


Evaluating Knowledge Graph Sources for Non-Personalized Financial Asset Recommendation: 10K Reports vs. Wikidata

Lubingzhi Guo ^[0009–0001–9283–5747], Javier Sanz-Cruzado^[0000–0002–7829–5174],
and Richard McCreddie^[0000–0002–2751–2087]

University of Glasgow, Glasgow G12 8QQ, UK
l.guo.1@research.gla.ac.uk,
{javier.sanz-cruzadopuig, richard.mccreadie}@glasgow.ac.uk

Abstract. Financial asset recommender (FAR) systems suggest investment assets to customers based on past market information. Many of these models choose those securities which they estimate to be more profitable for customers. Financial knowledge graphs (KGs) – data structures containing information about assets and their relations to other involved entities (companies, people) – have been one of the data sources exploited to drive asset selection. Although the construction of knowledge graphs from different sources (news, reports) has previously been investigated, there has been limited analysis of the effect these construction strategies have for FAR. In this work, we compare two different knowledge graphs representing U.S. stocks under a unified FAR framework: a knowledge graph crawled from a general knowledge base, Wikidata, and a knowledge graph built by extracting entities and relations from 10K financial reports using the GoLLIE open information extraction model. We show that integrating these KGs in FAR can lead up to 10.7% improvements in monthly ROI. However, the nature of these graphs makes algorithms prone to bias the recommendations towards different asset types. Therefore, we further propose and evaluate an adaptive graph selection strategy, which dynamically chooses the suitable graph prediction model—trained on either the 10K Graph or the Wikidata Graph—for each asset. The findings indicate that stock-level and sector-level selection strategies respond differently to the length of the recency window, reflecting, respectively, a preference for short-term responsiveness and long-term stability.

Keywords: Knowledge Graph Construction · Stock Recommendation · Information Extraction

1 Introduction

Financial asset recommender systems are tools to assist investors in making informed investment decisions [24,40,44,45]. These technologies aim to produce a ranking of financial securities (e.g. stocks) on which a customer might invest. As one of the main targets of these methods is to help customers increase their wealth, the majority of the methods proposed in this field rely on historical pricing information of assets to predict stock price movement [29,55]. This core signal is then often augmented through the

integration of evidence from external information sources such as textual data from news and social media [18,9,10].

A *financial knowledge graph* (KG) is a data structure that can be used to store such external information [10,11,50,57]. In such a graph, nodes represent entities (companies, people), while edges represent relations between them (e.g. between-company or board member relationships). Multiple methods have been proposed for the creation of financial KGs, including crawling general knowledge bases like Wikidata [14] or extracting facts and events from financial reports [20,33] or news [10,13].

Different KG creation techniques have advantages and disadvantages. On the one hand, graphs extracted from financial documents (such as news or financial reports) include timely information about companies and related entities, which is specific to the financial domain. However, they are generated by applying information extraction techniques over (typically) unstructured text documents [4], and as such are prone to hallucinating incorrect facts [32]. On the other hand, graphs produced from general knowledge bases like Wikidata or DBPedia have their factoids assessed by human annotators to ensure their correctness [48], but often incorporate connections that are irrelevant for the financial domain and are updated infrequently. While both approaches have been tested in isolation previously, no prior works have quantitatively compared these two different strategies. This is an important gap, as intuitively the graphs produced by these two strategies are likely to benefit different types of assets being recommended, introducing a structured bias. For instance, general knowledge bases result in larger graphs with imbalanced coverage toward well-known/long standing companies. Meanwhile, news-based graphs are smaller and more focused on companies that are newsworthy.

Hence, in this work, we tackle the above gap by analysing the impact that these two knowledge graph construction strategies have when predicting the future profitability of U.S. stocks. First, we produce a financial knowledge graph containing information about companies from Wikidata. Second, we build a knowledge graph by applying automated information extraction techniques over 10K reports using large language models (LLMs) [39]. As required by the U.S. Securities and Exchange Commission, these annual reports disclose detailed financial performance and announcements about publicly traded companies to stock investors and unsurprisingly trigger immediate market responses [15]. We then compare the utility of these knowledge graphs under a unified profitability prediction framework integrating knowledge graph embeddings [49] as features for the task. Afterwards, we analyze which types of assets are better promoted by each knowledge graph on the stock recommendation task and, finally, we propose and test ensemble strategies that adaptively select the best knowledge graph for scoring different stocks.

