Ontology Graph Embeddings and ILP for Financial Forecasting

Can Erten Dimitar Kazakov

26 September 2021

Ontologies, financial forecasting, SEC financial reports, ILP, unsupervised learning, graph embedding

Abstract

There is a history of hybrid machine learning approaches where the result of an unsupervised learning algorithm is used to provide data annotation from which ILP can learn in the usual supervised manner [7, 8]. Here we consider the task of predicting the property of cointegration between the time series of stock price 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. We start with an original FinTech ontology of relations between companies and their managers, which we have previously extracted from SEC reports, quarterly filings that are mandatory for all US companies. When combined with stock price time series, these relations have been shown to help find pairs of companies suitable to pair trading [3]. Here we use node2vec embeddings to produce clusters of companies and managers, which are then used as background predicates in addition to the relations linking companies and staff present in the ontology, and the values of the target predicate for a given time period. Progol [10] is used to learn from this mixture of predicates combining numerical with structural relations of the entities represented in the data set to reveal rules with predictive power.

1 Introduction

Financial forecasting is a broad discipline that includes the task of predicting future movements of price and volume of trade of company stock and financial products. This information can then be incorporated into profitable trading strategies. One such strategy that does not depend on the economic cycles and current market trends (bullish or bearish) is pair trading. This is a market-neutral strategy that aims to identify pairs of companies which have a predictable, stationary, long term average price when combined in a very simple portfolio in which each company is represented by a specified, optimal proportion of the total value. Maintaining this optimal proportion of each company as

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

Historically, this strategy was not widely known, but has now been implemented in a number of trading tools. The suitability to pair trading of two companies is determined through a numerical test of *cointegration* [6]. With this test now available even to private individuals, the expected profit margins are thought to have shrunk, and new sources of information have been sought to forecast this desirable property in a more reliable manner. 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. One such potential source of information is the regular SEC reports filed by all US companies to report trades of their stock by company staff, as well as any important changes in the company management. We have used these reports, which are usually grouped and archived on a quarterly basis, to populate an ontology representing the relationships between companies and their senior managers. The result is a relatively simple knowledge graph from which we can discover that two companies share certain managers, among other things.

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 look into 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 the day-to-day trades, and 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, we have considered a possible use of our ontology for this purpose, namely, we hypothesised that if two companies share directors or other senior staff that could lead to a flow of information that would make it more likely for the two companies to meet the cointegration test.

We tested this hypothesis in a previous study [3] from 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 around 5% of randomly selected company pairs to be 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, we have taken our work in two directions. We 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. However, this article focuses on another approach, namely, the use of graph embeddings as produced by the `node2vec` algorithm [4] to extract important properties of the

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

graph nodes reflecting the content and topology of each local neighbourhood in an unsupervised way. It is common to cluster the resulting node embeddings and visualise the results in order to discover similarity between nodes. We have adopted the same approach here, using k-means clustering with a hand-selected number of clusters ($k = 20$). The cluster membership was then encoded as background knowledge of a Progol [10] learning task, where the target predicate `coint(Comp1,Comp2)` represented whether the prices of two companies are cointegrated or not for the period in question. In addition, we make use of background predicates showing the affiliation of managers with companies, `keyperson(Company,Person)`, and the overall number of people connected to a company, `ccon(Company,PersonCount)`, resp. companies connected to a person `pcon(Person,CompanyCount)`. These are then used to define `ccon2(Company,MinPersonCount)` and `pcon2(Person,MinCompanyCount)` checking the number of connections of companies or people against the threshold value in their second attribute. We have also defined `connect(Comp1,Comp2)`, a predicate showing whether two companies share at least one person.

In our experiment, we select a list of companies at random, split it into two, and use the first list to generate training pairs of companies (cointegrated vs non-cointegrated), while the second list is used to generate test pairs in which neither company appears in the training data. We then use Progol to learn rules which are evaluated on unseen data. The result shows that the rules we find have high predictive power and the pairs covered by them have a much higher concentration of cointegrated pairs when compared to the unprocessed test data.

2 Background

2.1 Financial Forecasting

There are clear incentives to attempt to forecast the price of financial assets. In that strife, portfolios of assets are used to reduce risk. This can be done in two ways. Diversifying the assets so they are not equally affected by adverse events and spreading the investment across more companies to reduce the relative importance of each one is one obvious approach. The other one is to use a small portfolio, of as few as two companies whose movements are inter-related in a particular way.

Correlated assets are those that tend to move together with statistical significance. The main cause for this is that companies (and markets) do not exist in isolation. They trade with each other and compete for resources and markets. Some assets are correlated because they are in the same sector (e.g. transport or entertainment), or work with the same suppliers: if there is a drop in demand for the same product, or a supplier is having problems, this will likely affect companies across the sector.

