given task through exposure to data, enabling them to make predictions or decisions without being explicitly programmed for every possible scenario. This data-driven approach allows ML systems to adapt and generalise to new, unseen situations.

In the context of this research, machine learning techniques will be applied to financial data and the ontology derived from SEC reports. The task (T) could be predicting cointegrated pairs of stocks, the performance measure (P) might be the accuracy of these predictions, and the experience (E) would be the historical financial data and ontological relationships extracted from SEC reports.

Several forms of machine learning are particularly useful for analysing financial data and SEC reports:

Natural Language Processing (NLP), particularly BERT (Bidirectional Encoder Representations from Transformers) for understanding and extracting information from SEC filings. Named Entity Recognition (NER) to identify companies, people, and financial terms in reports.

These ML approaches can be combined in a hybrid learning system to leverage the structured data from the financial ontology, the unstructured data from SEC reports, and the time series data from stock prices. This multi-faceted approach allows for a more comprehensive analysis of the financial domain and the potential discovery of novel trading strategies.

2.4.1 Symbolic Machine Learning

Logic programming is declarative programming based on formal logic [32]. Logic programming is a different paradigm than imperative programming, where a program consists of a series of instructions. Logic programming views it as a proof of the theory using search. Logic programming is Turing complete, which generally means it can be used as a general-purpose programming language in many different fields.

Prolog is a logic programming language for symbolic computation and reasoning. It is especially useful for tasks that involve representing facts and rules. Like SQL, Prolog is also a declarative language, which means it describes what the algorithm needs to achieve, but not necessarily the steps needed, because it relies for that on an underlying inference engine that is at the core of its interpreter. Although its specification is not hardware-dependent, it assumes a Von Neumann architecture in which execution is sequential, with the use of backtracking to answer its queries, thus simulating reasoning in a way that is harder to implement in other programming languages.

Prolog is based on Horn clauses, and it is first-order logic. It is a rule-based language, where it is possible to define rules, and it will try to match the rules with the data. If the rules are not matching with the data, it will backtrack and try to find the next rule that is matching with the data. It is a powerful language for pattern matching, and it is used for constraint satisfaction problems.

The interpreter is based on Warren's Abstract Machine (WAM) which is an execution model for Prolog, and is the standard implementation technique. Like imperative virtual machines, WAM has sequential control but also supports unification and backtracking.

One of the fundamental benefits or the most interesting properties of logic programming [33] is the mix of input and output values. In traditional programming, there usually is a function, and the function has some input values. This is true in almost all programming paradigms.

However, in logic programming, program is able to produce input or outputs depending on the queries executed by the application.

Deductive applications [34] build programs using the full specification. Full specification states the requirements and behaviour of the desired program. It can also take informal full specification to build the automatic application.

2.4.2 Sub-symbolic Machine Learning

2.4.2.1 Embeddings

An embedding is a real-valued vector that represents a data object, such as a word, sentence, node in a graph, or the entire graph [35]. One such example is to use it to represent nodes in knowledge graphs [36]. This type of embedding is generated using algorithms such as node2vec [37]. Node2vec creates embeddings that encode information about the relationships between nodes in the graph. These embeddings can then be used to cluster similar nodes together. For example, in a knowledge graph of companies and their managers, node2vec might be used to cluster together companies in the same industry or with similar management structures.

Embeddings represent items as numerical vectors, enabling the use of distance metrics. These metrics, such as Euclidean distance or cosine similarity, quantify the similarity between the vectors. Items with similar embeddings will be closer together in the vector space, while dissimilar items will be farther apart. This capability is particularly valuable for knowledge graphs, where nodes representing entities can be embedded into a vector space. By calculating the distance between these embeddings, one can identify similar entities. For example, in a knowledge graph of companies and their managers, companies with similar management structures or operating in the same industry might be clustered based on the distance between their embeddings.

By representing items as embeddings, it can leverage mathematical tools to measure their similarity, uncovering hidden connections and patterns within the data, particularly in the context of knowledge graphs.

2.4.2.2 Node2vec Embeddings

Graph Representation Learning is a relatively new field which allows one to apply existing machine learning algorithms, such as Convolutional Neural Networks (CNN) [38] and Recurrent Neural Networks (RNN) [39] on graph data. In order to achieve that, the graph needs to be translated to the vector space and represented as a tensor, a generalisation of vectors and matrices to potentially higher dimensions (0D, 1D, 2D, 3D, and higher). The data is being converted from nodes and edges of graph information to semantic vector representations. Node2vec is one such algorithm generating real-valued vector represen-

tations of nodes on a graph [37]. Known as embeddings, such representations make it possible to define similarity between any pair of nodes, and therefore permit the use of a range of unsupervised machine learning approaches, such as clustering.The cluster number can then be used as a label on the node, which is obtained in an unsupervised way, but can be used as background knowledge to an ILP-based supervised learning. The result is a hybrid learning approach resembling some of the earlier work [40] where they combined Genetic Algorithm Search with ILP in order to achieve more accuracy and meaning in segmentations.

2.4.3 Knowledge Representation and Reasoning

Web-scale knowledge bases (KBs) provide a structured representation of world knowledge, with projects such as DBpedia [41], Freebase [42] or the Google Knowledge Vault [43]. They enable a wide range of applications such as recommender systems, question answering, or automated personal agents.

Resource Description Framework (RDF) is a graph data model created by Tim Berners-Lee, Jim Hendler, and Ora Lassila at W3C standard [44], the same organisation that created and maintains the World Wide Web. It is designed to be open, shareable, and standard. Initially, it was created to be the semantic web; however, it is essentially a standard to define how the data is stored with type information, and how the connections and relations of the data have been formed. In essence, it is a publication model for the web, this model also brings its own language called SPARQL (SPARQL Protocol and RDF Query Language). It is a query and manipulation language for the RDF systems. It is powerful, recursive, and expressive to explore the relations of the data.

SPARQL : It is a query language developed to query RDF datasets.

Sample SPARQL query :

```
Select ?student ?department ?course {
?student rdf:type Student
?department rdf:type Department
?course rdf:type Course.
}
```

```
SELECT distinct ?t1
WHERE {
?company <http://york.ac.uk/tradingsymbol> ?t1 .
 }
```

RDF and SPARQL are the way the graph structure is defined. It is not only one way or standard to define a graph database, but it is the only W3C standard; there is also property graph; however, I am focusing on SPARQL and RDF because they are explicit and definite. Being W3C is very important, as it is an open standard on how to publish data, so that it can be reused in an open protocol, SPARQL federation, and linked open data. Property graphs do not provide a standard for governing the data model and query language.

RDF is a triple store. It is identity, key, and value. This type information is available on key as metadata. The type can be a complex or a simple type. Simple type is integer, string, double, and complex type can be a reference to another RDF entity, to represent the object, like an address. It is different from relational databases and document-based systems. Relational databases are more data storage, where RDF excels at data discovery. In relational, the operation is on the physical data model, while RDF is a logical model. The advantage of the relational model is also a disadvantage at the same time, especially when modelling complex information systems with hundreds of concepts. Document-based systems focus more on unstructured data without schema constraints. However, they all solve different problems.

RDF databases are used for representing complex metadata, reference, and master data. With RDF you get the rich semantics, rich interlinked set of data, and inference. To illustrate, the concept of 'A is the same as B' is formalised in semantic web standards using constructs such as owl:sameAs [45].

It is important to note each model representation's strengths, mainly triplets, tables and documents, or knowledge graph databases, relational databases or document databases. Although it is possible to store a table in a graph representation, it will get complicated quickly and will become a problem to ingest or consume relational data easily. Similarly, storing documents in a graph is possible, but one has to go through structuring the data to fit into the graph, which defeats the point of discovery of unstructured data. One needs to use the right model for the right problem.

For instance, I have the data of addresses of people; in a relational database, it is possible to query for customers that have a certain postcode. In graph databases, this can be extended

to query the customers who have a postcode that is in a village, and the population of the village is less than 1000.

2.4.3.1 Reasoners

Ontology learning is a use case for concept learning. Concept Learning in Description Logic and the Web Ontology Language (OWL) has the goal of learning schema axioms, such as definitions of classes from existing ontologies and instance data. Most methods in this area are based on Inductive Logic Programming methods.

The problem of reaching a conclusion based on evidence and reasoning is called inference. Several inference engines and reasoners are available in Protégé, some selected ones are DReW [46], ELK [47], FaCT++ [48], HermiT [49].

Drew is an inference engine for DL programmes over description logic. ELK is a reasoner for ontologies that supports the OWL EL ontology language. Fact++ is a tableaux reasoner for OWL 2. HermiT is an OWL 2 DL reasoner.

2.4.4 Knowledge Graph - Ontologies

Knowledge graphs represent structured and semi-structured (combined with text) knowledge, integrating data from diverse sources. It can define entities, properties, and relationships in the data [50]. Unlike vanilla graph databases, knowledge graphs bring semantic foundations and data inference which allow reasoning and discovering new knowledge. There is a schema or ontology to formalise the semantics, types and relationships, forming a graph of interconnected information that supports complex queries.

Knowledge representation is essential for AI systems to perform complex reasoning tasks, such as inferential reasoning. Inference involves drawing new conclusions from existing knowledge. The traditional approach to reasoning relies on symbolic methods and mathematical proof theory [51]. This approach, however, faces challenges when dealing with incomplete, conflicting, or uncertain data. This also relates to SEC data used in the thesis. To overcome these challenges, a reasoner can be utilised to ensure logical consistency.

Description Logic (DL), a subset of First-Order Logic, plays a crucial role in knowledge representation by defining concepts, individuals, and roles, forming the basis for creating

ontologies [52]. The structure of DL, encompassing individuals as objects, concepts as subsets of individuals, and roles as binary relations, enables a structured representation of knowledge. Ontologies, a critical aspect of knowledge representation, provide a structured representation of knowledge, particularly useful in AI applications requiring reasoning and knowledge integration. They serve as formal representations of knowledge with well-defined semantics, employing a triple structure (subject, relation, object). The use of ontologies in AI extends to Inductive Logic Programming (ILP). ILP systems benefit from using ontologies for representing data and models. The outputs generated by ILP are valuable for explainable AI due to their inherent interpretability. Ontologies serve as the semantic foundation for knowledge graphs. Semantic is a logically defined framework that forms the meaning and relationships between concepts. DL follow some of the first-order logic paradigm; in addition, they define classes, relationships, and individuals within a domain. It also supports formal constraints for the values of these concepts. DL enables a knowledge graph to follow semantics with definitions and relationships. By leveraging an ontology with DL, a knowledge graph can become a machine learning system that can derive new knowledge with logical (semantic) structure and provide actionable knowledge.

Ontology is a way of storing data with relations in a graph structure [53]. It is different from relational or NoSQL databases. While relational databases rely on tables that require expensive operations to traverse deep, many-to-many relationships, NoSQL databases store disconnected, schema-free documents. Ontologies inherently link data as a semantic graph where traverse operations are native defined operation. The storage unit is formed of triplets: subject, predicate, and object. These combine to create a directed ontology. Entities are shown by URIs to construct relationships between triples.

Ontologies are formal representations of knowledge. In order to build a formal ontology, it is required to have specialised domain knowledge. Ontology learning is the process of automating the generation of ontologies. Ontologies can be simple dictionaries, complex structured web knowledge bases, or linked entities that can be queried. Ontologies provide a shared schema that can be used to access or generate data.

Ontologies consist of a domain model; a domain model is formal, explicit, and shared. It is used for interoperability, meaning disparate systems communicate through the same interface to share the definitions and meanings of the vocabularies. Inference is used to obtain actionable fragments of new knowledge from existing data.

An ontology is a structure that represents objects and their relations. These objects and relations are respectively called "individuals" and "roles". An ontology can group individuals

into classes, which are also known as concepts. Concepts and roles can be organised into hierarchies. The content of an ontology can be represented by a set of Description Logic (DL) axioms, which provide the semantics for reasoners and query languages.Ontologies are formal representations of knowledge that have well-defined semantics. They are also a basic component of knowledge graphs and are readable by both humans and computer programmes.

To contrast ontology and knowledge graphs [54] : an ontology provides formal representation of knowledge with clear semantics. It uses a triple structure: subject, predicate, and object. While related, a knowledge graph is a graph database that aggregates knowledge from different domains, which includes real-world entities and their relationships, often extracted from multiple sources. It can also leverage ontologies to provide a schema and enhance its structure and knowledge.

Ontologies are frequently expressed using Description Logic (DL) [55], a family of formal knowledge representation languages. DL provides a logical foundation for ontologies, enabling precise semantic definitions and automated reasoning. This formal approach offers several advantages, including clear semantics, decidability (the ability to determine whether a statement is true or false within the ontology), robust inference capabilities, and a balance between expressive power and computational efficiency.

2.4.5 Inductive Logic Programming

ILP is a machine learning technique that leverages the expressivity of First Order Logic (FOL) for knowledge, examples and learnt models (hypotheses). ILP is a subfield of symbolic AI, and its output makes it a prime candidate for explainable AI. ILP systems are a powerful paradigm for machine learning that uses background knowledge to produce theories expressed in logic [56]. Most ILP learners use the Horn clause subset of FOL (e.g. Golem [57], FOIL [58], Progol [59], Aleph [60], FOIDL [61]). All of these tools face significant challenges when scaling up because of the computational complexity of searching through the hypothesis space implied by the Horn clause representation.

ILP combines machine learning with logic programming. It produces first-order logic theories based on examples. ILP systems are used in various problem domains, such as identifying protein structures and natural language processing and others.

Unlike other machine learning approaches that rely on feature vectors, ILP, both data and

models are represented using subsets of First-Order Logic (FOL). ILP represents data using logic programs, which consist of facts and rules.

Facts are ground atoms, representing instances of relationships between entities, parent(john, mary) expresses the fact that "john" is a parent of "mary". Rules are Horn clauses, representing general relationships between entities, uncle(X, Y) :- brother(X, Z), parent(Z, Y) defines the uncle relation based on the brother and parent relations.

ILP data typically includes background knowledge, which can define relationships like parent, sibling, or domain-specific concepts. The data also includes examples, which can be positive or negative. These examples are usually ground unit clauses of a single target predicate, representing the concept the ILP system aims to learn.

Model Representation: ILP models are also represented as logic programs, specifically as sets of Horn clauses. These clauses are learned from the provided data, aiming to correctly classify new examples.

Let us consider some implications of using Horn clauses in the representation:

- Knowledge: Horn clauses can be used for background knowledge and the training data, such as positive and negative examples of the concept to be learnt. The Closed World Assumption (CWA), under which Horn clause-based ILP systems usually operate. Any statement that is not present or implied by the background knowledge and training data is considered false.
- Hypothesis: These attempts at defining the target concept, expressed in terms of the background knowledge, are defined in the same type of logic as the input.
- Generalisation and Hypothesis Search: ILP learners perform generalisation by searching through a space of possible hypotheses. This search is conducted using refinement operators that modify existing hypotheses to make them more general (bottom-up search) or more specific (top-down search). The search is guided by how well the hypothesis covers the training examples, i.e. whether it covers as many positive examples as possible while minimising (or reducing to zero) the number of negative examples covered. For example: top-down search is used in Foil, while GOLEM uses a bottom-up search. More advanced algorithms, such as Progol and Aleph, combine both ideas to prune and then search the hypothesis space.
- Hypothesis Cover Set Computation: This process is a computationally costly com-

ponent of ILP systems that is used to determine which positive and negative training examples are covered by a given hypothesis. The result is then used to assign a score to each hypothesis in order to guide the search algorithm. When using Horn clauses, the cover set computation is based on the use of Robinson's resolution [62], an algorithm for symbolic manipulation whose complexity affects all its uses.

The use of logic provides ILP with the advantage of interpretability. The symbolic nature allows it to effectively represent structural complexity, but the search faces computational challenges, mainly due to the complexity of the hypothesis cover set computation searches.

2.4.6 Critical Review of Machine Learning in Pair Trading

Existing computational approaches to pair trading have relied on pure quantitative and statistical methods, such as distance method, cointegration approaches and stochastic models. Traditional machine learning approaches to pair trading, such as Neural Networks or Support Vector Machines, primarily utilise the black-box feature vectors derived from price movements.

The primary strength of sub-symbolic models is their ability to process large, high dimensional numerical datasets to identify complex, non-linear price patterns. While these models are effective at capturing non-linear patterns, they suffer from critical limitations in financial context.

1. Transparency : They fail to provide a readable justification for a trade making reproducibility and explanations difficult.
2. Relational Properties and Reasoning : They struggle to incorporate qualitative domain knowledge without significant feature engineering, such as graph structures within the companies, the relationship of company assets.

ILP is a solution to these limitations. ILP is chosen for its explainability, like producing rules in human-readable clauses and the ability to reason about relationships between entities. Unlike sub-symbolic methods, ILP can directly consume background knowledge to hypothesise why a pair of stocks might move together based on their connectivity. ILP is designed to learn from relational data represented in First-Order Logic. This makes

it suitable to traverse and reason complex relationship between stocks (such as linked companies and shared directors).

In order for a machine learning system, like ILP, to learn, I need to generate all possible pair combinations of stocks. Therefore, I need to have all of the SEC report data on my dataset, augmented with the cointegration information of each possible pair. I also need to separate my data from test data to validate the hypothesis that my learner system can benchmark. Some of the systems include Progol [63], Metagol [64]. where the learning process is viewed as a search problem in the space of potential hypotheses.

ILP allows learning procedures from examples, whether examples are input or output which is one of the main advantages of ILP over other machine learning methods. It uses an expressive first-order framework instead of a simple attribute-value framework [65]. It can also take background knowledge into account. General procedure can be defined for transforming inputs into outputs.

In ILP, given some input or output examples, positive or negative, it produces an explicit program that, when it is run on the inputs, produces outputs, or similarly, when it is run on the outputs, produces inputs, or a mix of the input and outputs can also be induced. This process is called symbolic program synthesis (SPC). The generated programs are readable and understandable, which is why it is considered a white box system, rather than a black box like neural networks, which makes it suitable for verifiable, certifiable applications.

1. Data-efficient, meaning they do not require millions of samples to generate a program with a good accuracy.
2. Interpretable : Used by the ILP systems, or read by people.
3. General : Meaning that it can generate programs outside the training data (reasoning).
4. Mislabelled data : Often the mislabelled data is a problem, as it is based on Horn clauses logic [66] . However, negative examples often help to generate more precise application synthesis.
5. Ambiguous data is problematic because it will break the application logic.

For that reason, ILP systems problem domain is very specific and exact. For instance, the accuracy in image processing is poor compared to other novel approaches. On the other

hand, ILP is suitable for Natural Language Processing (NLP), or bioinformatics, like drug discovery, or indeed finance.

2.4.7 Accelerating ILP Learners: Parallel Computation Approaches

A number of approaches exist in the literature for speeding up ILP learners, for example, the use of query packs [67] that avoid the repeated evaluation of overlapping parts of hypotheses and the concurrent computation. Strategies for parallelising ILP systems can be categorised based on how the work is divided. Applying parallelism is possible by learning separate target predicate classes independently in parallel, potentially using a separate copy of the dataset for each class. The search space can also be explored in parallel by distributing the task among multiple processes. One can also split the data to process each part separately in parallel, mainly for testing hypothesis coverage, with the results merged in a final step (divide and conquer).

Another way to categorise parallel evaluation approaches is by their memory model, whether they use shared or distributed memory. Ohwada and Mizoguchi [68] used a shared memory approach to combine branch-and-bound ILP searches with a parallel logic programming language. Shared memory is required so that there is no need to have to clone the data and coordinate the intermediate results of the parallel search. ILP search with parallelised hypothesis evaluation is a viable approach. This can be achieved by assigning different clauses to different processors in order to evaluate simulataneously.

2.4.8 Description Logic Learners

Description Logic (DL) is used to represent knowledge in the form of ontologies. DL uses constants (individuals), unary predicates (concepts), and binary predicates (roles).

DL provides a formalism that defines the semantics of ontologies, establishing the means to determine whether a statement can be proven true or false given the knowledge expressed in the ontology.

DL-FOIL is an adaptation of the classical ILP algorithm FOIL for DL, and APARELL is an ILP algorithm for learning ordinal relations in DL that borrows ideas from algorithms such as Progol/Aleph.

DL-Learner is another algorithm, which provides a robust interface and is a mature, modular platform. DL-Learner allows users to choose between different reasoners for handling inference, such as Konclude. It also provides a selection of refinement operators, including OCEL and CELOE, which differ depending on the specific type of DL used. It also offers facilities to handle large datasets through techniques such as statistical sampling and allows the selection of different refinement operators. The OCEL refinement operator of the DL-Learner, particularly its OCEL algorithm, uses a top-down search. OCEL is of particular interest for the reasons set out in the subsequent sections. This is combined with the following scoring function to guide the search.

The base score, denoted as 'ocel score(N)' for a search node N with hypothesis C, is calculated as follows:

$$\text{ocel score}(N) = \text{accuracy}(C) + 0.5 \cdot \text{acc\_gain}(N) - 0.02 \cdot n$$

where accuracy(C) is the accuracy of the hypothesis C, acc_gain(N) represents the increase in accuracy relative to the parent node of N, and n is an upper bound on the length of child concepts, which is set to be equal to the total number of concepts in the ontology.

2.4.8.1 Description Logic

Description Logic (DL) is a family of formal knowledge representation languages widely adopted as a foundation for ontology languages and semantic web technologies [69]. This logic is particularly suited for representing and reasoning about conceptual knowledge in a structured and semantically well-defined manner.