Specifically, the primary contributions of this work are fourfold:

- We construct a financial KG from 10K reports using fine-tuned LLMs for open information extraction.
- We crawl a financial KG from Wikidata as a general knowledge base.
- We compare the effect these knowledge graphs have on profitability prediction over the U.S. stock market, demonstrating that integrating these KGs can lead up to 10.7% higher in monthly ROI. We also demonstrate that different KG construction strategies bias their results towards separate sets of assets.

- We compare two different adaptive graph selection methods that combines two distinct knowledge graphs for stock recommendation, showing that well-designed selection criteria can consistently enhance monthly ROI.

2 Related Work

To integrate external evidence into an asset recommendation system using a knowledge graph as an intermediate representation, we need two technologies: 1) a graph generator, that takes information about financial topics and extracts associated entities as well as their relations that form the graph; and 2) an entity embedder, which given an asset produces a vector embedding that encodes information related to the asset from the graph. We introduce past works regarding each below:

2.1 Knowledge Graph Generation

The first step in knowledge graph generation is to select the type of content that you want to extract information from. If producing a new knowledge graph from scratch, the most popular data source to use is financial news articles, as intuitively significant events that affect an asset will have associated news content, but conversely much irrelevant information will also be captured [13]. Instead, a clean data source such as financial filings can be used, which is more targeted and in-depth, but are published infrequently [20,33]. Alternatively, rather than building a completely new graph, some works have bootstrapped from an existing knowledge-base, such as DBPedia, where a large general knowledge graph needs to be filtered down to a useful subset for the financial domain [48]. To contain the scope for this initial work, we compare approaches to model what a company (that can be invested in) *is and does*. As such, we use U.S. 10K financial reports as a company provided overview of their operations and compare it to company information from WikiData.

Once we have selected our data sources, we next need to extract the entities and relationships that will form each graph. Depending on the type of data being used, the approach here will differ. If using an existing knowledge graph, then a set of filtering rules needs to be defined, as well as potentially entity disambiguation performed. However, for text-based data sources, Information Extraction (IE) techniques need to be applied to the raw text. This is usually comprised of two components: 1) entity identification (and linking), which identifies financial entities (companies, people, places) in the text; and 2) relationship extraction, which generates likely relationship tags between pairs of entities [33]¹. For instance:

$$\text{Entity1 'Nik Jhangiani', Relationship 'CFO', Entity2 'Diageo'} \quad (1)$$

Of note is that for relationship extraction, approaches can be either closed domain (a set of target relationship types are defined beforehand) or open domain (any relationship tag

¹ Relationships may also have extracted properties/qualifiers, such as an indicated date for when the relationship was formed.

can be generated) [19]. While most works focus on closed-domain extraction, the emergence of effective large language models has opened the door to less error-prone open-domain extraction than was previously possible, with models such as GoLLIE [39]. In this work, we use GoLLIE over 10K filings to perform open-domain extraction, where the model is guided to look for either business, transaction or personnel-related relationships.

2.2 Entity Embedding

Once we have a financial knowledge graph, given a financial asset representing a company that we want to recommend, we need to produce an embedding representing what the knowledge graph has about that company. To do this, knowledge graph embedding (KGE) models are used, which produce a low-dimensional vector representation given a starting graph node or edge to represent [49]. There are three families of KGE models:

Translation-based Translation-based techniques represent entities as points and relationships as transformations (translations or rotations) in vector spaces. The idea is that, if we apply the relation to the head entity vector, the resulting vector will be close to the tail entity. An early representation of this group of algorithms is TransE [6], which considers relations as translations between head and tail entities in a common embedding space. As this simple model struggles with one-to-many, many-to-one, symmetric or transitive relations [37], subsequent models like TransH [37] and TransR [22] introduce additional spatial dimensions to handle a variety of relations. TransH handles complex relationships via hyperplane projections, whereas TransR separates entity and relation spaces. Beyond translations, RotatE [42] represents relations as rotations in a complex vector space.