2.2 Pair Trading

Pair trading is a market neutral trading strategy enabling traders to profit from any market conditions through the use of cointegration. This is a test assessing the long term relationship between time series [6].

Two companies satisfying the statistical test for cointegration can create a portfolio which will have a stationary long term value even if the price of each company does not result in a stationary time series, provided the portfolio is regularly rebalanced to contain the same proportion of each stock. This amounts to statistical arbitrage and convergence trading strategy where the profit from the stock increasing in value is used to buy more of the stock going down in relative terms.

2.3 Securities and Exchange Commission (SEC) Reports

The U.S. Securities and Exchange Commission (SEC) is an independent agency for the United States federal government. It exists to protect investors, enforce the law against market manipulation and facilitate capital formation.[2] After the Great Depression, Congress passed Securities Act of 1934, which created SEC.

SEC requires the companies to file several different types of report on the company's activities and trades. They publish the formal specification document that every company in the United States has to provide.

The reports are mainly company statements, annual, quarterly and monthly. There are also reports to indicate an official statement or change of structure. Of particular interest to us are reports where the company needs to declare "corporate insiders," employees trading (buying or selling) more than 10% of the company's shares. Forms 3, 4 and 5 provide the framework for reporting insider trading information, where the company also discloses the title of the individual, and any important position, such as director or officer. This can be crucial information as it shows the staff member's role within the company. Their buying or selling can also be an indication of the company's financial performance.

The initial filing is on Form 3, which the insider must file within ten days of becoming an officer, director or beneficial owner. Form 4 reports changes in ownership within two business days. Form 5 is used to report any transactions that should have been reported earlier on a Form 4 as a means of deferred reporting.

SEC does not have an API, all the data is in strongly typed XML with the schema that they publish. They also provide an index file on their server for every report based on company name, period or type of report. We have built a web crawler to download the forms based on the report type and period, and we generate knowledge graphs from the downloaded data for SEC forms 3–5 to represent links between companies and people.

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

2.4 Ontologies

An ontology [9] is a structured database representing objects and their relations, known as *individuals* and *roles* in the field's parlance. Ontologies are formal representations of knowledge with well-defined semantics. An ontology allows for the grouping of individuals into classes (aka concepts). Both concepts and roles can form hierarchies, depending on the exact formalism used. The use of standard dictionaries of concepts and relevant roles not only assists the representation of domain knowledge, but also the easy integration of multiple an ontology is equivalent to a set of Description Logic (DL) axioms, which provides the semantics for query languages (such as SPARQL) and reasoners. The storage unit is a triplet with fields known as subject, predicate and object. These combine to a create a directed graph with entities represented by Unique Resource Identifiers (URI). Machine learning algorithms specialised in the use of DL to represent data and models also exist [9, 1].

2.5 Node2vec Embeddings

Graph Representation Learning is a relatively new field which allows one to apply existing machine learning algorithms, such as CNN and RNN neural networks on graph data. In order to achieve that the graph need to be translated to the vector space, and represented as a tensor. 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 representations of nodes on a graph [5]. 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 our earlier work [8].

3 Methodology

3.1 Data Collection

The data used is from the SEC reports, which are available to download in XML (XBRL) format. We have built an index parser to identify the available reports for a period and given type, and then built a crawler to download the actual report files. When all reports are downloaded, we apply an in-house custom parser to the XBRL (XML with metadata and schema) file in order to generate the ontology.

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 1: Knowledge graph sample

3.2 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 [2]. A sample of the data is shown in Figure 1.

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 on Figure 2 and Figure 3.

3.3 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;
- Run a query to select companies (background information);
- Run a query to select linked people with the company information (background information);

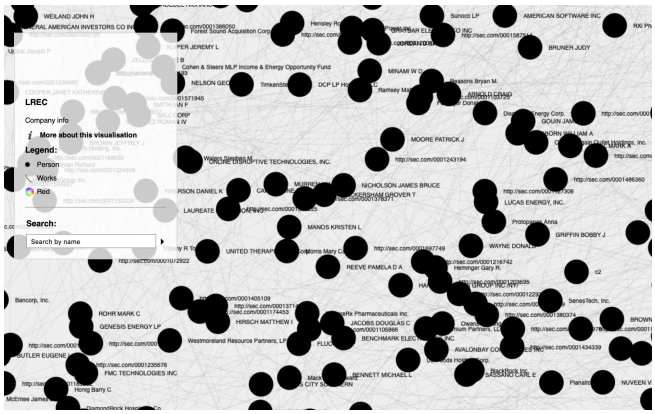

Figure 2: Ontology visualisation (part)