DL is distinguished by several key features:

- Decidability: They guarantee that reasoning tasks within their framework can be algorithmically determined, ensuring computational feasibility.

- Open World Assumption (OWA): They operate under the assumption that not all information about the world is explicitly stated, allowing for the discovery of new knowledge through inference.

Operating with formal semantics, DL provides unambiguous interpretations and supports various reasoning tasks, such as concept subsumption (determining whether one concept is a subclass of another) and consistency checking (verifying whether a set of axioms is logically consistent). These features make DL powerful tools for knowledge representation and reasoning.

In order to model any data, the fundamental building blocks of DL are established. Concepts are used for representing classes or sets of individuals that share common characteristics. Roles are to represents properties or relationships between individuals. Individuals represents specific entities within the domain.

These elements enable the construction of complex concept descriptions and relationships within a given domain. For instance, in the financial ontology developed for this research, one can combine these building blocks to formulate logical definitions. This allows the system to define a linked company as the intersection of the individual and the company.

2.4.8.2 Ontology Serialisation

RDF can be serialised in different formats. the common formats are n-triples(.nt), turtle (.ttl), JSON-LD (.json), or RDF/XML (.rdf). N-triples have some limitations for recursive graphs; thus, Turtle can address these issues; however, it also adds verbosity. For this work, due to the portability and linearity of the graph, I have decided to use n-triples. The main advantage of n-triples is that they can be compressed or sent line by line over a network. If a line is lost, then a triple is lost; in Turtle, one line can correspond to several thousand triplets. On the other hand, Turtle is more human-readable, but n-triples are easier to parse (by computers).

```
<http://sec.com/0001586842>
  <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
     <http://xmlns.com/foaf/0.1/Person> .
<http://sec.com/0001164863>
   <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
     <http://york.ac.uk/Company> .
```

I have generated and used, turtle serialised format from all the documents that have been downloaded. As it is a large amount of data, one of the other advantages of this serialisation

format, turtle, is that it can resume the operation if the download stops or the website does not respond, as it can just append to the file from where it left off, as long as it completes a full line.

2.5 Summary

This chapter has established the theoretical and technological background and techniques required for automating financial forecasting using qualitative data. There is a clear gap in existing computational approaches. While pair trading is a proven market-neutral strategy, current numerical models for pair trading ignore the underlying company and organisational structures.

SEC Forms 3,4 and 5 provide rich and relevant source of intelligence that may provide signals relevant for identifying market behaviour or trends. However, the semi-structured XBRL format of these reports makes them difficult for ingest directly using standard machine learning pipelines. To address this gap, the following chapters will detail the methodology used to systematically extract this data (Chapter 3), build it into a semantically rich ontology to filter trading pairs (Chapter 4) and leverage the explainable, relational learning of ILP and description logic to discover novel rules (Chapter 5 and Chapter 6).

Chapter 3

Extracting Ontologies from SEC Reports

In this chapter, I describe the process of extracting ontologies from financial reports. The financial reports are crucial data for financial analysis and decision making; however as in their raw form, they are not easily accessible or analysable. It discusses the data collection methodology, the parsing techniques, and the publishing and ontology development process to construct structured data in the form of an ontology.

3.1 Introduction

As discussed in Section 2.1 , SEC filings and XBRL documents provide a rich source of semi-structured data.

The extraction of structured knowledge from financial filings represents a significant challenge in the pipeline for my hybrid machine learning system.

This chapter details the implementation of an automated pipeline for converting heterogeneous SEC filings into a formal ontology. While Chapter 2 outlined the regulatory standards of XBRL (see Section 2.1.1), this chapter addresses the implementation of automating the transformation of these datasets into queryable formal ontology. The primary objective of this chapter is to detail a reproducible extraction methodology that bridges the gap between semi-structured XBRL documents and a semantic graph. I specifically address,

schema variation (handles variety in the reports), entity resolution (standardisation and taxonomy) and relational inference (mapping of corporate relationship). This stage is critical to the research methodology as it defines the search space for the ILP algorithms in Chapters 5 and 6.

3.2 Data Collection Methodology

3.2.1 The EDGAR Repository

The EDGAR (Electronic Data Gathering, Analysis, and Retrieval) system is a public repository of filings and reports that companies in the United States are required to submit to the SEC in their prescribed format. EDGAR holds data from 1995 to the present day.

At the time of writing this thesis, the U.S. Securities and Exchange Commission (SEC) does not provide an Application Programming Interface (API) for accessing their data. Instead, they offer a system called EDGAR (Electronic Data Gathering, Analysis, and Retrieval) that allows public access to corporate filings. EDGAR data is available for anyone to access at no cost. Users can browse filings or perform specific, detailed searches using EDGAR's public search features. The data can also be downloaded for further analysis or reporting purposes.

XBRL (eXtensible Business Reporting Language) is a standardised format for exchanging business information that enables the expression of semantic meaning commonly found in business reporting. XBRL is based on XML, extending its definition by focusing on business reporting taxonomies and structures. This includes the company's file registration statements, periodic reports, and other forms submitted electronically via EDGAR. This data is freely accessible to the public.

The primary purpose of extracting ontologies from SEC reports is to transform unstructured financial data into structured, machine-readable formats that can be easily analysed and interpreted.

3.2.2 XBRL Downloader

The XBRL Downloader, is an HTTP client that I developed to generate my dataset used in this research while facilitating the retrieval of all periodic reports. It includes different rules and instructions for downloading the data correctly. Upon downloading the data, we have approximately 100,000 different files for a quarter to analyse and work on, depending on the company and quarter.

It has the capability to download and parse different forms of data. Once the data is downloaded and parsed, it generates the ontology in 'turtle' format. Turtle (Terse RDF Triple Language) (Section 2.4.8.2) is a textual syntax for RDF (Resource Description Framework) that allows RDF graphs to be written in a compact and natural text form. It is designed to be more readable than XML RDF notation while still maintaining the full expressiveness of RDF. For instance, it is still possible to define recursive graphs in turtle format.

The downloader has options for the form types, namely 3, 4 and 5 (Section 2.1), for the parser that I developed. The other options are the location to save the files and the type of forms: daily or full. It has the configuration for the 'Quarter' and 'Year' in order to construct the URLs with the correct locations. Although the Http Client is a standard client, it needs to have some properties set. TLS is one of the latest security standards that come with major browsers. Only TLS 1.3 is supported by the EDGAR servers. This is normally not a problem for modern browsers; however, it might become a challenge if the http library implementation is not up to date with standards, as it will require more attention. Exchanging the certificates and keys (which happens automatically in browsers) will be necessary, and it is also problematic if it is behind a company firewall and proxies, even just to load the initial page. SEC does not allow custom clients without metadata being supplied. The following http headers need to be set as below :

```
{
    "Host": "www.sec.gov",
    "User-Agent": "Custom Crawler for research AdminContact@research.com",
    "Accept-Encoding": "gzip, deflate"
},
```

Once these properties are set, as long as the file exists, it can be accessed by constructing the correct URL. The index file plays a crucial role in identifying the files. The index file is constructed based on the options of quarter and year.

X >10K >index.xml

C1.brml

C2.brml

SEC - Accessing EDGAR Data Access

While there is no formal API, the SEC does provide a structured way to access their data through their static file server, which includes an index of all the files. The index files are updated regularly to provide access to the new reports submitted. The index file contains a relative path to each report, along with some other information, such as the quarter, company, and reporter information. The report files are formatted in XML/XBRL, PDF, JSON, and compressed formats. For this thesis, I have elected to use the XBRL format since it is structured and can be accessed using XML concepts such as querying and navigating the tree with existing libraries. Once the URL is constructed, it makes an HTTP request to download the desired report. There is also a rate limit to consider to prevent overloading their servers. The SEC allows for "fair use," which is typically interpreted as no more than 10 requests per second. during my experimentation, this limit has not been enforced, and I left the downloader experiment running for more than a week to gather 2 year's worth of data for all the filing companies. SEC also needs to have custom headers sent in the request to grant the file, including a User-Agent that identifies the purpose of the scraper and provides contact information.

While this system is not as convenient as a modern REST API, it does provide a reliable and consistent way to access SEC data programmatically. The custom web crawler I developed, as described in the previous sections, is designed to work within this framework, effectively serving as a client for this unofficial API. It is worth noting that EDGAR has expressed interest in modernising their data access systems. Future researchers may find that if there is an official API, it could significantly simplify the data collection process.

3.2.3 Directory Browsing

EDGAR allows directory browsing for specific directories related to CIK (Central Index Key) numbers and accession numbers. For instance:

- `https://www.sec.gov/Archives/edgar/data/51143/`
- `https://www.sec.gov/Archives/edgar/data/51143/000104746917001061/`

Each CIK directory contains three index files to assist with automated crawling:

- `index.html`: For browsers.
- `index.xml`: For XML clients.
- `index.json`: For JSON clients.

3.2.4 Data

The EDGAR system holds data from 1995 to the present. Every filing submitted to EDGAR is assigned a unique number. EDGAR offers vast amounts of financial data on publicly traded companies, making it an invaluable resource for researchers, analysts, and investors. Additionally, the structure of EDGAR's file system, including CIK and Accession Number directories, makes it easier to automate the retrieval of specific datasets for analysis.

For main experiments in this thesis, the dataset specifically focuses on SEC Forms 3, 4 and 5 which are the documents that deal with the ownership of securities by corporate insiders. Although the word 'insider' holds negative connotation, in this context and throughout, it is simply someone, whether they are an employee or director of the company who deals with the appointment or buying and selling of their shares. The extraction process targets the semi-structured XBRL files in the reports. The scale of this dataset can be imagined that, a single mid-size company files these reports for every appointment, issuance or sales of the shares, where a typical quarter produces approximately 100,000 individual reports that must be download, parsed and processed using the methods I described in this chapter.

The processed data includes not only the XML-based ownership documents, but also the metadata headers. These headers provide the "acceptance-datetime" and "accession-number", which are important for temporal sequencing in the ILP learners. Each report is

treated as a transaction log of relationships, where the state of a company's management hierarchy is updated incrementally. Each filing is treated as a state-transition event in the knowledge graph.

3.2.5 Downloader

The 'Download' method takes a year and an array of quarters as parameters. The method then iterates over the quarters and downloads the forms for each quarter of the given year.

For the requests, a standard http client can be used; however, only TLS 1.3 is supported. The SEC's requirement for TLS 1.3 support is primarily due to security concerns. TLS 1.3 is the latest version of the Transport Layer Security protocol, which provides enhanced security and performance compared to its predecessors. This also means that there is a need for a modern http library that supports the TLS protocol, which can be a limitation for some languages or approaches.

3.3 Data Parsing and Reproducibility

3.3.1 Deterministic URL Construction

To ensure reproducibility, the process of constructing the URLs to access the reports must be deterministic. For a given index that contains Central Index Key (CIK) and accession number, the URL is constructed using the SEC pattern below:

`https://www.sec.gov/Archives/edgar/data/[CIK]/[accession-number]/`.

To programmatically download the XBRL files, the crawler first retrieves the quarterly EDGAR master index file, such as `form.idx`. By parsing this index, the system extracts the CIK and the unique accession number for each target filing.

The constructed URL is then used to download the corresponding XBRL files in a loop until all the relevant index entries have been processed.

My parser automates this by the removal of hyphens (the filter) from the accession number to find the directory and then fetching the primary XML document defined in the index.

This allows independent verification of any extracted triples against the source filing to avoid re-downloads in case of restarts or refiling.

XML Data Extraction

Once the report is downloaded, the parsing logic starts. The FormParser class is designed to parse XBRL formatted documents, which have embedded XML with additional metadata, focussing on "ownership documents", Forms 3, 4 and 5.

The XmlXbrlExtracter class is designed to extract XML data from a file. This class assumes that the file contains one or more XML blocks, each starting with <XML> and ending with </XML>. It then extracts all lines between XML tags for each block.

It then parses the collection of XPath expressions for the fields that contribute to the structured file format. XPath (XML Path Language) is a query language for selecting nodes from an XML document. These paths are used to navigate through the XML document and extract specific pieces of information.

Subsequently, my crawler's parser generates tabular data from semi-structured data. Forms 3, 4, and 5 are semi-structured. Form 10, which is more text-based with predefined chapters, is unstructured. I also have parsers defined for Form 10, but the main focus of this data has been semi-structured data and Forms 3-5.

The key fields being indexed include:

- rootXpath: The root of the ownership document
- issuerCIKPath: Path to extract the issuer's Central Index Key (CIK)
- issuerNamePath: Path to extract the issuer's name
- issuerTradingSymbolPath: Path to extract the issuer's trading symbol
- namePath: Path to extract the reporting owner's name
- cikPath: Path to extract the reporting owner's CIK
- directorPath: Path to extract if the reporting owner is a director
- isOfficerPath: Path to extract if the reporting owner is an officer

- tenPercentPath: Path to extract if the reporting owner is a 10% owner
- otherPath: Path to extract if the reporting owner has other roles
- officerPath: Path to extract the reporting owner's officer title
- periodOfReport: Path to extract the reporting period
- documentType: Path to extract the type of document

3.4 Data Publishing and Resource Description Framework Graph Construction

Once the relevant fields (issuer CIK, owner and relationships) are extracted from XBRL, the data must be mapped into a Resource Description Framework (RDF). To ensure uniqueness, URIs have been constructed for each extracted entity from the reports.

In order to construct the RDF data, the ownership documents are being parsed. These documents contain information about a person, a company, and the shares they buy or sell from SEC reports. The information is then structured into a graph, which is a standard model for data interchange on the Web.

A base URI (http://york.ac.uk/) is combined with unique CIK to create distinct nodes. For example, a person with CIK 1234 is represented by the URI http://york.ac.uk/person/1234. The URI is then used in various nodes to create relationships between entities. Similarly companies are created using custom domain taxonomy (http://york.ac.uk/company).

The transformation of parsed data into the RDF model follows a set of rules. Nodes are created for each unique CIK URI. Predicates are constructed from the relationships with the typed values such as 'isDirector' (boolean), 'officerTitle' (string) and it is mapped to properties.

Relationships are then added to the graph between these generated URI nodes using the extracted fields from the document. If the reporting document indicates that reporting owner is a director of the company, an RDF triplet is generated where person URI is the subject, with a property (http://york.ac.uk/worksat) is the predicate and the company URI is the object.

The resulting Triplet (subject, predicate, object) is created for a given entity with the relationship.

¡http://york.ac.uk/PersonCIK¿ ¡http://york.ac.uk/isDirector¿ "true" xsd:boolean .

The parser then constructs URLs for a person, a company, and a report using the document's 'CIK' and 'URL' properties. It adds these URLs as nodes to the graph, along with their respective types (Person, Company, SECReport). It also adds to the crawling list for visiting later. It adds relationships between these nodes, such as a person working at a company and a report being filed for a person.

This mapping ensures that semi-structured report documents are standardised across the board, flattened and result in the same topology that can be queried, further enriched and suitable for description logic reasoning.

The benefit of using CIK as the component of the URI helps to stop entity duplication. If a director sits on the boards of two companies, the CIK remains the same across different filings of different companies. As a result, the crawler maps these filing to the same node in the graph, creating a person node potentially connected to multiple company nodes.

Finally these triplets are serialised into N-Triples (.nt) format where I load them to a graph database for data exploration and verification.

3.5 Contributions, Challenges and Successes

3.5.1 Knowledge Contributions

The primary contribution is ontology based extraction methodology that focusses on relationships which are optimised for symbolic AI. Unlike most of the other research and tools that target numerical data, this methodology extracts the relationships in a graph form which can show corporate insiders, their job description, role and financial contribution. This method prioritises the topological relationship between corporate entities and their insiders.

These methods provide a semantic bridge to a formal graph, which is essential for reducing ILP search space in subsequent chapters.

3.5.2 Challenges

Addressing Research Question 1 (RQ1) from Chapter 1 was found to be a non-trivial task, due to large number of reports. Evolution of the schema year on year and sometimes even quarter by quarter, forces the parser to utilise abstract templates rather than fixed paths. Some namespace changes (additions, removal or amendments) over the year makes the parser aware of the schema version for backward and forward compatibility. Extracting static relationships from dynamic filings requires the framework to handle the transactional nature of SEC reports, ensuring that the correction updates and removes the previous entries.

Evidence for the success of this procedure served as the essential dataset for the experiments in Chapter 4 and Chapter 5.

3.5.3 Evaluation and Success

The success of the ontology construction was evaluated using these criteria : Relational Fidelity, Logical Consistency and Reproducibility.

Relational Fidelity where a manual review of 100 triples conducted, showed 100% mapping accuracy. In this review, all 100 sampled triples matched their source evidence, and target rule mapping that were defined. This does not validate that a complete set has relational fidelity, but it gives confidence with evidence for the random sampling. More automated methods and a daily run of those methods could be build for further confidence.

Logical Consistency where the resulting ontology was validated using the Pellet reasoner to ensure no schema violations occurred during extraction. The populated ontology was checked for schema violations in class membership, property domains, property ranges and RDF constraints. No warnings or errors were reported by the reasoner for the generated ontology.

Reproducibility was assessed by re-running the pipeline over the input files and comparing the resulting output. Due to the rules being deterministic, the output is the same for the given input. This ensures that generated ontology is reproducible from the same corpus and implementation.

The successful automated extraction of 100,000 files into a semantically linked RDF graph

demonstrates that it is indeed possible to develop an automated procedure to structure SEC data, directly addressing Research Question 1.

3.5.4 Comparison with Existing Methods

Existing extraction tools for EDGAR (python-edgar [70] or EDGAR-Crawler [71]) are focussed on downloading the files in bulk and primarily focussing on numerical financial statements (Form 10). In contrast, my method provides a semantic bridge specifically for ownership filings (Form 3,4,5). While numerical methods are useful and produce feature vectors, my methods produce a relational graph which is uniquely suited for symbolic AI tasks mentioned in later chapters of this thesis.

The method presented in this chapter differs with the focus on relationships and ontologies. It is not intended to replace existing downloaders, parsers or tools. Existing tools are for ingestion and downloading the documents, where the contribution of this chapter is the construction of an ontology representation for the SEC reports data. The evaluation shows that this representation can be generated at scale, preserve and build the semantics, that the generated ontology is logically consistent under reasoning and the data pipeline is reproducible.

3.6 Summary

In this chapter, I have discussed the process of extracting ontologies from SEC reports. I began describing SEC reports, focusing on Form 3 reports to build the crawler and parser.

This chapter has demonstrated that an automated procedure can extract structured information from SEC reports. Directly addressing Research Question 1: Is it possible to develop an automated procedure to extract structured information from the SEC reports that can be represented as an ontology? It was possible as demonstrated in this chapter.

In conclusion, this chapter provides a comprehensive overview of the methods and techniques used to extract, parse, and publish data from SEC reports, ultimately transforming it into a structured ontology.

Chapter 4

Pair Trading with an Ontology of Financial Reports

In this chapter, I demonstrate that an ontology derived from SEC reports can be used to support the development of a stock market trading strategy, namely 'pair trading', by increasing the likelihood of selecting cointegrated stock pairs, with no negative effect on the survival time of such pairs when compared to random ones.

More specifically, I formulate a hypothesis that the information contained in the ontology is relevant to the discovery of cointegrated pairs of company stocks, and I show that the contribution of that information is statistically significant.

The objective is to enhance the selection process of cointegrated stock pairs and evaluate how these qualitative relationships impact the stability and longevity of cointegration over time.

Standard quantitative approaches for selecting cointegrated pairs relied on statistical methods, such as the Engle-Granger two step method or Johansen test [72]. More advanced techniques use Ornstein-Uhlenbeck processes for mean-reversion modelling [12]. However, these methods remain computationally expensive due to the $O(n^2)$ search space. My approach differs by utilising semantic background knowledge to prune this space before statistical testing begins.

Historically, choosing the right pairs of companies has relied heavily on quantitative data analysis, such as statistical correlations and price movements. While this approach can

identify potential pairs, it may miss important relationships by focusing solely on numerical patterns. Pure quantitative analysis may filter out meaningful connections that are not immediately apparent in the price data alone. This chapter studies the potential benefits of using additional qualitative information for pair trading. Here, I use an ontology to represent and structure information extracted from financial SEC reports in a way that is optimised for search. The mandatory SEC reports format, as in Chapter 3 , is not easy to query; for example, e.g. projections to fields or their composition can be hard to find even when using an XML store. My ontology-based approach provides uniformity of representation and the use of controlled vocabulary wherever possible. The ontology is then used to identify links between companies by finding senior employees, officers, or major shareholders in common. This information can also be potentially useful for identifying suitable pairs of stocks.

Cointegration between the stocks of two companies occurs when the price of a weighted portfolio of these stocks remains stationary. This property underlies pair trading, a potentially profitable market-neutral strategy that eliminates market exposure regardless of its direction. This chapter also studies the effect these connections have on the expected longevity of a cointegrated pair.

4.1 Introduction

Cointegration is a statistical property of time series that measures the long-term equilibrium relationship between two or more variables. Two time series are considered cointegrated if they share a common stochastic trend and if their linear combination is stationary.

For more details on cointegration, see Chapter 2

Pair trading is a market-neutral strategy that relies on identifying pairs of stocks whose price movements are statistically correlated over time. This strategy aims to capitalise on the relative price movements of the paired stocks, allowing traders to profit from the convergence of their prices.