Factorization-based Factorization methods on the other hand implement a scoring function based on semantic similarity to estimate the plausibility of a given triplet, typically by mining the latent semantics between entities and relations [37]. These models include RESCAL [31] which leverages three-way tensor factorization techniques and represents relations as full-rank matrices, outperformed but is computationally demanding. Several methods have made progress in refining the tensor factorization procedure in comparison to RESCAL, resulting in increased computational efficiency while preserving the ability to handle asymmetric relations. DistMult [54] simplifies the procedure by representing each relation as a diagonal matrix, thus shrinking the parameter space. TuckER [5] uses Tucker decomposition and combining relations in low-rank matrices. HolE [30] reduces the number of parameters by employing circular correlation operation.

Neural Network-based Neural networks are considered a promising solution in many domains since their large number of parameters enable them to learn complicated patterns and also encode weights and biases observed [37]. Therefore, several KGE methods have taken these algorithms as a basis. ConvE [12] introduces the utilization of 2-dimensional convolutions over embeddings for link prediction. Other models investigate the generalisation power of graph neural networks (GNNs) for the task. An early

example of these methods is the RGCN model [41], which adapts graph convolutional networks for their use on link prediction and entity classification on knowledge graphs. Subsequently, building upon graph attention networks [47], KGAT [51] effectively applies the attention mechanism for higher-order relation modelling.

In our later experiments, we compare asset embeddings produced by a range of algorithms across these three types for both 10K filings and Wikidata-based knowledge graphs.

2.3 Financial Asset Recommendation

Finally, having produced a knowledge graph embedding for a company/asset, we can then use that embedding to augment a downstream task. In this work, we target Financial Asset Recommendation as that task, where given a day, we want to rank assets on that day such that the future return-on-investment of the top ranked assets is maximised [2,3,14]. For this, we rely on non-personalized regression models like the ones used by [35,40]

Similarly to our work, several models have integrated pricing and knowledge graph information for stock predictions [14,56]. These models either exploit similarities between assets [23,50,56] or integrate KGs as features [10,11,57]. We explore the second way. However, previous feature-based approaches need specific KG structures or data sources to build those KGs. Differently, we propose a simple framework which integrates knowledge graph embeddings as features. This allows the use of any financial KG as input to our models – something that we can use to compare the effect that different knowledge graph structures have on the recommendations.

3 Knowledge Graph Construction

In this work, we construct two financial knowledge graphs from two different sources to compare the performance for financial asset recommendation / stock recommendation.

3.1 Knowledge Graph Definition

We first provide a formal definition of a knowledge graph. Following the property graph model defined by [16], a knowledge graph \mathcal{G} is defined by 5 components $\mathcal{G} = \langle \mathcal{E}, \mathcal{L}, \mathcal{V}, \mathcal{R}, \mathcal{P} \rangle$. \mathcal{E} is the set of entities in the graph. Entities represent objects, companies, people or abstract concepts. For instance, the board gaming company Hasbro and the Dungeons & Dragons (D&D) tabletop game represent entities in a knowledge graph. \mathcal{L} represents the relation labels (types) in the knowledge graph. An example of relation label is ‘CEO’. \mathcal{V} is the set of literal values which can be used to represent properties of entities and relations, such as a date or a number. The set $\mathcal{R} \subset \mathcal{E} \times \mathcal{L} \times \mathcal{E}$ contains triplets $r = (e_h, l, e_t)$ representing directed links between entities in the knowledge graph, where $e_h \in \mathcal{E}$ is the origin or head entity, $e_t \in \mathcal{E}$ is the destination or tail entity, and $l \in \mathcal{L}$ represents the type of the relation. An example of a relation would be $r = (\text{Hasbro}, \text{CEO}, \text{Chris Cocks})$ – indicating that Chris Cocks is the CEO of Hasbro.

| Element | Valid types |
|----------|--|
| Entity | Organization, Person, Location, Market, Product, Material, Activity, Award, Legal form, Form of government, Gender, Health problem |
| Relation | Ownership, Employment, Part of, Creation, Award, Location, Material, Skills, Condition, Sequence |
| Property | Time, Position, Location, Item or service, Amount |

Table 1: Entity, relation and property types in the Wikidata graph.