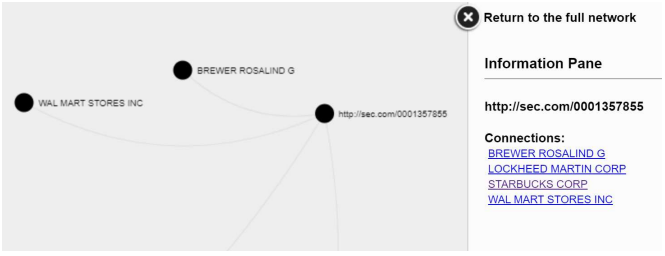

Figure 3: Ontology visualisation of a person represented by an URI

- Split the data into a training set and a test set;
- Run the cointegration test for all pairs of companies in the report for the period in question to generate positive and negative examples;
- Run the node2vec algorithm on the data to generate node embeddings;
- Run k-means on the embeddings to obtain clusters (to be used as background knowledge);
- Generate a Progol source file from the background knowledge and target predicate examples;
- Run Progol on the data.

4 Experiment Design and Results

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

The result of this is a graph with nodes representing either a company or a person, and edges showing when a person is linked to a company. The algorithm `node2vec` [4] is then used to generate an embedding for each of the graph nodes. The embedding is a real valued vector of size set to 20. For the remaining parameters the choice was: `walk_length=16`, `num_walks=100`, `workers=2`. All node embeddings were then clustered using k-means with $k = 20$. (There is no link between the dimensionality of the embedding and the choice for the number of clusters.) 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 can be modified through the choice of the parameter `walk_length`.

With these results in hand, we carried out an experiment with the purpose of learning rules that would predicate the cointegration of two companies 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.

[3] The second author is supervising an ongoing MScRes project on extracting relations from free text at the moment of writing.

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

Table 1: Dataset description

#of Training Companies	700
#of Test Companies	300
Positive Training Pairs	8,833
Negative Training Pairs	235,817
Positive Test Pairs	1,405
Negative Test Pairs	43,445

The experiment uses the knowledge graph generated from SEC reports for Q3/2017. There is data on a total of 1948 companies for this period. Of these, 1000 companies were randomly selected, and given a 70% : 30% split. All 244,650 possible pairs formed by companies in the first set were earmarked as training data, while all 44,850 pairs formed by companies in the remaining set were used as test data. As for each company we also had the time series with the movement of stock prices over the entire three-month period, it was possible to apply a cointegration test to each pair of companies and label it as a positive or negative example. After some experimentation, we opted for positive-only learning for reasons of both speed and accuracy. This meant that in reality 8,833 training examples were used, all of them positive, which represented 3.61% of the $244,650$ training pairs (see Fig. 1). The background predicates and mode declarations used for learning are as follows:

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

coint/2: Defines whether a pair of companies are cointegrated or not.

ccluster/2: Shows the cluster a company belongs to.

ccon2/2: The predicate succeeds if its first argument is a company 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 company is connected to a certain minimum number of people. Expressed as Prolog code:

```
ccon2(Company,Threshold):-
  ccon(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.
```

connect/2: Defines whether a company is connected to another company via at least one key person:

```
connect(CompA,CompB):-
  keyperson(CompA,Person),
  keyperson(CompB,Person),
  CompA \= CompB.
```

We should note here that even the background predicates that define propositional properties, e.g. the cluster number, are based on relational data about connections between graph nodes, while the remaining ones express a numerical relation ('$\geq$') summarising that relational data.

The result of learning is a list of clauses of the target predicate as shown in Tables 2–3. The overall accuracy of the theory learned is 50.42% as shown in Table 4, which is a very high result for such notoriously fickle data. The two classes for this binary classifications are not equally represented in the test set, as we wanted to stay faithful to the original purpose of this work, which is to eliminate negative examples from the data while preserving as many positive examples as possible. In this we succeed: 96.9% of the positive test examples are accepted by the theory, while around half of the negative examples are rejected. The resulting data sample, that is, the set of pairs accepted by the learned definition of the target predicate, contains 5.78% of positive pairs or by 85% more than the entire test set, where this percentage is 3.13%. This difference are statistically significant, and of substantial practical value, as it reduces the list of candidate cointegration pairs by half at almost no loss of 'good' pairs.

With the evaluation of the entire theory out of the way, 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.

We have grouped the rules learned by Progol by their type for ease of reading. First, we have nine rules only specifying the cluster membership of either company in the pair (see top of Table 2). Intuitively, the rules appear underconstrained, 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.