However, simply being statistically correlated is not sufficient for a pair to be considered suitable for trading. The correlation must be strong and stable over time, indicating a consistent relationship between the two stocks. Additionally, the stocks should ideally exhibit cointegration, which means that their price movements are not only correlated but

also share a long-term equilibrium relationship. This property provides a profitable trading opportunity.

To identify suitable pairs, traders often employ various analytical and statistical techniques, such as the business models of the companies, market conditions, sector performance, and conducting cointegration tests. This is to ensure that the selected pairs are fundamentally related.

I gathered a dataset of 2804 records. Using such a dataset and to generate cointegration of all possible pairs, I would be required to write nearly 3 million calculations. These are very expensive calculations; however, it is necessary to train machine learning models and also to verify my correlation experiments.

Choosing the right pairs of companies that exhibit such behaviour is a computationally heavy task if attempted with every single possible permutation of pairs. It involves analysing vast amounts of historical price data, running statistical tests, and continuously monitoring the relationships to adapt to changing market conditions. As a result, successful pair trading requires not only statistical analysis but also a deep understanding of market dynamics and the specific characteristics of the stocks involved.

Once the pairs are selected and known to be cointegrated, the next step is to monitor the pairs for any deviations from the cointegration relationship. This is to ensure that the pairs remain cointegrated and to identify any potential trading opportunities.

Cointegration, entry and exit strategies are the three main components of pair trading used in this experiment. Although pair trading can be profitable but it is not easy to identify such pairs and not all pairs are cointegrated.

Ontologies are a way to represent and structure information that makes it possible to link entities and their properties to other entities. It is similar to relational databases but with the added benefit of a graph structure and a powerful way of querying the connections between entities, which is not possible with relational databases.

In the rest of this chapter, I explore the potential benefits of incorporating qualitative information extracted from the U.S. Securities and Exchange Commission (SEC) reports into the pair trading strategy. By constructing an ontology from these reports, I aim to represent and structure financial information that helps in the discovery of cointegrated stock pairs. My hypothesis is that companies connected through certain relationships, such as shared senior employees or major shareholders, are more likely to have cointegrated stock

prices. I test this hypothesis and demonstrate that the use of the ontology increases the probability of selecting cointegrated pairs without negatively affecting their survival time.

4.2 Research Hypothesis and Research Question

This chapter directly address Research Question 2 (RQ2) from Chapter 1 : To what extent is the information captured by an ontology extracted from SEC reports relevant to stock market traders?

To address this, I formulate the following hypothesis : The topological proximity of companies in the financial knowledge graph (defined by shared board members or significant shareholders) is a statistically significant predictor of price cointegration. I structure that the hypothesis that topological information, the connections between companies with shared employees or major shareholders present in the SEC ontology, can give a strong signal for finding cointegrated pairs. Purely quantitative approaches require significant computation power and time to test all possible stock permutations. I hypothesise that filtering candidate pairs based on ontological links will provide a significantly higher probability of cointegrated stocks than a randomly selected stock pair.

To evaluate this hypothesis, I conduct two experiments, (1) cointegrated pairs found in linked pairs versus random sets, where I examine the possible connection between the number of links and the likelihood of cointegration. (2) Survival rates of these cointegrated pairs over time, which tests stability and viability of the pairs identified by my ontology. Specifically, I show that a link between two or more companies makes it more likely that their stocks are cointegrated.

4.3 Methodology

4.3.1 Data Collection

I collected stock price data and SEC filings for publicly traded companies over the period from Q2 2017 to Q2 2018. The historical stock data was obtained using the yfinance Python module [73], which provides daily closing prices and other relevant information. it is a free service for historical data; hence, older dates are included.

4.3.2 Ontology Construction and Dataset Alignment

The ontology used in these experiments did not previously exist. It was constructed entirely from scratch using automated extraction methodology detailed in Chapter 3 . By parsing semi-structured XBRL data from SEC Forms 3, 4 and 5, I built a customised graph representing companies as nodes and shared personnel as edges.

The ontology and the stock time series were generated using 3-month period, Q2 in 2017. However, to test the stability of the cointegration (namely 'survival rate') and ensure that the results were not an anomaly for a single quarter, the same experiment methodology has been applied to the gathered additional testing window of consecutive quarters, up to Q2 2018.

An ontology was constructed by extracting information from SEC reports, focusing on identifying relationships between companies through shared personnel and major shareholders, as described in Chapter 3. The ontology represents companies as nodes and connections as edges, where edges signify the presence of shared individuals.

4.3.3 Pair Selection and Relational Logic

To test the topological hypothesis, I categorised company pairs into two sets :

- **Random Pairs:** A control group generated through a uniform random sampling of 50,000 unique permutations of companies present in the ontology.
- **Linked Pairs:** A group defined by an adjacency constraints in the knowledge graph. A pair is considered linked when at least one individual shares a connection with a different triple with the same individual.

The subsets were created for pairs with a minimum of 1 to 3 links. This allowed me to observe the correlation between relations and probability of cointegration.

Linked pairs Figure 4.1 is a subset of all possible pair populations, but is selected separately from the random set. Random set serves as a stochastic baseline to represent the cointegration in the total population, where linked set is filtered via the ontology.

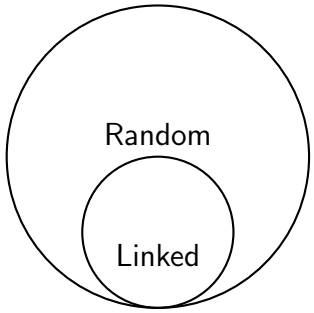

Figure 4.1: Venn diagram illustrating the selection of random and linked pairs.

4.3.4 Statistical Testing for Cointegration

A two-step method was employed to test for cointegration (see Chapter 2) between the stock prices of each pair. A pair was deemed cointegrated if the test statistic fell below a critical value at the 5% significance level.

4.3.5 Evaluation Metrics

I evaluated the effectiveness of using the ontology by comparing:

- The proportion of cointegrated pairs in the linked pairs set versus the random pairs set.
- The survival rate of cointegrated pairs over time is used to assess their stability. This metric tracks the decay of the cointegration property. In relation to my evaluation matrices, a higher survival rate implies that the shared management signal may be a transient indicator, failing to sustain the equilibrium over longer windows.

4.4 Implementation

To ensure the validity of my experiments and create usable sets of stock pairs, every pair added must be validated, and any erroneous entries must be discarded according to a stringent set of criteria. The criteria used for this filtering was as follows:

- **Removal of Reflexive Pairs:** Any pairs where a company is paired with itself, e.g. (TSLA, TSLA), were discarded.

- **Uniqueness of Pairs:** If there is an occurrence of a pair and its reversed pair (e.g. (INTC, AMD) and (AMD, INTC)) in a set, the first pair takes precedence to avoid duplication.
- **Data Completeness:** Any company with missing stock price data for any trading day in the time period tested was removed.

To create the set of randomly selected companies, all companies in the ontology for the time period of Q2 2017 are selected using the following query:

```
PREFIX my: <http://york.ac.uk/>
SELECT ?t
WHERE { {
    ?company my:tradingsymbol ?t . } }
```

Given this list of companies, a set of all possible combinations of pairs is created. Pairs are randomly selected using the choice method from the random Python module [74], filtered and appended to the final set until it contains 50,000 pairs.

Creating sets of pairs sharing some number of links is a little more involved, pairs are first selected from the ontology using the following query:

```
PREFIX my: <http://york.ac.uk/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?person ?p ?t1 ?t2
WHERE { {
    ?person my:worksat ?company .
    ?person my:worksat ?othercompany .
    ?person foaf:name p .
    ?company my:tradingsymbol ?t1 .
    ?othercompany my:tradingsymbol ?t2 .
    FILTER(?t1 != ?t2)
} }
```

This query returns the relevant SEC report, the name of the person, and the stock market symbol (ticker) for both companies the person is affiliated with. This information gets fed into a dictionary where the key is the pair of companies and the value is a set containing names of insiders affiliated with the company. Once the dictionary is fully populated, the

values for each key are replaced with the number of unique names linking each respective pair (i.e. the size of that set). With this dictionary, separate subsets of pairs are created at intervals between one and five inclusive and each interval indicates a gradually increasing minimum number of links between the two companies in each pair (see Table 4.1).

For each company, matching tickers are found and their closing price for the aforementioned time period retrieved. The historical stock data is provided by the yfinance Python module (through Yahoo Finance) [73] and comprises information about the stock using the date as an index. For example, a query on NVIDIA Corporation (NVDA) provides the information displayed in Figure 4.2.

	Open	High	Low	Close	Adj Close	Volume
Date						
2017-03-31	109.010002	109.889999	108.400002	108.930000	107.806709	11020200
2017-04-03	108.949997	109.650002	107.419998	108.379997	107.262360	11130800
2017-04-04	103.400002	104.419998	100.339996	100.779999	99.740738	31782000
2017-04-05	100.019997	102.370003	99.500000	100.029999	98.998482	18676200
2017-04-06	100.239998	101.250000	98.410004	100.760002	99.720970	15878000
...	...	...	...	...	...	...
2017-06-23	158.679993	159.320007	153.220001	153.830002	152.404037	27214700
2017-06-26	155.160004	156.600006	148.330002	152.149994	150.739578	26599000
2017-06-27	151.440002	151.789993	146.350006	146.580002	145.221252	24987300
2017-06-28	149.320007	151.940002	145.750000	151.750000	150.343323	24873700
2017-06-29	150.600006	150.720001	144.080002	146.679993	145.320282	26610600

Figure 4.2: Information provided by yfinance on NVDA (NVIDIA Corp).

Table 4.1: Number of linked pairs for each quarter

	Number of links				
	1+	2+	3+	4+	5+
2017Q2	5143	395	247	216	104
2017Q3	4207	2484	2351	850	829
2017Q4	1885	226	90	71	63
2018Q1	9026	3152	169	94	83
2018Q2	4192	314	80	59	46

April–June 2018 Close-of-day Price for Surviving Cointegrated Pairs with Links ≥ 2

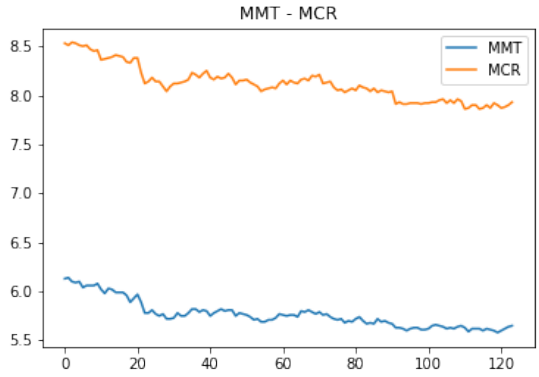

Figure 4.3: MMT and MCR close-of-day price

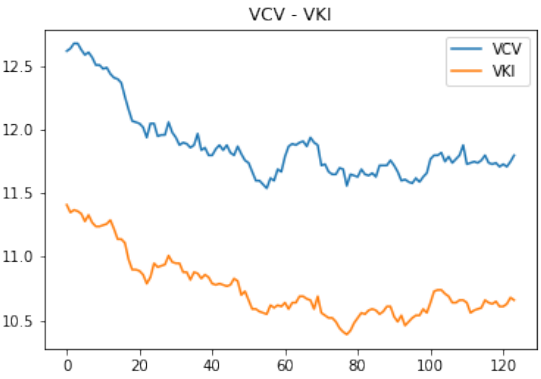

Figure 4.4: VCV and VKI close-of-day price

Visual representations of the close of day price trajectories for selected cointegrated pairs discovered via my ontological filter are provided in Figures 4.3, 4.4, 4.5, 4.6, 4.7 and 4.8

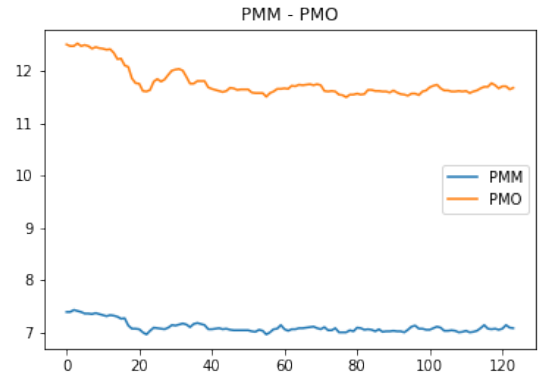

Figure 4.5: PMM and PMO close-of-day price

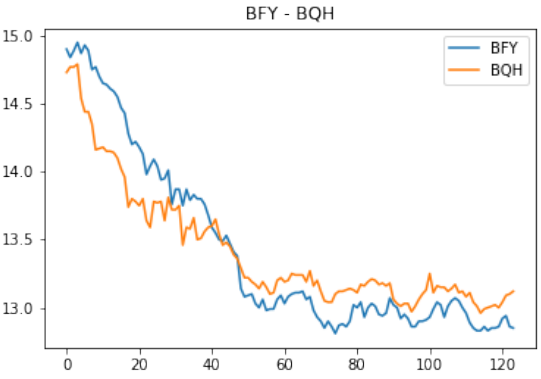

Figure 4.6: BFY and BQH close-of-day price

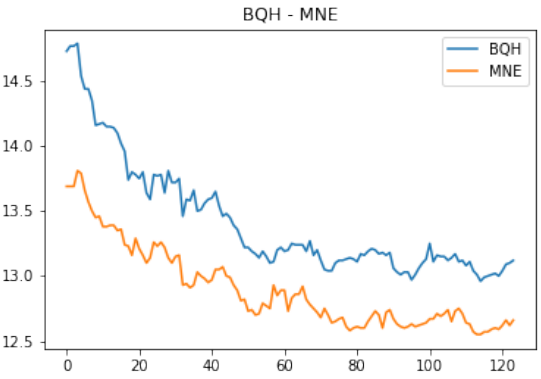

Figure 4.7: BQH and MNE close-of-day price

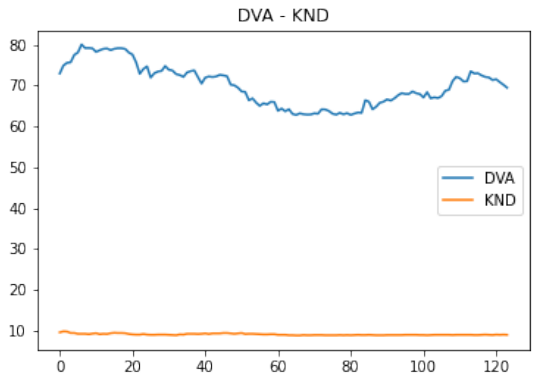

Figure 4.8: DVA and KND close-of-day price

Table 4.2: Experiment 1: Linked vs random pairs (2017 Q2)

	Coint.	Non-coint.	Total	Coint/Total
Random	3,406	46,594	50,000	6.81%
Links≥ 1	399	4,744	5,143	7.76%
	χ^2 test	p = .0108		

	Coint.	Non-coint.	Total	Coint/Total
Random	3,406	46,594	50,000	6.81%
Links≥ 2	41	354	395	10.38%
	χ^2 test	p = .0051		

	Coint.	Non-coint.	Total	Coint/Total
Random	3,406	46,594	50,000	6.81%
Links≥ 3	25	222	247	10.12%
	χ^2 test	p = .0397		

4.5 Results and Evaluation

4.5.1 Experiment 1: Cointegration Proportions

The results of Experiment 1 are presented in Tables 4.2 and 4.3. Table 4.2 shows the comparison of the Q2 2017 data between random pairs and linked pairs with varying minimum numbers of connections.

The proportion of cointegrated pairs is higher in the linked pairs set compared to the random pairs set for all link thresholds. Statistical significance is observed at the 95% confidence level, as indicated by the χ^2 test p-values.

To assess the stability of these findings over time, the experiment was repeated for three additional consecutive quarters. Table 4.3 summarises the cointegration ratios across these periods.

With few exceptions, linked pairs consistently show a significantly higher proportion of cointegration compared to random pairs. The best results are observed for pairs with at least two links, confirming my hypothesis. There were a few exceptions where linked pairs did not outperform random pairs occuring primarily during market transitions such as global news and macroeconomic shocks. In these periods, the cointegration from the

Table 4.3: Experiment 1: coint/total ratio ($p < 0.05$ winners in bold)

	2017Q2	2017Q3	2017Q4	2018Q1	2018Q2
Random	6.81%	5.20%	5.29%	6.59%	**5.87**%
Links≥ 1	**7.76**%	**8.72**%	5.15%	**9.92**%	4.53%
p-value	.0108	6E-22	.78	1E-29	.0004

	2017Q2	2017Q3	2017Q4	2018Q1	2018Q2
Random	6.81%	5.20%	5.29%	6.59%	5.87%
Links≥ 2	**10.38**%	**10.02**%	**10.62**%	**8.98**%	4.46%
p-value	.0051	4E-25	.0004	2E-7	.2895

	2017Q2	2017Q3	2017Q4	2018Q1	2018Q2
Random	6.81%	5.20%	5.29%	6.59%	5.87%
Links≥ 3	**10.12**%	**10.12**%	**15.16** %	7.10%	10.00%
p-value	.0397	8E-25	1E-5	.7897	.1162

corporate management signals is not maintained.

4.5.2 Experiment 2: Survival Rates

Experiment 2 examines the survival rate of cointegrated pairs over time by tracking the pairs identified in 2017 Q2 through 2018 Q2. Table 4.4 presents the findings. Specifically, I observe how many pairs identified as cointegrated in the initial quarter remain cointegrated in subsequent quarters. 'Survival rate' represents the percentage of pairs that keep the cointegration property.

As shown in Table 4.4, survival rate for random pairs over the 12-month period is 7.99% (272/3406) while for linked pairs, it is lower at 4.51% (18/399). Although this result appears to favour random pairs, the χ^2 test indicates that as the link strengthens, the difference loses statistical significance. This implies that while management links are indicative for cointegration, the longevity of the cointegration is affected by other factors over a longer period of time.

The survival rate is calculated as the percentage of pairs that remain cointegrated at the end of the period. In the 12-month quarterly survival test, random pairs have a higher survival rate than linked pairs for the Links ≥ 1 condition. However, the χ^2 test indicates that this difference is not statistically significant. Therefore, while the observed numbers

Table 4.4: Experiment 2: Survival among cointegrated pairs

	2017Q2	$\rightarrow$ 2018Q2	2018Q2 / 2017Q2
Random	3,406	272	**7.99**%
Links≥ 1	399	18	4.51%
	χ^2 test	p $= .0133$	

	2017Q2	$\rightarrow$ 2018Q2	2018Q2 / 2017Q2
Random	3,406	272	7.99 %
Links≥ 2	41	6	14.63%
	χ^2 test	p $= .1202$	

	2017Q2	$\rightarrow$ 2018Q2	2018Q2 / 2017Q2
Random	3,406	272	7.99 %
Links≥ 3	25	2	8.00%
	χ^2 test	p $= .9979$	

favour random pairs in this particular 12-month test, the result should not be interpreted that random pairs generally survive longer than linked pairs.

Another observation in Table 4.4 is that while linked pairs have a higher initial probability of cointegration, this 12-month result should be distinguished from the later day-by-day survival experiments, where linked pairs exhibit higher short-term survival rates in some windows.

This suggests that the relationship signal provided by shared management may be a high frequency or medium term indicator. Management transitions or strategy shifts involving shared directors may temporary align in stock price cointegration. This could be either an upward or downward trend depending on the company pair. However, as the strategies diverge or individuals shift focus to their other roles the cointegration weakens and disappears. On the contrary, the random pairs that were cointegrated without shared employees, may stay cointegrated, perhaps due to macroeconomic factors or sector specific dependencies that are less susceptible to personal influence.

4.5.3 Outcomes

These results support the hypothesis that using ontological connections between companies increases the likelihood of selecting cointegrated stock pairs. The statistical significance of the findings strengthens the validity of this approach. There is a trend, though not statistically significant, indicating that random pairs seem to be more stable. This observation requires further investigation to understand the factors contributing to this survival. This experiment has not been conducted in this thesis and is an area of future experimentation.

The data used to test the first of the two hypotheses formulated in the introduction was based on Form 3 SEC reports for 2017 Q2[1] and close-of-business stock prices for 01/04/2017 – 30/06/2017.

It should be noted that selecting pairs sharing links from the entire ontology is not particularly demanding in terms of time complexity. The result of this query leaves a small proportion of all possible pairs of stocks (of 10^4 order of magnitude or lower), and running the cointegration test on them is also a viable task without the need for special computational resources. On the other hand, testing approx. 10^7 possible pairs for cointegration becomes difficult, which is why only a random sample (of size 50,000) of all pairs was selected to estimate the proportion of cointegrated pairs when no restrictions on their pairing are imposed.

4.6 Surviving cointegrated pairs

This uses the same ontology generated as the other experiments and chapters. The period used here is 12 months, from 2017 Q2 to 2018 Q2. I start with the results from Experiment 1 on the 2017 Q2 data: all possible pairs are tested against the 'share-at-least-X-links' criterion (for $X = 1$, 2, and 3) to produce 3 data sets of cointegrated pairs at time 2017 Q2, one for each value of X. All cointegrated pairs from the sample of 50,000 random pairs constitute the fourth dataset of cointegrated pairs. I then use the time series for the period 1 Apr 2018 — 30 June 2018 to find the number of pairs in each of the 4 data sets that are still cointegrated during that period. Again, the results for each of the first three sets are compared with the fourth data set or random pairs, and the statistical significance of any difference is tested using the χ^2 test. Note that if a company has folded

[1] `https://www.sec.gov/Archives/edgar/full-index/2017/QTR2/form.idx`

in the 12-month period, its (no longer existing) stock is assumed not to be cointegrated with any other stock. This experiment is then repeated using a sliding window of three months to test the retention of the cointegration relationship between quarters and to study how linked pairs and random pairs compare.