2021 as our seed entities, since we aim to predict the future pricing of these stocks. We employ a semi-automated three-step process to match those companies with entities in the Wikidata knowledge base:

First, we use the SPARQL Wikidata Query Service² to filter and retrieve entities related to the stock exchanges of interest (NASDAQ, NYSE, AMEX) and gather their identifiers, names, and aliases in multiple languages. If a direct match between the company tickers and Wikidata entries is found, we link them automatically. Second, for assets without direct matches, we use DBpedia Spotlight [25] for entity recognition and linking to entries in DBpedia (another public knowledge base), which are then cross-referenced to Wikidata identifiers. We verify the results manually to ensure accuracy. Finally, for any unmatched entities, we conduct a manual search in Wikidata. If a company does not match any entity, it is excluded from our dataset, assuming no association with Wikidata exists. We started with a total of 5,823 assets from NASDAQ, NYSE and AMEX. Via entity matching we mapped 3,370 of these (57.9%) to Wikidata pages.

Entity and Relation Extraction Starting with mapped Wikidata entries, we extract metadata, relations, and properties for each entry, including any available temporal information for the relations. Our crawling method follows a breadth-first search algorithm, beginning with seed entities and expanding outward in a first-found, first-served manner. To avoid crawling information outside the financial domain, we cap the search depth from the seed entities. Furthermore, we have specified a list of valid financial relations and entity types to guide our crawler. The broad types of those entities and relations are summarized in Table 1.

3.3 10K reports graph

Second, we create a graph from financial texts using automated relation extraction. In this work, we construct our automated knowledge graph using some of the most comprehensive and official financial reports: the 10K annual filings. We next provide details of our information extraction procedure.

Pre-processing Considering that financial reports are lengthy and exceed the context window of the LLM we use, we initially perform sentence segmentation on each report,

² <https://query.wikidata.org/>

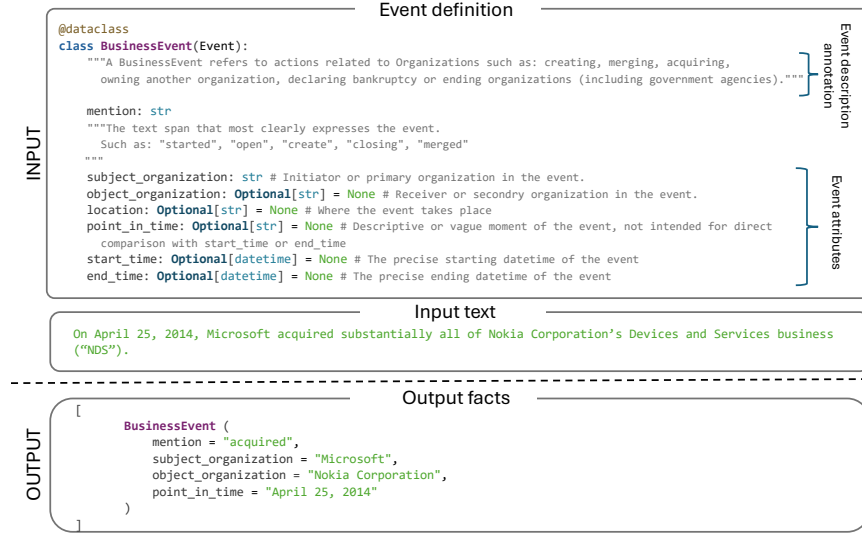


Fig. 2: An example of extracted hyper-relational facts

converting it into a list of sentences using Stanford NLP [34]. This enables us to generate hyper-relational facts. Furthermore, as in most reports, terms such as ‘we’, ‘the company’ and ‘the corporate’ refer to the reported company, as such we resolve/replace these with their corresponding company names.