Table 2: 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(B,17).	101	2138	0.05	**.0003**
coint(A,B) :- ccluster(B,14).	79	1473	0.05	**.0000**
coint(A,B) :- ccluster(A,13), ccluster(B,15).	10	54	0.16	**.0000**
coint(A,B) :- ccluster(A,6), ccluster(B,13).	13	100	0.12	**.0000**
coint(A,B) :- ccluster(A,7), ccluster(B,13).	15	120	0.11	**.0000**
coint(A,B) :- ccluster(A,4), ccluster(B,12).	15	114	0.12	**.0000**
coint(A,B) :- ccluster(A,5), ccluster(B,12).	5	58	0.08	**.0289**
coint(A,B) :- ccluster(A,9), ccluster(B,12).	8	53	0.13	**.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 3: 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 4: 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		

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 nor can we see any substantial number of connections between the two clusters when we look at the topology of the knowledge graph for the nodes in clusters 13 and 15 (Figure 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 we 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 [3].

Finally, we should mention the rules setting 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 Conclusion

The results show that the rules obtained are potentially interpretable, and that they have genuine predictive power. We conjecture that the results will further improve as we increase the data set, something we have tried to limit here to keep the complexity of the learning task under control. We shall also consider transferring the learning task on to a concurrent learner that scales up better [1]. It is important to note the hybrid nature of the approach, in which an ontology is populated with data extracted from company reports through text processing, then the embeddings for the graph nodes are generated, and clustered, all in an unsupervised manner. The cluster labels are used as background knowledge alongside other relations from the graph, and classic supervised ILP learning is applied to obtain the final results. So, we are dealing with a hybrid learning approach for two separate reasons, because it combines unsupervised with supervised learning, and as logic facts and relations are combined with the output of numerical computations based on time series. This is clearly a powerful and promising novel approach, and our future work will focus on scaling up the data set, and analysing the results with domain experts with the aim of revealing additional salient properties and relations to the range of background

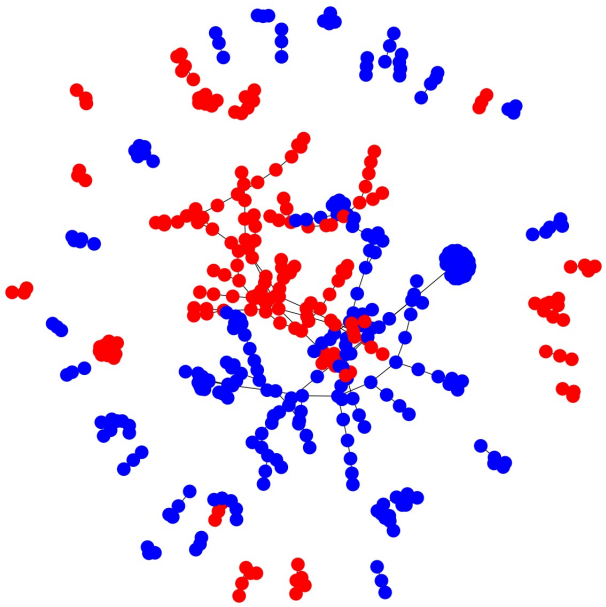

Figure 4: Knowledge graph subset containing clusters 13 and 15 (red vs blue)

predicates in order to allow for ever more accurate and revealing hypotheses to be learned.

References

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

[2] DuCharme, B.: Learning SPARQL. O'Reilly Media, second edn. (2013)

[3] Erten, C., Chotai, N., Kazakov, D.: Pair trading with an ontology of SEC financial reports. In: The 2020 IEEE Symposium Series on Computational Intelligence (2020)

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

[5] Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: KDD (2016)

[6] Johansen, S.: Statistical analysis of cointegration vectors. Journal of Economic Dynamics and Control **12**(2), 231–254 (1988).

https://doi.org/https://doi.org/10.1016/0165-1889(88)90041-3, https://www.sciencedirect.com/science/article/pii/0165188988900413

[7] Kazakov, D.: Achievements and prospects of learning word morphology with inductive logic programming. In: Cussens, J., Dzeroski, S. (eds.) Learning Language in Logic, pp. 89–109. LNCS 1925, Springer (2000)

[8] Kazakov, D., Manandhar, S.: Unsupervised learning of word segmentation rules with genetic algorithms and inductive logic programming. Mach. Learn. **43**(1/2), 121–162 (2001), http://dblp.uni-trier.de/db/journals/ml/ml43.html#KazakovM01

[9] Lehmann, J., Völker, J.: An introduction to ontology learning. Perspectives on Ontology Learning **18**, ix–xvi (01 2014)

[10] Muggleton, S.: Inverse Entailment and Progol. New Generation Computing, Special issue on Inductive Logic Programming **13**(3-4), 245–286 (1995), citeseer.nj.nec.com/muggleton95inverse.html