I found that restricting the search for trading pairs to companies that share a certain number of links does not affect the percentage of pairs that remain cointegrated over time. The above restriction will significantly affect the proportion of pairs that remain suited to pair trading over time.

The following analysis of survival time constitutes a specific contribution to the evaluation of ontological relevance. I assess whether the cointegration is more robust than random cointegration in high frequency domain.

The focus of this chapter is on cointegrated pair survival over short periods of up to a few days. In order to be considered cointegrated, the pairs had to pass both the Engle-Granger [21] and Johansen's [22] cointegration tests. This strict criterion ensures robustness in identifying truly cointegrated relationships, minimising the risk of false positive results. Two sets of results are reported in this chapter.

Set 1 Here, the cointegration property is established on the basis of a single day's worth of intra-day data ('Day 1'); then, its survival is studied in the following four days of trading, on a day-by-day basis.

The initial ratio of cointegrated pairs to total pairs on Day 1 is similar for linked and non-linked pairs, at around 16% (Table 4.5).

Table 4.5: Linked vs Non-linked Pairs

	Coint.	Non-coint.	Total	Coint/Total
Non-linked	409	2100	2509	16.30%
Links ≥ 1	107	560	667	16.04%
	χ^2 test	p = .9185		

Table 4.6 and Figures 4.9 and 4.10 show the cumulative surviving cointegration (pairs that remained cointegrated every day from 1 to 5) pairs on each of the following days. Here the cointegration property is calculated for each individual day, but the results only show pairs which also remained cointegrated on each of the days up to and including the day in question. In other words, a pair is listed as cointegrated on Day 3 if it was also cointegrated on Day 2. (Day 1 is the reference day on which all cointegrated pairs were selected, so the

result for that day is 100% for both classes.)

Table 4.6: Day-by-day survival rate, no return from the dead

Day	1	2	3	4	5
Links ≥ 1	107	18	4	0	0
		16.82%	3.74%	0.00%	0.00%
No links	409	33	4	0	0
		8.07%	0.98%	0.00%	0.00%
χ^2 test p-value		.0118	.1056	.8882	.8882

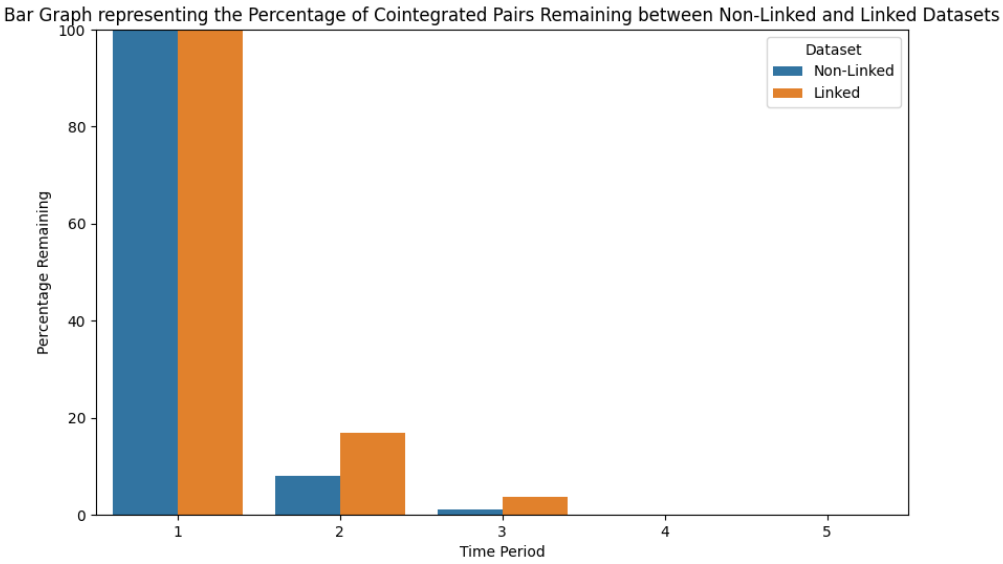

Figure 4.9: Day-by-day survival rate, no return from the dead

To rephrase it in a more idiomatic way, once the cointegration property 'dies', there is no return from the dead. Here linked pairs have double the survival rate (16.82%) compared to non-linked pairs (8.07%) on Day 2, but the difference is not significant in later periods. The survival plot underwent log-rank testing [75] and gave significant p-value (0.0044), thus there is a statistically significant difference in survival between the two groups for the described set up.

Table 4.7 represents a point in time test at day 5 regardless of whether they are out of cointegration or not on days 2-4, there is no significant difference between linked (14.02%) and non-linked pairs (11.00%) (Table 4.7) but the proportion is higher for the linked class.

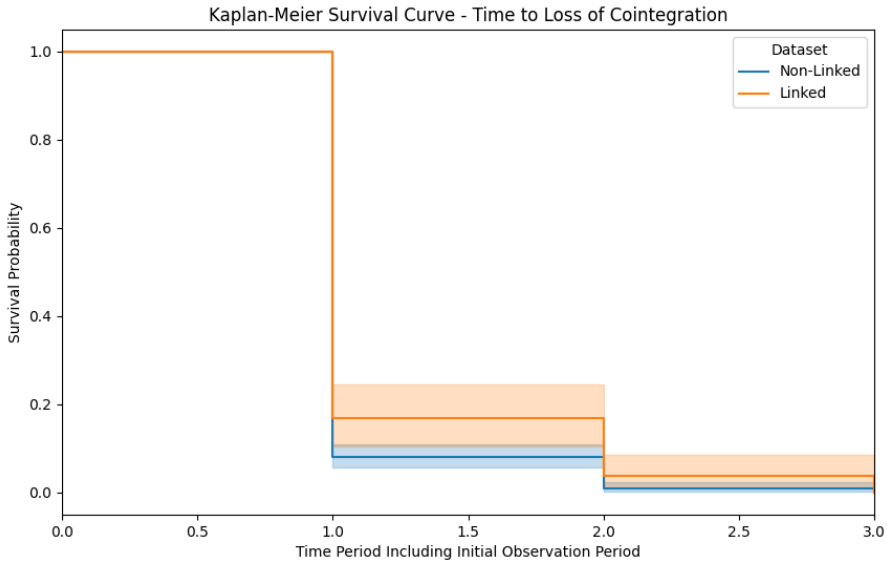

Figure 4.10: Set 1: Day-by-day survival curve, no return from the dead

The non-zero values indicate that some pairs returned to a stationary state by the final day of the window.

Table 4.7: Cointegration survival: Day 5 vs Day 1

Day	1	5	Day 5 / Day 1
Non-linked	409	45	11.00%
Links ≥ 1	107	15	14.02%
χ^2 test	p = .4857		

Table 4.8: Expanding test window survival rate

Day	1	2	3	4	5
Non-linked	16.30%	7.57%	3.27%	5.38%	5.82%
Links ≥ 1	16.04%	**12.74%**	**6.90%**	**9.45%**	**9.15%**
χ^2 test p-value	0.9185	3E-5	4E-5	0.0002	0.0027

Set 2: This uses an expanding window rather than individual daily slices. The cointegration must hold over the entire cumulative duration of 1 to D. For this set of results, the cointegration property is again established on Day 1 but is then tested on a gradually expanding window of data [Day 2–Day D], where D ranges from 2 to 5 (see Table 4.8 and Figures 4.11 & 4.12). The lower survival rates compared to Set 1 reflects the increased statistical significance. The linked pairs have a significantly higher cointegrated/total ratio compared to non-linked pairs in all survival periods from Day 2 to Day 5. The survival

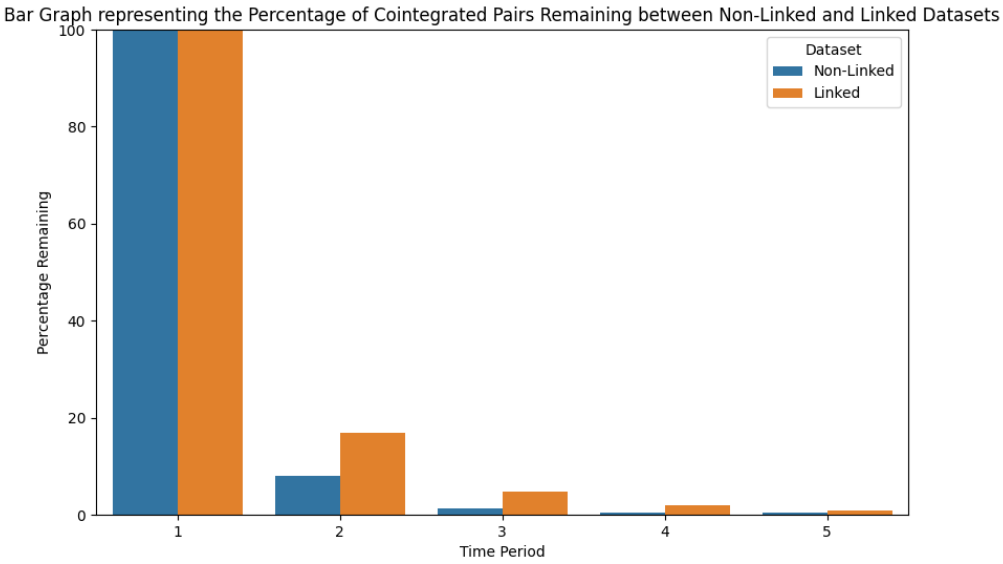

Figure 4.11: Expanding test window survival rate, no return from the dead

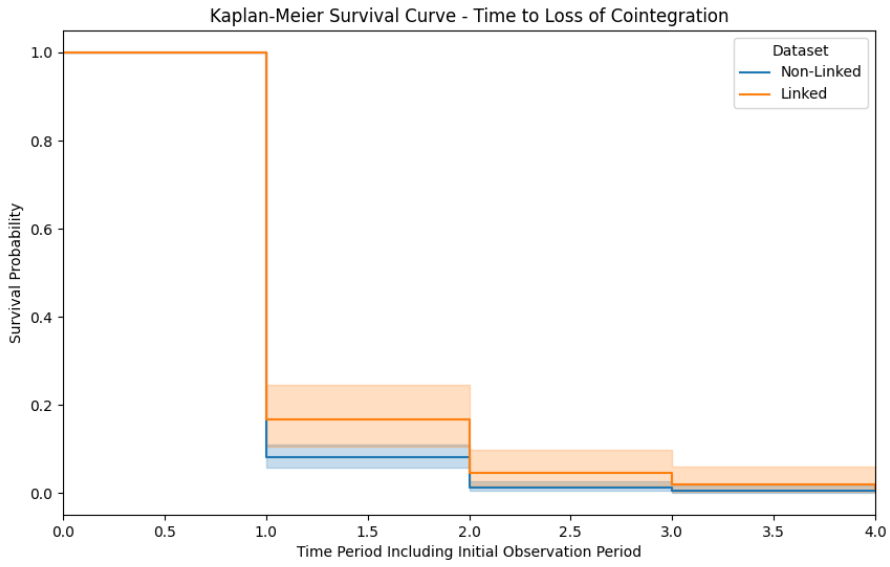

Figure 4.12: Survival curve: expanding test window, no return from the dead

plot underwent log-rank testing and yielded a significant p-value (0.0091); thus, there is a statistically significant difference in survival between the two groups when considering cointegrated pairs from the observation period each day in the survival period individually.

Table 4.9 shows the numbers of surviving pairs tested during the Day 2 to Day 5 period against the Day 1 baseline.

Table 4.9: Cointegration survival: Days 2–5 vs Day 1

	1	5	Day 5 / Day 1
Non-linked	409	25	6.11%
Links ≥ 1	107	5	4.67%
χ^2 test	p = 0.7380		

4.7 Discussion and Overall Evaluation

To summarise the empirical findings : the data across consecutive quarters demonstrates that restricting the search space to linked pairs (where $\theta \geq 2$) consistently yields at least 50% higher ratio of cointegrated pairs compared to a random selection. Specifically, in Q2 2017, the linked set reached a cointegration ratio of 10.38% against 6.81% for the control; a delta that is statistically significant ($p = .0051$). This confirms that the qualitative relational data extracted in Chapter 3 provides a filter for reducing the noise of potentially quadratic complexity of pair selection.

The results of the two experiments are listed in Tables 4.2–4.4. First, Table 4.2 lists the results of Experiment on the 2017 Q2 data, comparing the number of cointegrated pairs in the random pair set with one of the sets of linked pairs (with a minimum number of links equal to 1, 2, and 3). The table shows that in all three cases, the proportion of cointegrated pairs in the set of linked pairs is higher, and the results are statistically significant at the 95% level. The number of linked pairs rapidly decreases as the required minimum number of links is increased. This negatively affects the significance levels (p-value) calculated by the χ^2 test, which is why this number was capped at three and results for higher values are not reported here.

The experiment was repeated with data from four further consecutive quarters in order to study how stable my initial findings are over time. Table 4.3 lists only the number of cointegrated pairs as a fraction of the set of pairs in question. With one exception, all

results show that selecting only linked pairs is significantly better than using random pairs with respect to this criterion or there is no clear winner. The best results are achieved for links ≥ 2 with 4 wins and one draw in the studied period.

While the first experiment tested – and confirmed – the hypothesis that information about links between companies can be used to increase the likelihood of selecting pairs that are cointegrated, the second experiment focuses on the survival levels among cointegrated pairs over time; that is, measuring the percentage of pairs that remain cointegrated at the end of the studied period. Again, the survival in a set with a given minimum number of links between each pair is compared to that of a set of random pairs. The only statistically significant difference is observed between the set with Links ≥ 1, and the sample of random pairs, with the latter showing a lower survival rate.

4.7.1 Addressing Research Question 2

The experiments conducted in this chapter support the hypothesis that incorporating ontological information from SEC reports enhances the identification of cointegrated stock pairs. By leveraging connections between companies through shared personnel and major shareholders, I demonstrated a statistically significant increase in the proportion of cointegrated pairs compared to random selection.

These findings suggest that qualitative information embedded in financial reports can provide valuable insights beyond traditional quantitative methods. Specifically, while quantitative models treat companies as blackboxes and look for correlations in historical data, qualitative structural data uncovers hidden topology in corporate networks that influence price movements. The use of ontologies in financial analysis presents a promising avenue for developing more effective trading strategies.

However, the observed higher survival rates among linked pairs over time indicate a need for further research to understand the longevity of these cointegration relationships. Future work could explore additional factors influencing cointegration stability and extend the ontology to incorporate other types of corporate relationships.

The statistical analysis provides strong evidence in support of the hypothesis, with results showing significant statistical confidence. The findings demonstrate that utilising the ontology dataset derived from SEC reports can enhance a trader's ability to identify potentially profitable pairs. These results directly address Research Question 2, confirming

the practical utility of ontological information in trading strategies.

This chapter shows that converting information from the SEC reports into an ontology provides a way to answer queries of quadratic complexity, which would be unfeasible without such automation. The (rather restricted) part of the reports extracted here is already showing its potential relevance to traders. I proposed two hypotheses based on a combination of common expectations about the relevance of additional information (which is commonly used in fundamental trading) and generalisations from previous insights about trading pairs of stocks of the same company.

The results for the first hypothesis have statistical significance for certain common values of its only parameter. The findings here have the potential to inform a more efficient search for companies suited for pair trading. At the same time, it is also evident that while my initial intuition was good, the choice of parameter, namely 'the minimum number of links between a pair of companies', was important.

The operational relevance of the second experiment is that the percentage of cointegrated pairs that remain cointegrated over three months is not negatively affected by the use of linked pairs only, which means that we can benefit from the advantages of this approach. This approach increases the probability of finding a cointegrated pair without any downside on the expected longevity of such pairs.

A few final considerations: it is also interesting to look at each of the six pairs of stocks linked through $2+$ links that remain cointegrated after a year (Table 4.10). Here, one finds that, with one exception, the two members of a pair have very similar names. Pairs of this type could potentially be selected on the basis of simple string similarity, at the risk of generating many false positives. Note that in my approach, no information about name similarity was used. I had also thought of a simpler criterion for pre-selecting pairs, where both companies would have to be in the same sector. However, it is not clear what the relevance of such criteria would be in the case of trusts and funds with their diversified portfolios.

In summary, linked pairs seem to exhibit higher survival rates compared to non-linked pairs, especially in the short term (1-2 trading days). The difference is always statistically significant on Day 2 and persists until Day 5 when cointegration is tested over the entire period following Day 1. It is possible that this masks, to some extent, the random noise in the data, but also the trend that the advantage of linked pairs diminishes over time. The results still indicate that only 1 out of 6 linked cointegrated pairs survives to the end of Day

2. This probably means that using the link property alone will provide enough support to use pair trading beyond the span of a single day, but it may provide additional assurances about the likelihood that the pair would stay cointegrated over this period – a conjecture that deserves further attention.

Table 4.10: Surviving cointegrated pairs: Links≥ 2, $2017\,\mathrm{Q2}\rightarrow 2018\,\mathrm{Q2}$

Pair 1:
MMT: MFS Multimarket Income Trust
MCR: MFS Charter Income Trust
Pair 2:
VCV: Invesco California Value Municipal Income Trust
VKI: Invesco Advantage Municipal Income Trust II
Pair 3:
PMM: Putnam Managed Municipal Income Trust
PMO: Putnam Municipal Opportunities Trust
Pair 4:
BFY: BlackRock New York Municipal Income Trust II
BQH: BlackRock New York Municipal Bond Trust
Pair 5:
BQH: BlackRock New York Municipal Bond Trust
MNE: BlackRock Muni New York Intermediate Duration Fund, Inc.
Pair 6:
DVA: DaVita Inc.
KND: Kindred Healthcare, Inc.

4.8 Summary

I demonstrated that ontological links provide a statistically significant heuristic for pair trading. By leveraging connections network topology, I successfully identified pairs with 10.38% cointegration probability against a 6.81% random baseline. This addresses Research Question 2 by confirming that the qualitative data extracted in Chapter 3 is directly translatable to quantitative gains.

In summary, the integration of ontological metadata into the pair trading pipeline provides a significant heuristic for identifying cointegrated assets. By leveraging the insider network reports with structured data construction, I have provided empirical evidence to address Research Question 2, showing that the qualitative relational links are strong predictors of quantitative price. Future refinements should consider the decay of these links to mitigate the survival observed in Experiment 2.

Finally, it must be acknowledged that the success of this hybrid strategy is dependent on the quality and completeness of the underlying ontology. The semantic accuracy of the links, such as a director that is mapped between two companies is the same person is crucial. If the parsing methodology introduced in Chapter 3 were to introduce wrong edges to name collisions or misconnections due to wrong filing at the time, the proportion of linked companies would degrade. Consequently, the statistical advantage over random pair selection observed in these experiments would likely be affected. Maintaining structural integrity and semantic accuracy of the ontology is a prerequisite for its use in other experiments or real life applications.

Chapter 5

Learning from Reports with Inductive Logic Programming

This chapter directly addresses Research Question 3 (RQ3) defined in Chapter 1 : Is it possible to use a machine learning algorithm to learn from data represented as an ontology in order to search through a range of automatically formulated hypotheses and test their relevance to a financial forecasting-related application?

I investigate how a hybrid machine learning approach can automatically discover and evaluate pair trading strategies. This chapter builds directly on the foundational work established in earlier chapters of this thesis. Chapter 3 showed the data extraction methodology and ontology construction from financial reports data; Chapter 4 demonstrated the validation of the ontology's relevance to cointegration. While I demonstrated the possibility with handcrafted queries such as shared directors, and filters that are suitable for trading pairs, this chapter will scale that concept by using ILP (using Progol) to automatically explore a much wider range of hypotheses. By enriching the ontology with node2vec embeddings and applying more background knowledge by clustering these embeddings, I aim to present more accurate, human-readable rules that can reduce the computationally expensive task of testing millions of stock combinations.

This chapter is structured as follows: Section 5.1 outlines the task and initial analysis. Section 5.2 details the experiment design with clustering parameters and background knowledge encoding. Section 5.3 presents the evaluation of two experiments and Section 5.4 summarises the findings, addressing RQ3 and outlining the limitations that have been experienced, leading to the research findings in Chapter 6.

In this chapter, I demonstrate how a hybrid learner can be utilised to find strategies and rules using the ontology that I built from financial reports. Ontologies possess the expressive power of subsets of first order logic. I also demonstrate the capability of Inductive Logic Programming (ILP),specifically Progol, to systematically explore a diverse range of automatically generated hypotheses derived from SEC reports. This process facilitates the identification of the most pertinent hypotheses for the financial application domain at hand.

There is a history of hybrid machine learning approaches in which the results of an unsupervised learning algorithm are used to provide data annotation from which ILP can learn in the usual supervised manner [76]. Here, I consider the task of predicting the property of cointegration between the time series of stock prices of two companies, which can be used to implement a robust pair-trading strategy that can remain profitable regardless of the overall direction in which the market evolves. I start with an original FinTech ontology of relations between companies and their managers, which I have previously extracted from SEC reports and quarterly filings that are mandatory for all US companies in Chapter 3. When combined with stock price time series in Chapter 4, these relations have been shown to help find pairs of companies suitable for pair trading [77]. I use node2vec embeddings to produce clusters of companies and managers, which are then utilised as background predicates in addition to the relations linking companies and staff present in the ontology, as well as the values of the target predicate for a given time period. Progol is used to learn from this mixture of predicates, combining the numerical and structural relations of the entities represented in the data set to reveal rules with explanatory and predictive power.