Entity and Relation Extraction Our information extraction pipeline is based on GoLLIE [39]. GoLLIE is a recent LLM-based model for zero-shot information extraction (IE). This model has been successfully applied to multiple IE tasks across multiple domains, so we use it in our work to extract financial entities and relations from the 10K filings. This model is based on Code-LLaMA [38] and represents both input and output using Python classes. It receives two inputs: first, a text from which to extract information, a second, a list of Python class definitions describing the information to extract. For each label, there is a Python class detailing its structure (where class attributes represent specific information pieces to extract). Extraction guidelines are embedded as comments in the Python class to assist the model. The output of the model is a list of instances of the pre-defined Python classes, each containing a pair of entities, a relation and any relevant properties.

In our information extraction procedure, we perform event argument extraction to produce hyper-relational facts using event templates provided for GoLLIE³. We define three types of events to extract:

³ <https://github.com/hitz-zentroa/GoLLIE/blob/main/notebooks/EventExtraction.ipynb>

| Entity | Valid Type |
|---------------------------|---|
| Head Entity & Tail Entity | Organization, Person |
| Property Label | Time, Position, Location, Item or service, Amount |

Table 2: Entity types in the 10K graph.

- **Business event:** actions related to organisations. For instance, creating, acquiring or ending other organisations as well as declaring bankruptcy.
- **Transaction event:** this event type refers to exchanges between organisations, either of artefacts, people or money.
- **Personnel event:** interactions between people and organisations. For instance, foundation of a company, or election of a person for a position.

All of these events are represented by a hyper-relational fact (e_h, l_r, e_t, l_p, v) , where $e_h, e_t \in \mathcal{E}$ are the head and tail entities, l_r is the label indicating the type of the relation between them and, if any, l_p represents the type of a link property with v as the property value. For each event type, we define the types of the entities, and allow the model to extract the relation types. We broadly classify the link properties into five groups: ‘time’, ‘position’, ‘location’, ‘item or service’ and ‘amount’.

Figure 2 shows an example of the input and output of the model. In the predefined ‘BusinessEvent’ data class, we first provide a text describing the types of events we want to extract as a comment. In order to capture the business facts related to organizations, we define the head entity as ‘subject organization’, and tail entity as ‘object organization’, adding location and time as properties. As it is common that the extracted facts may not always include all the components we thus define Optional typings to handle diverse matching patterns. This example demonstrates an extracted ‘BusinessEvent’ instance from the input text, with ‘Microsoft’ as the head entity and ‘Nokia Corporation’ as the tail entity. The relationship between them is described by the term ‘acquired’, with ‘April 25, 2014’ serving as the only time-related property.

Relation Clustering The identified relational phrases are continuous text spans directly extracted from the sentence, which tend to be noisy. Therefore, additional steps are needed to match different relation types. Inspired by Hu et al. [17], we use clustering to group similar relational facts in an unsupervised manner.

We first perform lemmatization on the extracted relations to reduce variations in their word representation. To perform the clustering, we represent the relation mentions as vectors using the Sentence-BERT pretrained language model [36]. This enables us to establish distances between relation phrases, which we can use to perform the clustering. Then, we use agglomerative clustering [26] to group the vectors and combine similar relations – thus shrinking the number of different relations in our knowledge graph. We use this simple and hierarchical algorithm as it allows us to establish a distance threshold for grouping instead of a number of clusters – something that aligns well with our open information extraction task. In our configuration, we consider that two vectors with distance (within-cluster variance) smaller than 1 belong to the same

cluster. Finally, to further refine our dataset with higher qualified relations, we exclude relations that occurred with a frequency below the 75th percentile threshold for the entire collection (6 times in our data).

Entity Linking Similar to extracted relations, the entities extracted from texts are text spans that need to be unified and linked to real-world entities. Given the difficulty in accurately mapping individual names, we only focus on organisations. We apply two methods. First, we use a zero-shot entity linking method based on BERT known as BLINK [53]: this approach matches spans of text to entities in Wikipedia. Second, we use the company name normalisation functionality of the John Snow Labs NLP library⁴. This feature maps extracted company names to the name registered with the SEC in the Edgar Database, which is useful when handling 10K filings and aims to enhance the accuracy of linking organisational entities.