5.1 Tasks and Objectives

To answer RQ3, I propose a hybrid symbolic and sub-symbolic architecture that leverages the strengths of ILP and the feature extraction capabilities of graph embeddings. The objective is to automate the discovery of cointegration rules, moving from the handcrafted links in Chapter 4.

The idea of using non-numerical information for trading is not novel and, in fact, constitutes the basis of the so called fundamental trading strategies, which examine the details of the company finances, assets, plans, and the membership of their senior management. However, there is no standard, well established way to incorporate such knowledge into

day-to-day trades, and, in fact, it can be argued that the short-term movements of the market have little to do with the long term viability of an investment. Nevertheless, I have considered a possible use of my ontology for this purpose; namely, I hypothesised that if two companies share directors or other senior staff, it could lead to a flow of information that would make it more likely for the two companies to meet the cointegration test.

I tested this hypothesis in Chapter 4, in which it transpired that applying this criterion[1] does increase the proportion of cointegrated companies in a statistically significant way. It would be quite common to find that around 5% of randomly selected company pairs are cointegrated, while after applying this criterion, their share may increase by a couple of percent or even double in some cases. Overall, such handcrafted rules bring palpable benefits, but their potential is still limited.

Since that study, I have taken this work in two research directions to automate the process. The first direction explores a hybrid machine learning approach that combines sub-symbolic graph embeddings with an existing symbolic ILP system, Progol. Second direction involves the development of concurrent machine learning system in Description Logic which I have been working on the use of an in-house learner from ontologies in description logic [1] to automate the search for suitable ontology-based hypotheses. I will discuss the second approach more in Chapter 6. This chapter focuses on first direction, namely, the use of node embeddings produced by the `node2vec` algorithm [78] to extract important properties of the graph nodes, reflecting the content and topology of each local neighbourhood in an unsupervised way because the immediate goal is to validate the pheasibility of a hybrid symbolic and sub-symbolic architecture. It is common to cluster the resulting node embeddings and visualise the results in order to discover similarity between nodes. I have adopted the same approach here, using k-means clustering with a hand-selected number of clusters ($k = 20$). This number has been selected after conducting several experiments with the ontology dataset that I have for different quarters. 20 is chosen based on my observation because it gave the right level of distribution for each cluster. The cluster membership was then encoded as background knowledge of a Progol learning task. This is done programmatically from my custom cluster encoder generator to Progol files, where the target predicate `coint(Comp1,Comp2)` represented whether the prices of two companies are cointegrated or not for the period in question. Since the cointegration property is symmetric, this is also reflected in the training data. In addition, I make use of background predicates showing the affiliation of managers with companies, `keyperson(Company,Person)`, the overall number of people connected to a company, `ccon(Company,PersonCount)`, and

[1] modified by a parameter setting a minimum threshold on the number of shared staff.

the companies connected to a person, `pcon(Person,CompanyCount)`.

5.1.1 Data Analysis

After the data is parsed, it is stored in triplet form. This is then serialised as n-triples to be imported into an RDF graph database and our own application for analysis using the SPARQL query language [79]. A sample of the data is shown in Figure 5.1. This shows the triples generated by my ontology generator, see Chapter 3 for more details.

This can be used to construct powerful SPARQL queries to extract information from this representation and use rich visualisation tools to see and understand the data. A sample of such data visualisations is shown in Figure 5.2 and Figure 5.3.

5.1.2 Learning and Experiment Preparation

Each experiment is prepared by our metaprogramming application generating Prolog/Progol source files. The steps that are followed each time are listed below:

- Load the serialised ontology data : The ontology that I prepared was in turtle format see Chapter 2. The ontology is loaded into the graph database of my choice OntoText, because it supports SPARQL and W3C standards, however it could be any graph database and adapted.
- Run a query to select companies (background information) : Then SPARQL has been executed to extract interesting companies.
- Run a query to select linked people with the company information (background information) : Then SPARQL query has been executed to extract the linked people to those companies.
- Split the data into a training set and a test set : This is done so I can evaluate my machine learning approach, which is a standard approach to score the performance.
- Run the cointegration test for all pairs of companies in the report for the period in question to generate positive and negative examples : The cointegration tests become additional background information for the pairs of companies and in return it helps the ILP system to works to that goal.

Source: http://sec.com/0001438253

subject | predicate | object | context | all | Explicit only | Show Blank Node

	subject	predicate	object
1	http://sec.-com/0001438253	http://schema.org/jobTitle	Chief Executive Officer
2	http://sec.-com/0001438253	http://schema.org/jobTitle	President & CEO
3	http://sec.-com/0001438253	http://schema.org/jobTitle	President and CEO
4	http://sec.-com/0001438253	rdf:type	http://xmlns.com/foaf/0.1/Per-son
5	http://sec.-com/0001438253	http://xmlns.com/foaf/0.1/name	Forman Michael C.
6	http://sec.-com/0001438253	http://york.ac.uk/cik	0001438253
7	http://sec.-com/0001438253	http://york.ac.uk/is10percent-owner	"false"^^xsd:boolean
8	http://sec.-com/0001438253	http://york.ac.uk/is10percent-owner	"true"^^xsd:boolean
9	http://sec.-com/0001438253	http://york.ac.uk/isdirector	"true"^^xsd:boolean
10	http://sec.-com/0001438253	http://york.ac.uk/isofficer	"true"^^xsd:boolean
11	http://sec.-com/0001438253	http://york.ac.uk/isother	"false"^^xsd:boolean

Figure 5.1: Knowledge graph sample

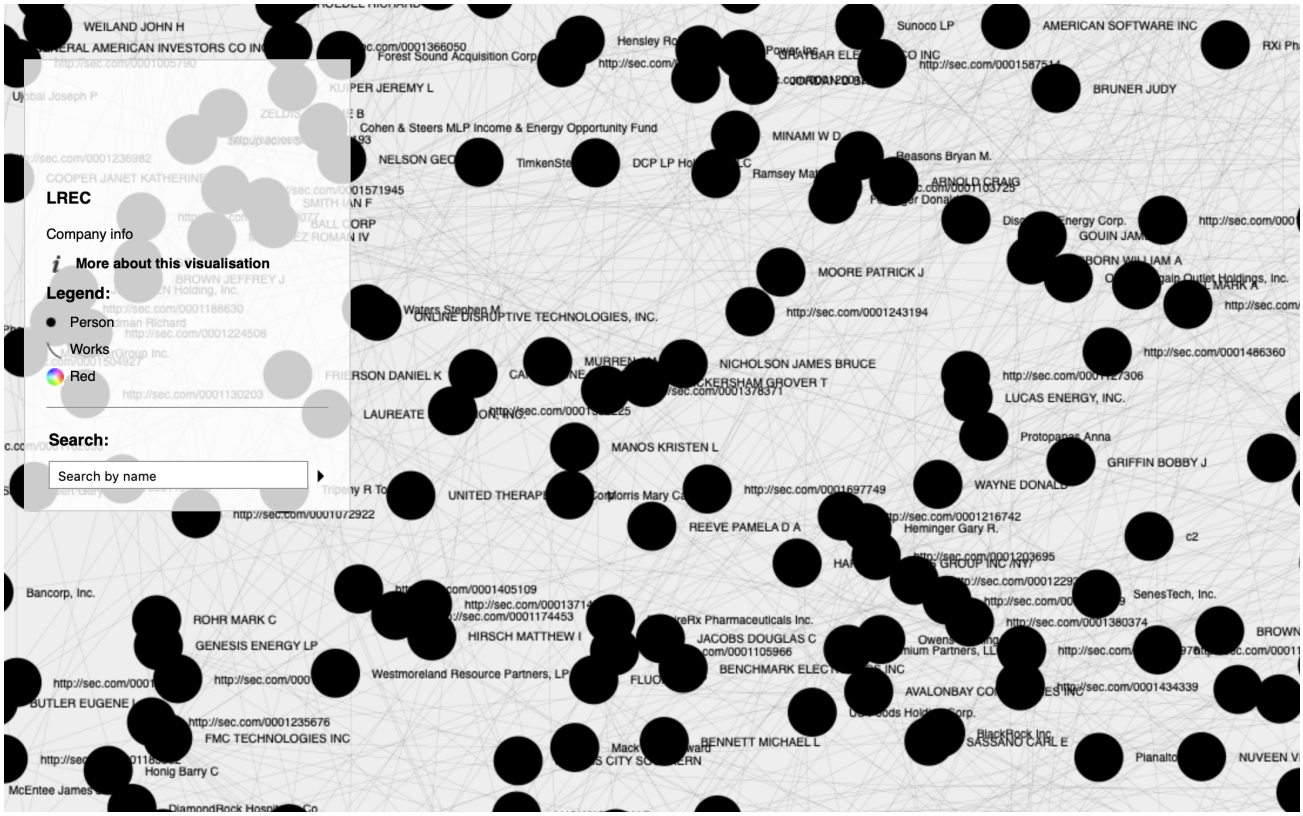

Figure 5.2: Ontology visualisation (part)

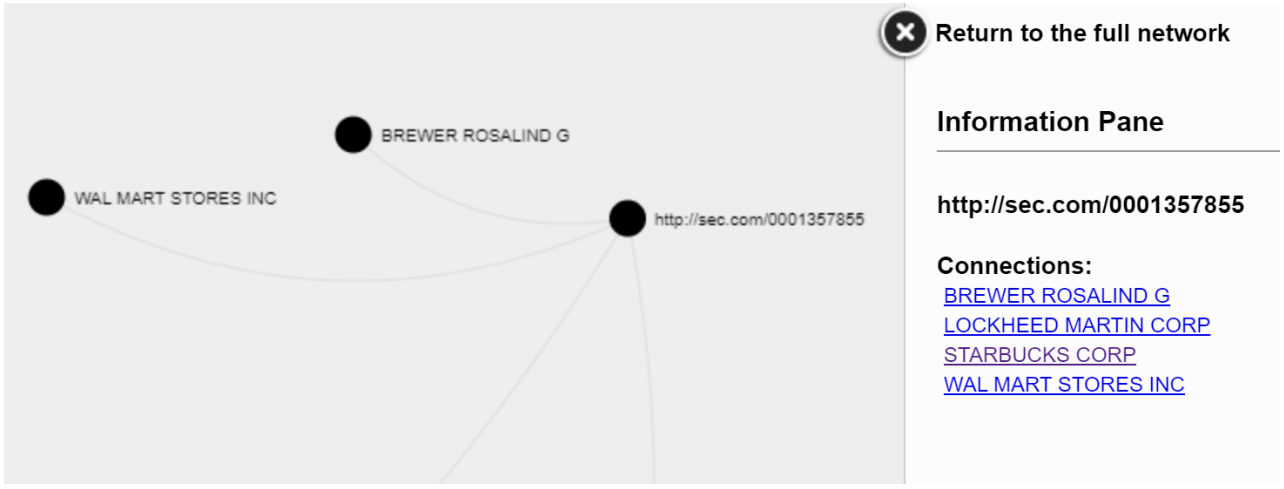

Figure 5.3: Ontology visualisation of a person represented by an URI

- Run the node2vec algorithm on the data to generate node embeddings : In order to run the clustering algorithm, the raw ontology needs to be converted to vector space. The graph embedding algorithm does that.
- Run k-means on the embeddings to obtain clusters: This is additional background information that I will feed in to the ILP system. The methodology has been explained earlier and numerical values of embedding buckets will be stored in the ILP for rule discovery.
- Generate a Progol source file from the background knowledge and target predicate examples : This is an automated process code generator that takes all the files generated previously and creates Progol source files. Progol as a system does not understand ontologies, hence the source files are generated to include all the information. For Progol, background information becomes the source code for the rules. In addition, the actual mode directives becomes the goals to drive the ILP discovery.
- Run Progol on the data : At this point progol can be instructed to run against the source files generated automatically.
- Repeat : Then this process can be executed again for each quarter and data points as necessary.

This multi-stage pipeline constitutes a hybrid learning approach. Sub-symbolic node2vec embeddings condense the local topology of the financial graph into a vector space, which is then clustered by k-means algorithm. These clusters serve as semantic labels (e.g., companycluster(A,10)) and augment the background knowledge of the ILP system Progol.

5.2 Experiment Design

Two types of experiments have been carried out. In the first experiment, I used Progol for descriptive learning, testing the relevance of the ontology node embeddings for discovering cointegration rules. To reduce the number of instances here, I only considered pairs that are already connected in the ontology through a manager who works for both companies. The result is a set of rules that utilises constraints on the number of managers linked to a company and on its membership in a particular cluster. These rules have high accuracy, and some of them identify groups of companies with 10 or more members, where each pair of companies in the group is cointegrated! This is a significant result that deserves further analysis but can already be used by a trader.

In the second set of experiments, I selected a list of companies at random, split it into two, and used the first list to generate training pairs of companies (cointegrated vs non-cointegrated), while the second list was used to generate test pairs in which neither company appeared in the training data. I then use Progol to learn rules that are evaluated on unseen data. The results show that the rules I find also have high predictive power.

One of the steps in the experiment design was the generation of node embeddings and clustering, which served as a background knowledge for the ILP system. The node2vec algorithm was applied to the ontology graph to generate vector representations of each company. The parameters for the random walks were chosen based on empirical testing to capture sufficient topological structure without additional computational overhead, which involved several trials to capture. The walk_length was set to 16 to ensure deep exploration of a node's neighbourhood, num_walks was set to 100 to provide sufficient statistical significance and workers 2 was selected to parallelise the task.

Once the embeddings were generated, k-means clustering was applied directly to these vector representations. The number of clusters, 20, was selected through several experiments' observations of the datasize size and visualisation of clusters, ensuring that clusters were neither too broad to be generic nor too close to cause overfitting. The resulting cluster membership for each company was then encoded into prolog syntax with my custom Python code generator scripts (companycluster(compA, 12)). The script parsed the k-means output and generated corresponding text strings creating sub-symbolic clustering output with the symbolic ILP environment.

5.3 Results and Evaluation

In each of the following experiments, a new ontology is generated from scratch from three months' worth of SEC reports. Only tabular data is used, and the free form text is ignored. I should clarify that, at the moment, the only parts of the SEC reports that I use are related to insider trading: '*The federal securities laws require certain individuals (such as officers, directors, and those who hold more than 10% of any class of a company's securities, together I will call "insiders") to report purchases, sales, and holdings of their company's securities by filing Forms 3, 4, and 5.*'[2]

Each ontology is a graph with nodes representing either a company or a person, and edges showing when a person is linked to a company, see Section 2.4.4 for more details. The algorithm `node2vec` is then used to generate an embedding for each of the graph nodes. The embedding is a real-valued vector of size 20. For the remaining parameters, the choice was: `walk_length=16`, `num_walks=100`, `workers=2`. These numbers have been chosen by various experiments, observation and intuition. The performance of the algorithm was another concern in order to generate the data so one could conduct the experiments quickly. All node embeddings were then clustered using k-means with $k = 20$. The resulting clusters reflect the similarity between nodes as a combination of two factors, *homophily*, i.e. sharing the same neighbourhood, and *structural equivalence*, reflecting the role a node plays in the graph (e.g. a 'hub' node). The relative importance of either factor is modified through the choice of the parameter `walk_length`. While a cluster could bring together nodes representing companies with those representing people, I will only take into account the former.

With these results in hand, I set up several experiments with the purpose of learning rules that would predicate the cointegration of two companies based on their membership in a given cluster, the number of connections to a person or company, and the existence of people connected at the same time to a pair of companies.

5.3.1 Experiments

The first experiment uses ontologies generated from SEC reports for Q3 2017 and Q4 2017. There is data on 1948 companies for Q3 and 1910 companies for Q4.

[2] `https://www.investor.gov/introduction-investing/investing-basics/glossary/forms-3-4-and-5`

Mode declarations in ILP systems [80] define templates, types and direction of information for the predicates the system allows when hypothesising new logical rules.

modeh/2 : Mode Head constrains the format of the predicate allowed in the head of the hypothesised clause.

modeb/2 : Mode Body constrains the format of the predicates allowed in the body of clause.

/2 or other number is prolog concept where it define number of arguments for the predicate.

The mode declarations used are as follows:

```
:- modeh(1,coint(+company, +company))?
:- modeb(1,companycluster(+company,#int))?
:- modeb(1,ccon2(+company,#int))?
:- modeb(1,pcon2(+person,#int))?
```

coint/2: Defines whether a pair of companies is cointegrated.

companycluster/2: Shows the cluster to which a company belongs.

ccon2/2: The predicate succeeds if its first argument is a company with a number of connections that is equal to or greater than the number in the predicate's second argument. In other words, the predicate checks that a company is connected to a certain minimum number of people. Expressed as Prolog code:

```
pcon2(Company,Threshold):-
  pcon(Company,NumberOfConnectedPeople),
  NumberOfConnectedPeople >= Threshold.
```

pcon2/2: The predicate succeeds if its first argument is a person with a number of connections that is equal or greater than the number in the predicate's second argument. In other words, the predicate checks that a person is affiliated with a certain minimum number of companies. Expressed as Prolog code:

```
pcon2(Person,Threshold):-
  pcon(Person,NumberOfConnectionsToCompanies),
  NumberOfConnectionsToCompanies >= Threshold.
```

The first experiment follows up on the findings from the previous Chapter 4 by moving to automated rule discovery. Instead of analysing survival rates, the first experiment utilises Progol ILP learning to automatically generate and evaluate cointegration rules based on the background knowledge. To reduce the search space, I limited the experiment to related company pairs across two quarters (see Table 5.3).

The second experiment tries to predict unseen data for the entire population of reporting companies for these quarters (Q3 2017 and Q4 2017). Generation of all possible pair combinations of the entire pool is computationally very expensive, hence the case of randomly sampled subsets from the main pool. It contains two disjoint sets; first containing 300 companies to generate the training pairs and second, containing 400 companies to generate testing pairs. Both the first and second experiments utilised the same ontology dataset that was generated earlier, but split with training and test data to evaluate predictive rules on unseen data.

5.3.2 Outcomes

In these experiments the ground truth was the cointegration status of a pair of stocks, strictly calculated using historical time-series data via the Engle-Granger and Johansen tests (as defined in Chapter 2)

It is important to note that the structural differences between the rules discovered in Q3 2017 (Table 5.1) and Q4 2017 (Table 5.2), where the difference reflects the evolution of the underlying knowledge graph. As the new reports are filed for each insider trading or corporate management change, the graph and the derivative node2vec embeddings evolve. As a result ILP learner also adapts to these topological changes and resolves different logical rules.

The overall accuracy of the theory learned by Progol on the test set is 50.42% as shown in Table 5.6 where it functions as a confusion matrix evaluating the predictive perfomance of the ILP model on unseen data used in Experiment 2. It represents the actual ground truths of the data (cointegrated vs not cointegrated) as concluded from the time series analysis. Highly volatile data and noisy financial market data, where predicting price movements is inherently difficult due to unpredictable macroeconomic noise. This represents a strong result.

The practical value of these learned rules becomes obvious when analysing the filtering

power of the ILP learner. For instance, the unfiltered test set which is the baseline proportion of naturally cointegrated pairs was 3.13%. However, when looking at the resulting data sample, the subset of pairs specifically accepted by the ILP learned rules, the proportion of cointegrated pairs rises to 5.78%. This represents an 85% relative increase in the density of successful pairs over the random pairs. This difference is statistically significant in that relational and cluster-based features can effectively reduce $O(n^2)$ search space. This means the ILP rules effectively cut the list of candidate pairs in half, saving computational time while discarding almost none of the good cointegrated pairs.

In the Table 5.4 and Table 5.5, the quality and coverage of the individual rules are presented using positive (pos) and negative (neg) metrics where they represent the hypothesis cover set computed on the test data. The positive metric indicates the exact number of test examples covered by a specific rule that are geniunely cointegrated. Similarly negative metric indicate the number of pairs covered by the rule that are not cointegrated.

We can look into the quality of the individual rules listed in the two tables, which also show the number of positive and negative test examples each of them covers, their precision and whether that figure is statistically different from the distribution of the entire test set. A two-tailed χ^2 test was used for that purpose.

I have grouped the rules learned by Progol by their type for ease of reading. First, I have nine rules only specifying the cluster membership of either company in the pair (see top of Table 5.4). Intuitively, the rules appear under-constrained, as it is unlikely that any company from a given cluster will be cointegrated with all other companies. This intuition is supported by evidence as when the rules are tested, they have large coverage but low precision (defined as the proportion of true positives in all pairs selected by the rule). Even so, some of the results show statistical significance at the $\alpha = .05$ significance level when compared with the 3.13% ratio of cointegrated pairs in the entire test set.

The second and most numerous type of rule is one that specifies the clusters each of the two companies belong to. In one case, the rule asks for the two companies to belong to the same cluster, namely 'cluster 10'. Most rules however specify two different clusters. Some of these rules substantially outperform the default strategy of selecting pairs at random, e.g. choosing pairs combining clusters 13 and 15 to select pairs from the unseen test data increases the proportion of cointegrated pairs 5-fold. It is tempting to find out a common denominator for all companies in each cluster, but our data does not contain additional features that can be used for that purpose. Non-systematic manual inspection of the sectors to which some of the companies belong did not show any obvious patterns either

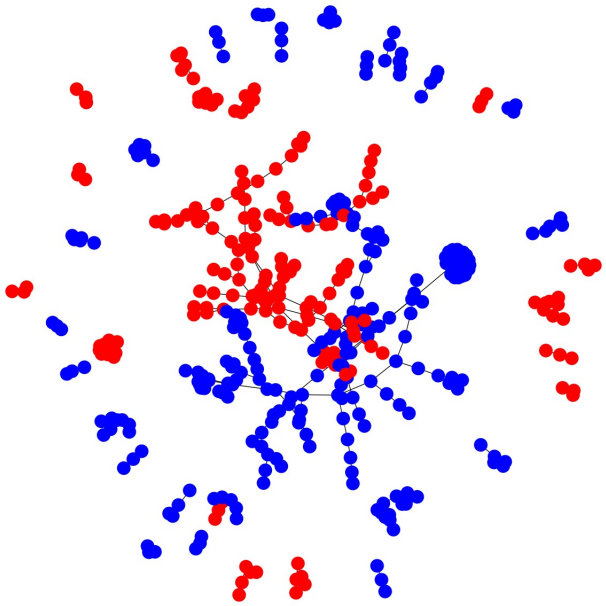

Figure 5.4: Knowledge graph subset containing clusters 13 and 15 (red vs blue)

nor can we see any substantial number of connections between the two clusters when I look at the topology of the knowledge graph for the nodes in clusters 13 and 15 (Figure 5.4). It is nevertheless the case that the majority of the rules do make use of the cluster labels, and most perform better than random in a statistically significant way even when their results are taken out of context, on their own. It is also to be noted that once I started using node embedding based cluster labels as background predicates, no rule based on the number of connections between the two companies in the pair has been learned, despite the positive results such rules showed in our previous research [77].

The lack of interpretability of specific clusters (e.g. why cluster 13 is more cointegrated than others) shows the trade-off of the hybrid approach. While the sub-symbolic embeddings capture homophily in the company network, the ILP rules shows the predictive power. However, the economic data does not marry the two approaches all the time.

Finally, while analysing the outcomes, I should mention the rules that set a minimum threshold for the number of people connected to one or both companies. When the constraint appears in conjunction with the previous type of rule, the result is rules with zero positive examples covered on the test data, which suggests these rules are likely to overfit the data.

5.4 Summary

A relational learner (in FOL) can be used to automatically find hypotheses using SEC-based information that are useful to a trader. This addresses Research Question 3.

In this chapter, I have shown how a SEC report can be used for learning rules by augmenting the set with embeddings and generating additional metadata to assist in rules learning. Combining logic learning with graph data and embeddings based cluster information showed that promising results can be achieved, which would otherwise be impossible or hard to conclude. The combination of inductive learning and deductive learning with numerical cointegration data again shows insights and adds interesting properties to my learner that are otherwise not possible with pure methods.

The results of this chapter affirm RQ3, demonstrating that a general-purpose ILP learner like Progol can successfully utilise ontology derived data and sub-symbolic embeddings to generate interpretable, profitable trading hypotheses. The combination of ILP and node embeddings with numerical clustering provides deep insights into the corporate networks that affect stock movements.

The primary contribution is the demonstration of two tier hybrid learner where unsupervised graph features are successfully consumed by a supervised symbolic engine to produce interpretable trading rules. The experimental results confirm that the system can formulate and test thousands of hypotheses automatically, identifying rules with predictive power that performs better than random selection while also providing interpretable justification for the rule unlike neural networks.

However, there is another challenge, Progol is a general ILP learner and sequential processor. The data needs to be converted and flattened into Prolog background knowledge files prior to execution, because it cannot work natively with the ontology I built. As the size of the financial ontology grows, the translation step and especially the serial hypothesis search space become computationally bottlenecked, a known limitation of most ILP systems. This limitation motivated me to the research in Chapter 6 and led me to build a specialised, concurrent ILP learner system capable of operating with ontologies that also use multi-processor or GPU for large scale ontologies using description logic and my parallel learner.

In the following chapters, I examine how this can be done using a specialised learner that operates directly on data represented as an ontology.

Table 5.1: Q3/2017 rules

coint(A,B) :- ccon2(A,54), ccon2(B,54).
coint(A,B) :- companycluster(A,12), companycluster(B,12).
coint(A,B) :- companycluster(A,14), companycluster(B,14).
coint(A,B) :- companycluster(A,13), companycluster(B,13).
coint(A,B) :- companycluster(A,6), companycluster(B,6).
coint(A,B) :- companycluster(A,5), companycluster(B,5).
coint(A,B) :- companycluster(A,16), companycluster(B,16).
coint(A,B) :- companycluster(A,3), companycluster(B,3).
coint(A,B) :- companycluster(A,17), companycluster(B,17).
coint(A,B) :- ccon2(A,8), ccon2(B,54), companycluster(A,5).
coint(A,B) :- ccon2(A,54), ccon2(B,8), companycluster(B,5).
coint(A,B) :- companycluster(A,2), companycluster(B,2).
coint(A,B) :- companycluster(A,19), companycluster(B,19).
coint(A,B) :- companycluster(A,1), companycluster(B,10).
coint(A,B) :- companycluster(A,10), companycluster(B,1).
coint(A,B) :- companycluster(A,18), companycluster(B,18).
coint(A,B) :- companycluster(A,1), companycluster(B,1).
coint(A,B) :- companycluster(A,0), companycluster(B,0).
coint(A,B) :- companycluster(A,5), companycluster(B,16).
coint(A,B) :- companycluster(A,9), companycluster(B,9).
coint(A,B) :- companycluster(A,8), companycluster(B,8).
coint(A,B) :- companycluster(A,7), companycluster(B,7).

Table 5.2: Q4/2017 rules

```
coint(A,B) :- ccon2(B,113).
coint(A,B) :- ccon2(B,66).
coint(A,B) :- ccon2(A,16), ccon2(B,16).
coint(A,B) :- ccon2(A,12).
coint(A,B) :- companycluster(A,17), companycluster(B,17).
coint(A,B) :- ccon2(A,10), ccon2(B,10).
coint(A,B) :- ccon2(B,4), companycluster(A,4).
coint(A,B) :- companycluster(A,6), companycluster(B,4).
coint(A,B) :- companycluster(B,14).
coint(A,B) :- ccon2(A,2), companycluster(B,11).
coint(A,B) :- ccon2(A,5), ccon2(B,2), companycluster(A,10).
coint(A,B) :- ccon2(A,2), companycluster(B,10).
coint(A,B) :- companycluster(A,12), companycluster(B,12).
coint(A,B) :- ccon2(A,9), ccon2(B,9).
coint(A,B) :- companycluster(A,8), companycluster(B,1).
coint(A,B) :- companycluster(A,2), companycluster(B,2).
coint(A,B) :- ccon2(B,4), companycluster(A,11).
coint(A,B) :- companycluster(B,7).
coint(A,B) :- companycluster(A,16), companycluster(B,16).
coint(A,B) :- companycluster(A,10), companycluster(B,5).
```

Table 5.3: Experiment 1 – Related Pairs

	Quarter 3	Quarter 4
Number of Companies	1948	1910
Cointegrated Pairs	652	184
Not Cointegrated Pairs	7326	3344

Table 5.4: Q3/2017 rules: results for which the χ^2 test yields $p \leq .05$ are in bold.

Rules	pos	neg	$\frac{\text{pos}}{\text{pos+neg}}$	p-value
coint(A,B) :- ccluster(B,18).	102	2327	0.04	**.0036**
coint(A,B) :- ccluster(A,3).	145	1987	0.07	**.0000**
coint(A,B) :- ccluster(B,19).	102	2690	0.04	**.0000**
coint(A,B) :- ccluster(A,2).	71	2005	0.03	.4633
coint(A,B) :- ccluster(A,0).	77	2290	0.03	.7434
coint(A,B) :- ccluster(B,16).	81	1904	0.04	**.0184**
coint(A,B) :- ccluster(A,1).	218	1956	0.10	**.0000**
coint(A,B) :- ccluster(A,8), ccluster(B,12).	3	41	0.07	.1609
coint(A,B) :- ccluster(A,12), ccluster(B,13).	10	65	0.13	**.0000**
coint(A,B) :- ccluster(A,11), ccluster(B,12).	10	112	0.08	**.0014**
coint(A,B) :- ccluster(A,10), ccluster(B,12).	3	44	0.06	.2013
coint(A,B) :- ccluster(A,7), ccluster(B,12).	14	121	0.10	**.0000**
coint(A,B) :- ccluster(A,4), ccluster(B,7).	19	238	0.07	**.0001**
coint(A,B) :- ccluster(A,7), ccluster(B,9).	8	121	0.06	**.0460**
coint(A,B) :- ccluster(A,7), ccluster(B,8).	5	115	0.04	.5163
coint(A,B) :- ccluster(A,7), ccluster(B,11).	16	123	0.12	**.0001**
coint(A,B) :- ccluster(A,7), ccluster(B,15).	4	107	0.04	.7761
coint(A,B) :- ccluster(A,7), ccluster(B,10).	8	98	0.08	**.0093**
coint(A,B) :- ccluster(A,5), ccluster(B,7).	13	112	0.10	**.0000**
coint(A,B) :- ccluster(A,6), ccluster(B,7).	23	191	0.11	**.0000**
coint(A,B) :- ccluster(A,4), ccluster(B,6).	22	143	0.13	**.0000**
coint(A,B) :- ccluster(A,4), ccluster(B,8).	5	133	0.04	.7412
coint(A,B) :- ccluster(A,4), ccluster(B,9).	6	116	0.05	.2586
coint(A,B) :- ccluster(A,4), ccluster(B,13).	18	118	0.13	**.0000**
coint(A,B) :- ccluster(A,4), ccluster(B,15).	11	105	0.09	**.0001**
coint(A,B) :- ccluster(A,4), ccluster(B,5).	14	143	0.09	**.0000**
coint(A,B) :- ccluster(A,4), ccluster(B,10).	6	101	0.06	.1425
coint(A,B) :- ccluster(A,4), ccluster(B,11).	17	124	0.12	**.0000**
coint(A,B) :- ccluster(A,5), ccluster(B,6).	9	68	0.12	**.0000**
coint(A,B) :- ccluster(A,6), ccluster(B,9).	10	95	0.10	**.0002**
coint(A,B) :- ccluster(A,6), ccluster(B,11).	9	104	0.08	**.0033**
coint(A,B) :- ccluster(A,6), ccluster(B,12).	14	101	0.12	**.0000**
coint(A,B) :- ccluster(A,6), ccluster(B,10).	10	83	0.11	**.0000**
coint(A,B) :- ccluster(A,9), ccluster(B,13).	5	61	0.08	**.0386**
coint(A,B) :- ccluster(A,11), ccluster(B,13).	13	114	0.10	**.0000**
coint(A,B) :- ccluster(A,5), ccluster(B,13).	4	63	0.06	.1830
coint(A,B) :- ccluster(A,8), ccluster(B,13).	3	45	0.06	.2155
coint(A,B) :- ccluster(A,8), ccluster(B,11).	4	40	0.09	**.0235**
coint(A,B) :- ccluster(A,8), ccluster(B,9).	2	48	0.04	.7250
coint(A,B) :- ccluster(A,8), ccluster(B,10).	1	39	0.03	.8184

Table 5.5: Q3/2017 rules (cont.)

Rules	pos	neg	$\frac{\text{pos}}{\text{pos+neg}}$	p-value
coint(A,B) :- ccluster(A,10), ccluster(B,13).	5	44	0.10	**.0046**
coint(A,B) :- ccluster(A,5), ccluster(B,15).	9	43	0.17	**.0000**
coint(A,B) :- ccluster(A,5), ccluster(B,10).	6	46	0.12	**.0005**
coint(A,B) :- ccluster(A,5), ccluster(B,9).	6	53	0.10	**.0020**
coint(A,B) :- ccluster(A,5), ccluster(B,11).	9	52	0.15	**.0000**
coint(A,B) :- ccluster(A,9), ccluster(B,11).	7	57	0.11	**.0003**
coint(A,B) :- ccluster(A,9), ccluster(B,15).	3	58	0.05	.4239
coint(A,B) :- ccluster(A,9), ccluster(B,10).	2	53	0.04	.8303
coint(A,B) :- ccluster(A,8), ccluster(B,15).	4	34	0.11	**.0090**
coint(A,B) :- ccluster(A,14), ccluster(B,15).	4	84	0.05	.4474
coint(A,B) :- ccluster(A,11), ccluster(B,15).	10	101	0.09	**.0004**
coint(A,B) :- ccluster(A,10), ccluster(B,11).	6	45	0.12	**.0004**
coint(A,B) :- ccluster(A,5), ccluster(B,8).	4	56	0.07	.1166
coint(A,B) :- ccluster(A,10), ccluster(B,10).	1	35	0.03	.9027
coint(A,B) :- ccluster(A,10), ccluster(B,15).	5	41	0.11	**.0026**
coint(A,B) :- ccluster(A,6), ccluster(B,15).	7	93	0.07	**.0268**
coint(A,B) :- ccluster(B,15), ccon2(A,4).	40	507	0.07	**.0000**
coint(A,B) :- ccluster(A,6), ccon2(A,2), ccon2(B,4).	26	694	0.04	.4652
coint(A,B) :- ccluster(A,7), ccluster(B,7), ccon2(B,3).	8	74	0.10	**.0006**
coint(A,B) :- ccluster(A,6), ccluster(B,6), ccon2(A,4).	0	60	0.00	.1636
coint(A,B) :- ccluster(B,13), ccon2(A,310).	4	25	0.14	**.0010**
coint(A,B) :- ccluster(A,10), ccon2(B,310).	3	7	0.30	**.0000**
coint(A,B) :- ccluster(A,13), ccluster(B,13), ccon2(A,5).	4	62	0.06	.1727
coint(A,B) :- ccluster(A,5), ccluster(B,5), ccon2(B,4).	0	34	0.00	.2944
coint(A,B) :- ccluster(A,9), ccluster(B,9), ccon2(B,2).	0	40	0.00	.2554
coint(A,B) :- ccluster(A,15), ccluster(B,15), ccon2(B,2).	0	37	0.00	.2740

Table 5.6: Experiment 2 – Test results

Q3/2017			
	A	¬A	Total
P	1,361	22,192	23,553
¬P	44	21,253	21,297
Total	1,405	43,445	44,850
Accuracy	50.42% ± 0.24%		
χ^2 probability	0.0000		

Chapter 6

Scalable Learning from Ontologies with a Description Logic Learner

In this chapter, I describe how a concurrent learner in description logic can be used to learn from ontologies, especially without specialist hardware (GPU based), in commodity hardware (CPU based).

6.1 Introduction

While Chapter 5 successfully demonstrated that ILP can be used to generate interpretable rules from financial reports ontology data, it also highlighted a bottleneck where ILP learners require ontology data to be converted to the syntax of the ILP system as background knowledge files. Even when translated, the ILP learners are sequential executors. This translation and execution step limits scalability when applied to large web-scale financial datasets.

This chapter addresses the scalability problem. The purpose of this chapter is to introduce a specialised, concurrent learner that operates directly on the native ontology data. I achieve this with my description logic ILP learner by modelling the ontology as matrices and tensors. Elimination of the translation and parallelising the cover set testing phase across CPU and GPU, this chapter provides methodological framework that addresses the scalability issue of the financial rules discovery, although it can be applied to any ontology with the framework.

This chapter details the development and design of P-CONNER, a concurrent description logic (DL) learner system. Although, symbolic methods offer interpretability and explainable AI, they suffer from exponential complexity of searching hypothesis space. I developed P-CONNER to overcome these limitations, mainly in the cover set computation, which is the most resourceful and expensive component of the refinement process.

The chapter is structured as follows: I will discuss the limitations of existing DL learners and the motivation for a new concurrent approach. Then I will detail the architecture and implementation of P-CONNER, showing my design choices. Then I will show the experimental evaluation of the system using benchmark dataset. I conclude by linking the scalability improvements back to the goal of financial pair trading and forecasting.

6.2 Limitations of CONNER and motivation for P-CONNER

The initial work on accelerating the cover set computation, which led to the proof-of-concept DL learner called CONNER, focused on GPU-accelerated cover set testing. While this approach successfully tapped into the great potential of General-Purpose computation on Graphics Processing Units (GPGPU), the specific implementation had certain limitations.

A key limitation of the GPU-based concurrency used in CONNER was that it tied the tool to one specific hardware platform, namely 'NVIDIA GPUs', and required the use of the CUDA programming model. This reliance on a hardware vendor and programming model restricted the portability and accessibility of CONNER.

Previous work by Algahtani and Kazakov, [81], [1] demonstrated that cover set training, the most computationally expensive component of ILP systems could be parallelised efficiently using a GPU, resulting in a proof of concept DL learner, CONNER. However one of the limitations of CONNER was that it was dependent on Windows and NVIDIA GPUs and the CUDA programming model.

To address this, I developed Python CONNER (P-CONNER). The key design and my novel contribution in P-CONNER is the abstraction of the concurrency model to make it entirely hardware-independent and OS agnostic. Rather than relying on vendor specific GPU instructions, P-CONNER evaluates into a matrix operations problem where it can be executed on a CPU with vectorised instructions or GPU (AMD or NVIDIA) CUDA

cores working on it. This allows the workload to be distributed across multiple CPU cores on commodity hardware or across generic GPUs. Although Python is not a high performance language, the native bindings and rich libraries make it the language of AI. Specifically, I use numpy library here to represent the ontology modelled as a matrix and parallel execution engine on top where it runs on the native instructions rather than the runtime abstraction. This also allows the same abstraction to run on the GPU cores without any change other than copying the matrices to GPU memory. The full source code for P-CONNER implementation described in this thesis is available from the public GitHub repository `https://github.com/certen/p-conner`. The implementation is released under the GNU Affero General Public License version 3 only.

The issues associated with previous implementations of CONNER, such as being limited to a specific hardware platform , led to P-CONNER. It was reimplemented and evaluated with a focus on its scalability. The objective was to provide full oversight over its inner workings and to make it hardware-independent both on CPU and GPU. The use of CPU concurrency is less restrictive, as most modern processors have multiple cores and threads, while not every machine has a GPU. This design choice makes the learner useful to a much greater range of potential users. It also facilitated the use of other GPUs rather than NVIDIA.

To achieve hardware independence, P-CONNER abstracts the concurrency layer from vendor specific CUDA implementation (CONNER), by utilising a vectorised approach using high-level library in Python, namely 'numba' which is a JIT compiler that translates a subset of Python and NumPy code into fast machine code. P-CONNER also abstracts GPU kernel code generation using similar abstraction, thus it is possible to run both CPU and GPU. This ensures that the speedups reported in Section 6.3 are accessible on commodity hardware to build scalable DL Learner.

Implementing P-CONNER itself borrows a refinement operator from DL-Learner's OCEL algorithm.

6.3 Concurrent Implementation of Cover Set Testing

In ILP, the process of learning is often reduced to the search for the hypothesis with the best score among a potentially rich set of alternatives produced by a refinement operator.

Cover set testing is a costly process and a critical component in ILP learners. It involves

sequentially considering all negative examples and all relevant positive examples to determine whether a hypothesis covers them. The process is usually implemented sequentially. This computation relies on set membership tests. This is a costly process and a critical component in classical ILP learners. It requires considering all negative examples and all relevant positive examples in a sequential manner.

The idea that this specific component could be significantly accelerated through concurrency was first established in 2018 [81] and resulted in a speedups of several orders of magnitude. Their work proposed a GPU-accelerated approach to the computation of the cover set for a given hypothesis expressed. They demonstrated that testing which positive and negative examples are covered can be sped up considerably through concurrency, particularly when using Description Logic (DL) for both the background knowledge and the target concept examples (+, -).

The extension of this work, which leads to P-CONNER, builds by extending the concurrency-accelerated cover set computation to include both CPU- and GPU-based concurrency, rather than being limited to specialised GPU hardware.

Previous work demonstrated that cover set testing, a core component of ILP algorithms, could be parallelised very efficiently using a GPU, resulting in speed-ups of several orders of magnitude.

This enhanced concurrent cover set testing can then be integrated with a refinement operator and search strategy to create a complete concurrent ILP learner [1].

The implementation of learning query is shown below on listing A.1. For a given positive and negative concept, the learning query identifies by querying the ontology for the positive and negative examples. This returns the number of samples hit by the query which are then used against the scoring function to guide the search to find the hypothesis.

6.3.1 Refinement Operator and Search

P-CONNER operates within Description Logic (DL) and explores the hypothesis space using a top-down search strategy.

For the generation of candidate hypotheses, P-CONNER uses the refinement operator from the DL-Learner, specifically the OCEL operator. This operator has been extensively

described by Lehmann [82] . My implementation in P-CONNER utilises this same operator for guiding the exploration of the hypothesis space.

6.3.2 Scoring Function and Search Guidance

P-CONNER employs a top-down search strategy to explore the hypothesis space. This search process is informed and guided by a scoring function, which determines the direction of exploration.

The scoring function used in P-CONNER is based on the one used in the DL-Learner with its OCEL learning algorithm.

As in CONNER, P-CONNER extends this function to incorporate the length of the hypothesis and its depth in the search tree, starting from zero. The resulting scoring function in P-CONNER is defined as:

$$\text{P-CONNER score}(N) = 10 \cdot \text{ocel\_score}(N) - \text{length}(C) - \text{depth}(N)$$

This extended score guides the search algorithm in finding hypotheses that are complete and consistent with the data.

6.3.3 P-CONNER Interface

P-CONNER has a Web interface where users can run sample datasets or their own datasets. This interface is available online at `https://p-conner.herokuapp.com` Figure 6.1. Users can utilise the interface to reproduce the results obtained from the Michalski's trains task. Additionally, the interface enables users to learn from another data set of their choosing. The user interface is a three column layout. The first column is related to the ontology where the user can choose an ontology or upload an ontology of their own. They can then tag the positive and negative example class name in the interface. P-CONNER will show all the available classes in the positive and negative dropdowns. The middle column is for fine tuning the learning parameters where the user can choose the maximum rule length to be evaluated before giving up, the maximum learning time befpre exiting their learning (it may go on forever or until the solution is found), minimum accuracy for the learning algorithm to count an hypothesis to be counted as a solution. The last column is the result

of learning whether it is succesful or not with the potential results.

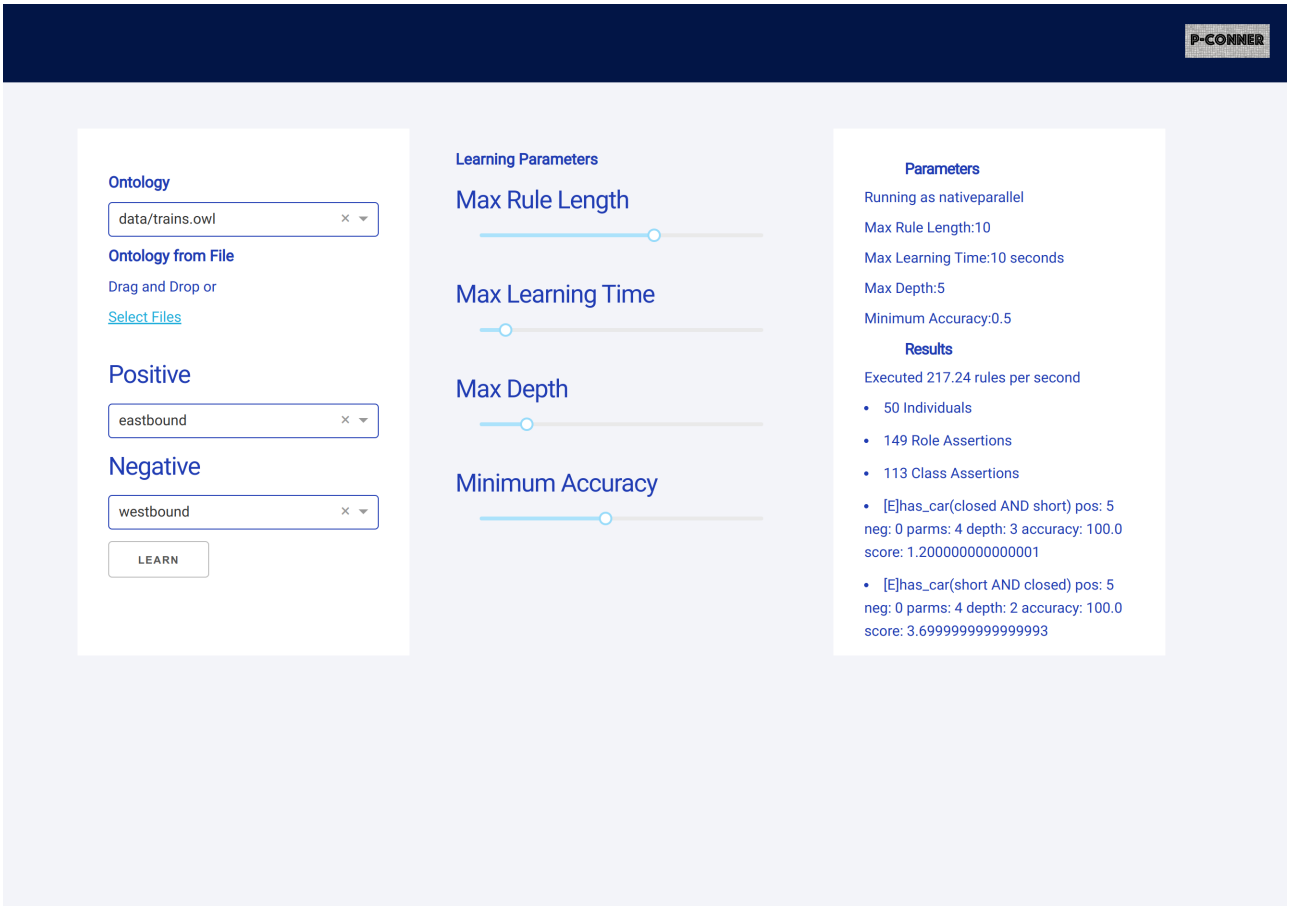

Figure 6.1: Running P-CONNER on Michalski's trains

To validate the correctness and scalability of P-CONNER, I evaluated my ILP system using the classic Michalski's trains dataset. This dataset is universally recognised benchmark in ILP research for testing structural and relational learning.

Although it may look unrelated to financial domain, Michalski's trains dataset serves as an ideal case for the SEC financial ontology built in Chapter 3. The dataset can be very small for quick evaluation or it can be scaled. Both domains require the learner to navigate highly connected, multi-model relational graphs. For instance, a train has a car which has a shape and load, which mirrors a company which has a director who sits on a board of multiple companies). P-CONNER can successfully discover complex relational rules in the trains benchmark. In parallel, I demonstrate the structural capability to traverse the dataset at scale and verify.

P-CONNER implements the cover set testing directly within a matrix-based representation, which is the main enabler of the parallelism by formulating the problem as matrix

operations. By representing the ontology as a Boolean matrix, I trade off some logic expressiveness for the ability to test hypotheses in parallel across millions of individuals.

6.4 Experimental Evaluation

The problem that P-CONNER is trying to solve has been converted to an embarrassingly parallel problem and application. Hence, the evaluation of P-CONNER demonstrates the execution stages and its speed. The experiments were conducted on a PC equipped with an AMD 3900x processor, 16GB of RAM, and an NVIDIA GeForce RTX 2080 Ti GPU with 4352 CUDA cores. Similar results could be reached through the use of either GPU or CPU concurrency. Given that CPU-based concurrency is less restrictive as most modern processors have multiple cores and threads, while not every machine has a GPU, the reported results are based on the use of CPU concurrency, making them relevant to a wider number of potential users.

P-CONNER was first tested for correctness using the original Michalski's trains dataset. This dataset contains 50 individuals, 149 role assertions, and 113 class assertions. P-CONNER found a solution path, one of which had the highest score and is equivalent to a solution produced by the DL-Learner using the OCEL refinement operator and Pellet reasoner. This search path to a solution is illustrated in Figure 6.2.

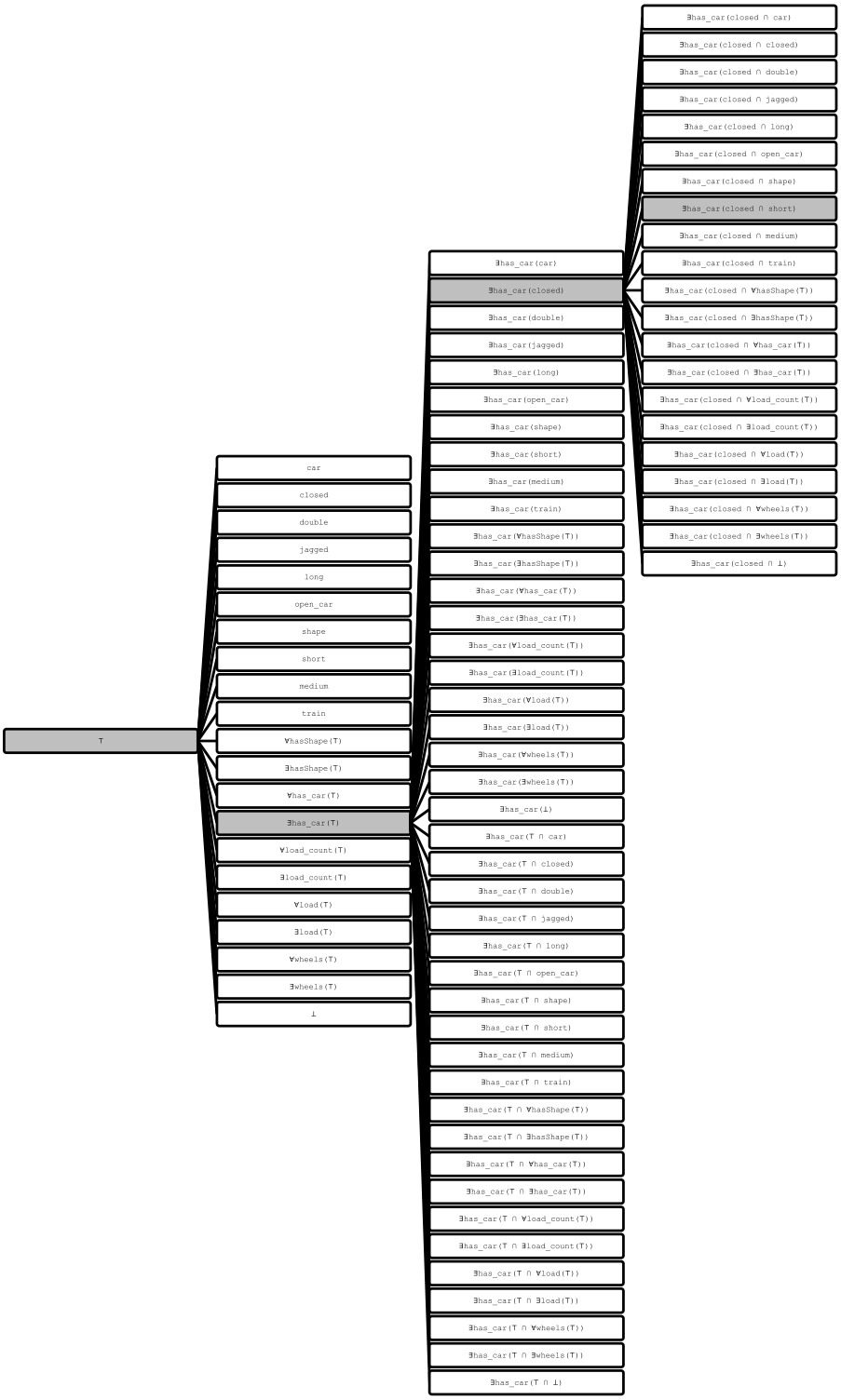

Figure 6.2: A sample refinement path to a solution: Michalski's trains. Reprinted with the author's permission [1]

A second experiment utilised a larger, generated dataset in the style of Michalski's trains (specifically, trains of length 4). This dataset is significantly larger, containing 21,156 unique trains (5,184 eastbound and 15,972 westbound), with 105,780 individuals and 148,092 assertions. Using this entire dataset, the overall learning time was measured at 58,233µs, which is less than 0.06ms.

The final set of experiments involved learning the concept of "Father" from a synthetically generated dataset. This dataset is based on individuals belonging to either the Man or Woman classes and appearing in the 'has child' role. The underlying rule to be learned is that a father is a man who is someone's parent. The size of this dataset was varied as listed in Table 6.1. This time only includes the hypothesis search and testing, excluding input parsing and reasoner usage. The results, plotted in Fig. 6.3, show that the time complexity growth rate is O(n) $\leq$ n over the studied range of dataset sizes. Doubling the largest dataset size (to 480,000 individuals) resulted in a sharp, unstable increase in execution time, potentially due to reaching the memory limitations of the hardware platform.

Individuals	Role Assertions	Class Assertions	Time [μs]
240006	160004	960009	13844
120006	80004	480009	8345
60006	40004	240009	6549
30006	20004	120009	4984
15006	10004	60009	4206
7500	5000	29985	3670
3750	2500	14985	3390

Table 6.1: Learning the Father concept

Table 6.1, and plot the time the learner takes to find the underlying rule: a father is a man who is someone's parent (see Fig. 6.3). As illustrated in Fig. 6.3, the execution time shows a linear growth rate relative to the number of individuals. The results are baselined against DL-Learner [82], a standard description logic learning ILP system for ontologies. Although DL-Learner is a state of the art accurate ILP learner, it is not known for its scalability. In P-CONNER, the scalability is achieved by the cover set algorithm (which is an embarrassingly parallel algorithm) where each individual's test with a hypothesis can be verified independently at scale. The observation of 4.8 log individuals confirms P-CONNER's scalability.

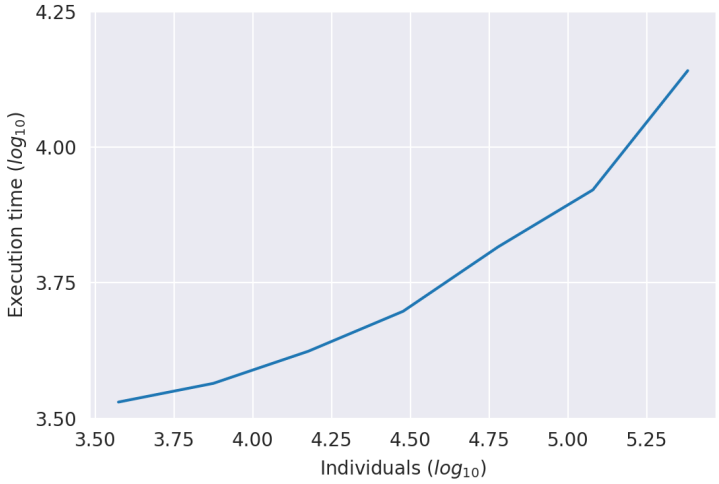

Figure 6.3: Execution times [μs] from Table 6.1: Learning the Father concept

6.5 P-CONNER

P-CONNER is a hardware-independent implementation of concurrent cover set testing for Description Logic learning, addressing the limitations of previous GPU-specific approaches while maintaining the computational advantages of parallelisation in both CPU and GPU (not vendor specific).

6.5.1 Algorithm Overview

The core learning algorithm in P-CONNER implements a best-first greedy search strategy, leveraging concurrent cover set evaluation to efficiently explore the hypothesis space. Algorithm 1 is the main restriction operator for evaluating the hypothesis.

Algorithm 1 Existential Restriction Cover Set ($\exists Role$.Concept)

procedure ParallelExistentialRestriction(Concept,Role)

```
Given:
M: Boolean 2D matrix of size (individuals x concepts)
R: Integer 2D matrix of size (individuals x 3) //Storing (subject,role,object)
H: hash table // Role -> (Offsets for first and last entries in R)

Concept: Pointer to a column in M
Role: Integer

result: Boolean 1D array of size NumberOfIndividuals

Do:
parallel_foreach thread T_j
| for each individual i in result
| | set result[i] = 0
| endfor
endfor
set role_range := H[Role] //Retrieve the range of Role assertions in R from H
parallel_foreach thread T_j
| foreach roleAssertion in role_range
| | set IndvA := R[row(roleAssertion),1] // first column of roleAssertion
| | set IndvB := R[row(roleAssertion),3] // third column of roleAssertion
| | set local_result := M[row(IndvB),Concept]
| | atomicMAX(result[row(IndvA)],local_result)
| endfor
endfor
```

return result(1..numberOfIndividuals)
end procedure

6.5.2 Concurrent Cover Set Testing

The computational bottleneck of cover set evaluation is addressed through data-parallel algorithms that can leverage both CPU and GPU architectures. Algorithm 2 shows the parallel conjunction operator for evaluating the refinements as potential hypothesis.

Algorithm 2 For a Boolean matrix M (individuals $\times$ concepts)

procedure ParallelConceptConjunctionCoverSet
  $S \leftarrow$ list of concepts C_i
  for all thread T_i in parallel **do**
    for all individual I_j in thread T_i **do**
      $result(row(I_j)) \leftarrow 1$
      for all concept C_k in set S **do**
        $result(row(I_j)) \leftarrow result(row(I_j)) \wedge M(row(I_j), column(C_k))$
        if $result(row(I_j)) = 0$ **then**
          break ▷ only used with lazy evaluation
        end if
      end for
    end for
  end for
  return Boolean array: $result(1..numberOfIndividuals)$
end procedure

6.5.3 Implementation Details

The learning loop is shown below in listing 3. It is a top-down search strategy that uses a refinement operator to generate refinements of the current hypothesis. The refinements are then evaluated using the concurrent procedure. The best scored refinement drives the next search step. The process continues until the best hypothesis is found or exhausted all possible refinements within the limits that are defined.

Algorithm 3 Core Learning Loop (Top-Down Search)

```
1: Initialise priority queue OpenList and set ClosedList
2: Generate the most general concept ⊤                                  ▷ Thing
3: Add ⊤ to OpenList with Score = 0
4: while Learning is active and OpenList is not empty do
5:     Current_Rule ← Pop highest scoring rule from OpenList
6:     Add Current_Rule to ClosedList
7:     Refinements ← Generate candidate rules using Refinement Operator on
   Current_Rule
8:     for Candidate ∈ Refinements do
9:         if Candidate ∉ ClosedList then
10:            Execute Concurrent Cover Set Evaluation on Candidate     ▷ Algorithm 2
11:            Calculate Accuracy and Score                             ▷ Algorithm 4
12:            if Accuracy ≥ Minimum_Accuracy_Threshold then
13:                Save Candidate to Results
14:                Push Candidate to OpenList based on Score
15:            end if
16:        end if
17:    end for
18: end while
```

6.5.4 Scoring Function

P-CONNER employs a multi-criteria scoring function that balances accuracy, complexity, and information gain:

Algorithm 4 Score Calculation

Require: Hypothesis or rule h

Ensure: Score value for h

1: $ocel \leftarrow \text{OCEL}(h)$
2: $paramCount \leftarrow \text{params}(h)$
3: $ruleDepth \leftarrow \text{depth}(h)$
4: $score(h) \leftarrow 10 \times ocel - paramCount - ruleDepth$
5: **return** $score(h)$

where the OCEL (Ontology-based Concept Evaluation Learning) component is defined as:

$$\text{OCEL}(h) = \text{accuracy}(h) + 0.5 \times \text{accuracy\_gain}(h) - 0.02 \times |\text{concepts}| \tag{6.1}$$

It is somewhat similar to the one used in DL-Learner with its OCEL algorithm, which is catered to the ontology context. The scoring function is shown below in Listing A.3.

The concurrent query execution leverages data parallelism through matrix-based representations and parallel reduction operations, enabling efficient processing of large-scale ontological datasets while maintaining the expressiveness of Description Logic reasoning.

6.6 Summary

This chapter introduced P-CONNER, a concurrent description logic learner that is hardware independent (CPU and any GPU) and a scalable ILP system. By parallelising the cover set testing phases across standard CPU vectorised instructions, P-CONNER reduces the time required to evaluate relational hypotheses on large datasets. I showed that the performance improvements of symbolic learning using existing controlled datasets.

To answer RQ3, the primary contribution is the parallel abstraction of the computation that is platform and vendor agnostic, so that existing parallel algorithms can be applied from the early work in order to build ILP learner. The evaluation of synthetic and benchmark datasets proved that P-CONNER can efficiently search through automatically formulated hypotheses.

The scalable architecture provides the missing link between large SEC financial ontologies extracted in Chapter 3 and the pair-trading strategy rules that are discovered in Chapter 5. With rule discovery happening directly with the ontology format without any translation, P-CONNER will help to minimise sampling and work with large datasets for automated financial forecasting.

Chapter 7

Conclusions

7.1 Summary of Findings and Research Questions

This thesis investigates the automated processing of multimodal financial data, SEC reports, and time series, with the goal of extracting actionable information to develop pair trading strategies. The reports are complex and large numbered, this research tries to make sense of these financial reports by transforming them into structured, machine-readable knowledge that could be leveraged by advanced machine learning techniques.

The research successfully addressed three core questions:

- Research Question 1: Is it possible to develop an automated procedure to extract structured information from SEC reports that can be represented as an ontology?

  Chapter 3 demonstrated the feasibility of such a procedure by detailing the development of the XBRL Downloader and parser. This system successfully collected and transformed semi-structured XBRL data from SEC Forms 3, 4, and 5 into a structured ontology, representing companies and their personnel links. This formed the foundational financial ontology used in subsequent chapters.

- Research Question 2: Can we show that the information contained in the ontology extracted from the SEC reports is relevant to stock market traders (specifically for pair trading)?

  Chapter 4 provided an affirmative answer by applying the constructed ontology to the domain of pair trading. The experiments showed that using ontology-derived links

(specifically, companies sharing personnel) significantly increased the probability of identifying cointegrated stock pairs compared to random selection. For instance, restricting the search space to company pairs with at least two shared links consistently yielded better cointegration accross consecutive quarters. In Q2 2017, this ontological filtering achieved a 10.38% cointegration probability, compared to 6.81% baseline for random pairs. This demonstrated the practical relevance of the ontological information for a specific trading strategy, highlighting its utility in improving the efficiency of pair selection.

- Research Question 3: Is it possible to use a machine learning algorithm (ILP) to learn from data represented as an ontology in order to search through automatically formulated hypotheses and test their relevance to a financial forecasting-related application?

  This question was addressed in two chapters. Chapter 5 showcased the use of the Progol ILP learner, augmented with node2vec embeddings and k-means clustering, to learn rules for predicting stock pair cointegration from the SEC ontology. The learnt rules demonstrate predictive power, effectively filtering candidate pairs. Subsequently, Chapter 6 introduced P-CONNER, a novel concurrent ILP learner in Description Logic designed to address the scalability challenges of learning from large ontologies. While P-CONNER's direct application to the SEC ontology was not evaluated, the development and successful evaluation on benchmark datasets show promising results for scalable learning from ontologies, as initially demonstrated in Fig. 6.3.

7.2 Contribution to Knowledge

This research makes several substantial original contributions to the fields of Artificial Intelligence and Financial Engineering :

Hybrid Machine Learning for Pair Trading : This thesis introduces a novel combination of symbolic machine learning (ILP) and sub-symbolic methods (graph embeddings) applied to financial pair trading. By merging structural ontology relations with numerical cointegration data, the system successfully discovers interpretable, statistically significant trading rules that outperform pure quantitative methods.

Automated SEC Ontology Construction : The automated extraction and transformation of

large quantities of semi-structured XBRL filings into a uniform, semantically rich financial ontology provides a new methodology for traversing the networks of employees that is not possible using standard relational or XML databases.

Hardware Independent Scalable Learning : The design and implementation of P-CONNER contributes a scalable, concurrent description logic learner. Parallelising the computationally expensive cover set testing phase across both CPU and GPU, removes the hardware dependencies and operating system dependencies of the previous systems. Matrix-based description logic learner achieves scalability for symbolic learning.

The methodologies developed in this thesis have wider implications beyond financial pair trading. The automated extraction pipeline can be utilised to map graphs in other domains, such as drug discovery or compliance reporting. Additionally, P-CONNER's hardware independent, operating system agnostic concurrent approach provides support for the semantic work loads and research for large knowledge graph and ontologies without specialised hardware. Interpretability of the rules generated by my ILP learner, also aligns with explainable AI fields, such that the decisions can be understood by other analysts.

7.3 Limitations of the Study

While the hypotheses were successfully validated, there are some limitations. First, the success of the hybrid trading strategy is heavily dependent on the quality and completeness of the underlying SEC ontology. Any changes to the SEC's reporting schema or errors in entity deduplication during the parsing phase could degrade the topological structure used by the learner. Secondly, financial markets are affected by multiple factors, including macroeconomic factors that are not captured by the insider trading reports and also personal factors of directors (that affect the performance of the company) may not be captured. The current ontology is built with structured and semi-structured data, leaving the unstructured free-text sections of other filings un-analysed. Finally, while P-CONNER was successfully validated the scalability of standard ILP benchmarks (Michalski's trains), the direct integration and benchmarking against my own SEC financial ontology remains to be trialled with my learner which is potential area of further research.

7.4 Future Work

Key limitations include the current focus on structured and semi-structured data from SEC reports, where unstructured textual data remains largely unexploited. The challenges in the financial interpretability of clusters derived from node2vec embeddings (Chapter 5. Another challenge is the full application and comparative analysis of P-CONNER on the developed SEC financial ontology, which remains a prospective task.

The aim is to extend the capabilities of the developed methodologies and further explore the intersection of AI and finance.

1. Real-time Data Integration: Incorporate real-time market data and continuously update the ontology.

2. Add relations extracted from unstructured data and free text rather than tabular data using Large Language Models (LLMs):

   The current financial ontology primarily leverages structured and semi-structured data from SEC reports. A significant extension would be to incorporate Large Language Models (LLMs). This could uncover more nuanced relationships, sentiment indicators, and risk factors, enriching the ontology with free text and providing deeper insights for financial forecasting and trading strategy development.

3. Dynamic Ontology Evolution and Financial Knowledge Representation:

   Financial markets and reporting standards change constantly because of market events and regulatory changes. Future work can address these changes by developing methods for ontology evolution, including more robust handling of XBRL schema changes and data inconsistencies. The ontology could also be expanded to model a wider array of financial entities and relationships beyond shared personnel, such as supply chain connections, competitive relationships or macroeconomic influences.

4. Full Application and Benchmarking of P-CONNER on Financial Data:

   A crucial next step is the direct application of the P-CONNER system (Chapter 6) to the SEC financial ontology developed in this thesis. This would involve a comprehensive evaluation of its performance in learning financial trading rules, directly comparing its scalability, rule quality, and computational efficiency against the Progol-based approach used in Chapter 5.

5. Validation and Real-World Deployment:

   The trading strategies and learnt rules should be subjected to further validation across diverse market conditions, longer time horizons, and different geographical markets. This would involve back-testing and potentially paper trading to assess their robustness and practical viability before any consideration of real-world deployment.

By pursuing these future research directions, the work initiated in this thesis can continue to contribute to the development of more sophisticated and explainable learner systems for financial modelling and decision-making.

List of References

[1] E. Algahtani and D. Kazakov, "Conner: A concurrent ILP learner in description logic," in *Proc. of the* 29^{th} *International Conf. on Inductive Logic Programming*, 2020.

[2] C. Erten, E. Algahtani, I. Fairlamb, E. Garcia-Padilla, J. Lynam, S. Manandhar, J. Slattery, and D. Kazakov, "Unsupervised learning of functional groups for computational chemistry," 2019.

[3] D. Kazakov and C. Erten, *Inductive Logic Programming: 29th International Conference, ILP 2019, Plovdiv, Bulgaria, September 3–5, 2019, Proceedings*. Springer Nature, 2020.

[4] C. Erten, N. Chotai, and D. Kazakov, "Pair trading with an ontology of sec financial reports," in *IEEE Symposium on Computational Intelligence for Financial Engineering and Economics (CIFEr)*, 2020.

[5] C. Erten and D. Kazakov, "Ontology graph embeddings and ilp for financial forecasting," in *30th International Conference on Inductive Logic Programming*, 2021.

[6] Wikipedia contributors, "XBRL." `https://en.wikipedia.org/wiki/XBRL`, 2024. [Online; accessed 2024-10-02].

[7] U. Securities and E. Commission, "Sec.gov — filings & forms." `https://www.sec.gov/edgar.shtml`, 2021. Accessed: 2021-07-01.

[8] J. Y. Campbell, A. W. Lo, A. C. MacKinlay, and R. F. Whitelaw, "The econometrics of financial markets," *Macroeconomic Dynamics*, vol. 2, no. 4, pp. 559–562, 1998.

[9] P. M. Garber, "Tulipmania," *Journal of political Economy*, vol. 97, no. 3, pp. 535–560, 1989.

[10] J. Hull, S. Treepongkaruna, D. Colwell, R. Heaney, and D. Pitt, *Fundamentals of futures and options markets*. Pearson Higher Education AU, 2013.

[11] N. Huck and K. Afawubo, "Pairs trading and selection methods: Is cointegration superior?," *Applied Economics*, vol. 47, no. 6, pp. 599–613, 2015.

[12] B. Inder, "Estimating long-run relationships in economics: A comparison of different approaches," *Journal of Econometrics*, vol. 57, no. 1, pp. 53–68, 1993.

[13] M. Whistler, *Trading pairs: capturing profits and hedging risk with statistical arbitrage strategies*, vol. 216. John & Sons: Wiley, 2004.

[14] A. Brim, "Deep reinforcement learning pairs trading with a double deep q-network," in *2020 10th annual computing and communication workshop and conference (CCWC)*, pp. 0222–0227, IEEE, 2020.

[15] G. R. Madhavaram, "Statistical arbitrage using pairs trading with support vector machine learning," 2013.

[16] H. Qu, M. S. Nascimento, N. N. Qomariyah, and D. L. Kazakov, "Integrating time series with social media data in an ontology for the modelling of extreme financial events," *in LREC*, vol. 2016, pp. 57–63, 2016.

[17] C. J. Granger, "Developments in the study of cointegrated economic variables," *Oxford Bulletin of Economics and statistics*, vol. 48, no. 3, pp. 213–228, 1986.

[18] H.-L. Han and M. Ogaki, "Consumption, income and cointegration," *International Review of Economics & Finance*, vol. 6, pp. 107–117, 1997.

[19] M. P. Murray, "A drunk and her dog: an illustration of cointegration and error correction," *The American Statistician*, vol. 48, no. 1, pp. 37–39, 1994.

[20] S. Johansen, "Statistical analysis of cointegration vectors," *Journal of Economic Dynamics and Control*, vol. 12, no. 2, pp. 231–254, 1988.

[21] R. F. Engle and C. W. J. Granger, "Co-integration and error correction: Representation, estimation, and testing," *Econometrica*, vol. 55, no. 2, pp. 251–276, 1987.

[22] S. Johansen, "Estimation and hypothesis testing of cointegration vectors in gaussian vector autoregressive models," *Econometrica*, vol. 59, no. 6, pp. 1551–1580, 1991.

[23] J. Perktold, S. Seabold, and J. Taylor, "Statsmodels.tsa.stattools.coint, [online]," *Available: (visited on)*, vol. 12, 2017.

[24] R. A. Maller, G. Müller, and A. Szimayer, "Ornstein–uhlenbeck processes and extensions," *Handbook of financial time series*, pp. 421–437, 2009.

[25] S. Johansen, "Cointegration: Overview and development," *Handbook of financial time series*, pp. 671–693, 2009.

[26] A. Banerjee, J. Dolado, and R. Mestre, "Error-correction mechanism tests for cointegration in a single-equation framework," *Journal of time series analysis*, vol. 19, no. 3, pp. 267–283, 1998.

[27] J. Melton and A. R. Simon, *SQL: 1999: understanding relational language components*. Elsevier, 2001.

[28] I. Robinson, J. Webber, and E. Eifrem, *Graph databases: new opportunities for connected data*. " O'Reilly Media, Inc.", 2015.

[29] O. S. Technologies, "What is a triple?." `https://www.oxfordsemantic.tech/faqs/what-is-a-triple/`, 2023. Accessed: 2023-08-01.

[30] R. Angles, "The property graph database model.," in *AMW*, 2018.

[31] T. Mitchell, *Machine Learning (McGraw-Hill International Edit)*. McGraw-Hill, Jan. 1997.

[32] L. Sterling and E. Y. Shapiro, *The art of Prolog: advanced programming techniques*. MIT press, 1994.

[33] J. W. Lloyd, *Foundations of logic programming*. Science & Business Media: Springer, 2012.

[34] Z. Manna and R. J. Waldinger, "A deductive approach to program synthesis," *ACM Trans. Program. Lang. Syst.*, vol. 2, no. 1, pp. 90–121, 1980.

[35] D. Bergmann, "What is vector embedding?." `https://www.ibm.com/think/topics/vector-embedding`, 2023. Accessed: 2023-08-01.

[36] A. Bordes, J. Weston, R. Collobert, and Y. Bengio, "Learning structured embeddings of knowledge bases," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 25, pp. 301–306, 2011.

[37] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks.," in *KDD*, 2016.

[38] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: analysis, applications, and prospects," *IEEE transactions on neural networks and learning systems*, vol. 33, no. 12, pp. 6999–7019, 2021.

[39] A. Taheri, K. Gimpel, and T. Berger-Wolf, "Learning graph representations with recurrent neural network autoencoders," *KDD Deep Learning Day*, 2018.

[40] D. Kazakov and S. Manandhar, "Unsupervised learning of word segmentation rules with genetic algorithms and inductive logic programming.," *Mach. Learn.*, vol. 43, no. 1/2, pp. 121–162, 2001.

[41] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer, and C. Bizer, "Dbpedia—a large-scale, multilingual knowledge base extracted from wikipedia," *Semantic Web Journal*, vol. 6, 2014.

[42] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: a collaboratively created graph database for structuring human knowledge," in *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, (New York, NY, USA), p. 1247–1250, Association for Computing Machinery, 2008.

[43] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang, "Knowledge vault: a web-scale approach to probabilistic knowledge fusion," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, (New York, NY, USA), p. 601–610, Association for Computing Machinery, 2014.

[44] T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic web," *Scientific american*, vol. 284, no. 5, pp. 34–43, 2001.

[45] S. Bechhofer, "Owl web ontology language reference." `https://www.w3.org/TR/owl-ref/`, 2004. Accessed: 2023-08-01.

[46] G. Xiao, S. Heymans, and T. Eiter, "Drew: a reasoner for datalog-rewritable description logics and dl-programs," in *Proc. 1st Int. Workshop on Business Models, Business Rules and Ontologies (BuRO'10)*, pp. 1–14, 2010.

[47] Y. Kazakov, M. Krötzsch, and F. Simancik, "Elk reasoner: architecture and evaluation.," in *ORE*, 2012.

[48] D. Tsarkov and I. Horrocks, "Fact++ description logic reasoner: System description," in *International joint conference on automated reasoning*, pp. 292–297, Springer, 2006.

[49] B. Glimm, I. Horrocks, B. Motik, G. Stoilos, and Z. Wang, "Hermit: an owl 2 reasoner," *Journal of automated reasoning*, vol. 53, no. 3, pp. 245–269, 2014.

[50] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. D. Melo, C. Gutierrez, S. Kirrane, J. E. L. Gayo, R. Navigli, S. Neumaier, *et al.*, "Knowledge graphs," *ACM Computing Surveys (Csur)*, vol. 54, no. 4, pp. 1–37, 2021.

[51] P. Hohenecker and T. Lukasiewicz, "Ontology reasoning with deep neural networks," *Journal of Artificial Intelligence Research*, vol. 68, July 2020.

[52] B. N. Grosof, I. Horrocks, R. Volz, and S. Decker, "Description logic programs: Combining logic programs with description logic," in *Proceedings of the 12th international conference on World Wide Web*, pp. 48–57, 2003.

[53] J. Lehmann and J. Voelker, *An introduction to ontology learning*. AKA/IOS Press, 2014.

[54] D. Oliveira, R. Sahay, and M. d'Aquin, "Leveraging ontologies for knowledge graph schemas.," in *KGB@ ESWC*, pp. 24–36, 2019.

[55] I. Horrocks, "Owl: A description logic based ontology language," in *International conference on principles and practice of constraint programming*, pp. 5–8, Springer, 2005.

[56] S. H. Muggleton and L. D. Raedt, "Inductive logic programming: Theory and methods," *Journal of Logic Programming*, vol. 19, p. 20, 1994.

[57] S. Muggleton and C. Feng, "Efficient induction of logic programs," in *Proceedings of the Turing Institute*, pp. 368–381, 1990.

[58] R. Quinlan, "Learning logical definitions from relations," *Machine Learning*, vol. 5, pp. 239–266, 1990.

[59] S. Muggleton, "Inverse entailment and progol," *New Generation Computing*, vol. 13, no. 3, pp. 245–286, 1995.

[60] A. Srinivasan, "The aleph manual," 2001.

[61] M. Califf and R. Mooney, "Advantages of decision lists and implicit negatives in inductive logic programming," *New Generation Computing*, vol. 16, pp. 263–281, 1998.

[62] J. A. Robinson, "A machine-oriented logic based on the resolution principle," *Journal of the ACM (JACM)*, vol. 12, no. 1, pp. 23–41, 1965.

[63] S. Muggleton, "Inverse Entailment and Progol," *New Generation Computing, Special issue on Inductive Logic Programming*, vol. 13, no. 3-4, pp. 245–286, 1995.

[64] S. H. Muggleton, D. Lin, N. Pahlavi, and A. Tamaddoni-Nezhad, "Meta-interpretive learning: application to grammatical inference," *Machine learning*, vol. 94, no. 1, pp. 25–49, 2014.

[65] D. Kazakov and D. Kudenko, "Machine learning and inductive logic programming for multi-agent systems," in *ECCAI Advanced Course on Artificial Intelligence*, pp. 246–270, Springer, 2001.

[66] S.-Å. Tärnlund, "Horn clause computability," *BIT*, vol. 17, no. 2, pp. 215–226, 1977.

[67] H. Blockeel, L. Dehaspe, B. Demoen, G. Janssens, J. Ramon, and H. Vandecasteele, "Executing query packs in ilp," in *International Conference on Inductive Logic Programming ILP 2000*, pp. 60–77, 2000.

[68] H. Ohwada and F. Mizoguchi, "Parallel execution for speeding up inductive logic programming systems," in *Proceedings of the 9th International Workshop on Inductive Logic Programming*, pp. 277–286, 1999.

[69] M. Krötzsch, F. Simancik, and I. Horrocks, "Description logics," *IEEE Intelligent Systems*, vol. 29, no. 1, pp. 12–19, 2013.

[70] "Python edgar." `https://pypi.org/project/python-edgar/`. Accessed on 2022/08/30.

[71] "Edgar crawler." `https://github.com/lefterisloukas/edgar-crawler`. Accessed on 2022/08/30.

[72] N. Huck and K. Afawubo, "Pairs trading and selection methods: is cointegration superior?," *Applied Economics*, vol. 47, no. 6, pp. 599–613, 2015.

[73] R. Aroussi, "yfinance library," 2020. `https://github.com/ranaroussi/yfinance`. Accessed: 2020/05/09.

[74] G. Van Rossum, "random library, the python library reference, release 3.8.2," 2020. `https://docs.python.org/3.8/library/random.html`. Accessed: 2020/08/05.

[75] N. Mantel, "Evaluation of survival data and two new rank order statistics arising in its consideration," *Cancer Chemotherapy Reports*, vol. 50, pp. 63–70, 1966.

[76] D. Kazakov and S. Manandhar, "Unsupervised learning of word segmentation rules with genetic algorithms and inductive logic programming," *Machine Learning*, vol. 43, pp. 121–162, April 2001.

[77] C. Erten, N. Chotai, and D. Kazakov, "Pair trading with an ontology of SEC financial reports," in *The 2020 IEEE Symposium Series on Computational Intelligence*, 2020.

[78] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2016.

[79] B. DuCharme, *Learning SPARQL*. O'Reilly Media, second ed., 2013.

[80] S. Muggleton, "Learning from positive data," in *Selected Papers from the 6th International Workshop on Inductive Logic Programming*, pp. 358–376, 1996.

[81] E. Algahtani and D. Kazakov, "Gpu-accelerated hypothesis cover set testing for learning in logic," in *CEUR Proceedings of the 28th International Conference on Inductive Logic Programming*, CEUR Workshop Proceedings, 2018.

[82] L. Bühmann, J. Lehmann, and P. Westphal, "Dl-learner – a framework for inductive learning on the semantic web," *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 39, pp. 15–24, 2016.

Appendix A

Code

Listing A.1: Core Learning with query

```
self.ontology.ontologyConceptTree.posC = self.positiveConceptName
self.ontology.ontologyConceptTree.negC = self.negativeConceptName

qr = self.query(self.positiveConceptName)
print("positive examples(\'{}\') pos: {} neg: {}".format(self.
↪ positiveConceptName, qr.pos, qr.neg))
self.positiveNUM = qr.pos

qr = self.query(self.negativeConceptName)
print("negative examples(\'{}\') pos: {} neg: {}".format(self.
↪ negativeConceptName, qr.pos, qr.neg))
self.negativeNUM = qr.neg

self.timetools.start()

if self.testRule != '':
    print("Testing...")
    qr = self.query(self.testRule)
    self.timetools.sample()
    print("Rule:(\'{}\') pos: {} neg: {}".format(self.testRule, qr.pos, qr.
↪ neg))

    self.timetools.averageLog()
    return
print("Learning...")
resultStr = []
```

```
    resultStr.append(f'{len(self.onto.instances)} Individuals')
    resultStr.append(f'{len(self.ontology.roleAssertions)} Role Assertions')
    resultStr.append(f'{len(self.onto.classAssertions)} Class Assertions')

    print("----------------------------")
    for r in resultStr: print(r)
    print("----------------------------")
    self.learning = True

    # this is single thread for testing
    # sampling

    if self.learningParams.debug:
        self.learning_thread(self.positiveNUM, self.negativeNUM)
    else:
        # separate thread to stop
        self.startStopLearning(self.positiveNUM, self.negativeNUM)

    print(f'--------------------------Learning [STOPPED] {self.timetools.
    ↪ averageLog()}')
```

Listing A.2: Core Learning Loop Implementation

```
def learning_thread(self, positiveNUM, negativeNUM):
    closedList = set()
    openlist = PriorityQueue()

    # Initialize with most general concept
    rn = RuleNode()
    rn.rule = self.ontology.thing.name
    rn.score = 0

    openlist.put((rn.score, rn, Hypothesis([rn.rule])))

    while self.learning:
        score, bestRule, hypo = openlist.get()
        closedList.add(bestRule.rule)

        # Generate refinements using refinement operator
        hypothesis = self.refinementOperator.generateRefinements(
            bestRule.rule, hypo.refinements)
```

```
        for hypo in hypothesis:
            rule = hypo.refinementStr
            if rule not in closedList:
                # Concurrent cover set evaluation
                qr = self.query(rule)

                # Calculate accuracy and score
                rn = self.createRuleNode(rule, qr, bestRule)

                if rn.accuracy >= self.learningParams.minimumAccuracy:
                    self.learningResults.append(rn)

                openlist.put((rn.score, rn, hypo))
```

Listing A.3: Scoring Implementation

```
def calculateScore(self, rn, bestRule, positiveNUM, negativeNUM):
    tp, fp = rn.pos, rn.neg
    fn, tn = positiveNUM - tp, negativeNUM - fp

    accuracy = (tp + tn) / (tp + tn + fp + fn) if (tp + tn + fp + fn) != 0 else 0

    # Calculate parent accuracy for gain computation
    if bestRule.rule != self.ontology.thing.name:
        parent_tp, parent_fp = bestRule.pos, bestRule.neg
        parent_fn, parent_tn = positiveNUM - parent_tp, negativeNUM - parent_fp
        parentAccuracy = (parent_tp + parent_tn) / (parent_tp + parent_tn +
    ↪ parent_fp + parent_fn)
    else:
        parentAccuracy = 0

    accuracyGain = accuracy - parentAccuracy
    ocel_score = accuracy + (0.5 * accuracyGain) - (0.02 * self.ontology.
    ↪ conceptsNum)
    score = (10 * ocel_score) - rn.parms - rn.depth

    return score, accuracy
```