Both methods provide, as outputs, an entity name and a confidence score. To ensure reliability of the collected entities, for each method, we only keep those text-entity pairs with confidence scores above the median score obtained for all the analysed text spans. We keep the matching if the confidence score for one of the two methods is above the median for the entity linker. In case we have positive matchings for a company in both methods, we keep the one provided by BLINK.

4 Profitability Prediction with Knowledge Graph Embeddings

After constructing our financial knowledge graphs, we aim to use them to improve price prediction accuracy, where future predicted prices are used to recommend financial assets. We therefore define a simple aggregation framework that enables us to combine knowledge graph information with temporal pricing information, depicted in Figure 3.

For a point in time t , given a set of financial assets, we first feed the gathered knowledge graph into a knowledge graph embedding (KGE) model. These models produce embeddings summarizing the information we have about the entities representing the assets in the knowledge graph. Separately, the historical price data is processed to create the technical indicator sequence for each stock. Both technical indicators and knowledge graph embeddings are then concatenated and given as input to a profitability prediction method that estimates the future profitability of the assets and then ranks them in descending order according to that estimation.

This is a classical feature aggregation approach that has previously been shown to be effective in a wide range of scenarios and so we won't discuss it further. Rather, in the remainder of this section we describe how the two types of features (technical indicators and entity embeddings) are generated in more detail and provide a general definition of the profitability prediction regression algorithms.

4.1 Asset vector generation

As a first step to perform profitability prediction, we need a representation of the different financial assets that we want to predict for. There are two types of information

⁴ <https://github.com/JohnSnowLabs/johnsnowlabs>

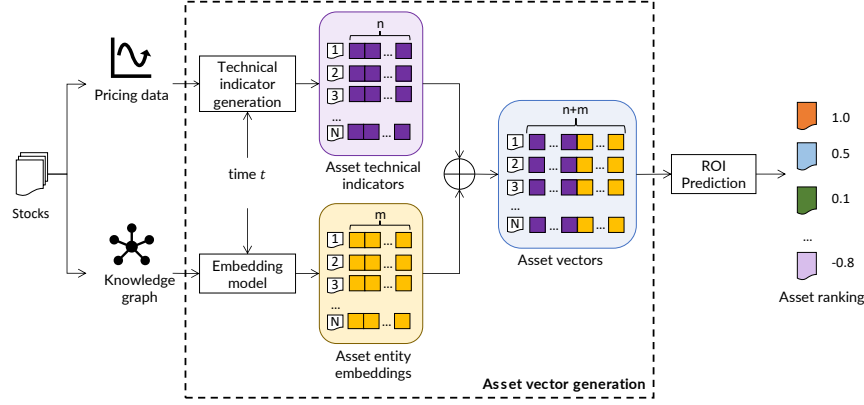


Fig. 3: Profitability Prediction Architecture

that we consider in this work to represent assets: 1) asset past technical data (i.e. information derived from the price of an asset over time); and 2) information regarding the company associated to the asset (based on our knowledge graph). Notably, these asset features will vary over time, as market pricing is updated frequently and information about the company gets updated periodically.

Given a particular time, t , we represent a stock $s \in \mathcal{S}$ as a numerical vector of dimension D , which we denote as $s(t) \in \mathbb{R}^D$. In our framework, $s(t)$ is defined as

$$s(t) = s_{TI}(t) \oplus s_{KG}(t) \quad (2)$$

where $s_{TI}(t) \in \mathbb{R}^n$ is a vector containing technical indicators for the asset s at time t , with n representing the number of indicators and $s_{KG}(t) \in \mathbb{R}^d$ is a d -dimensional embedding of the entity representing the asset in the knowledge graph at time t . We provide more information about these vectors next.

Technical indicators Technical indicators (also known as key performance indicators, KPIs) are heuristics that encode some aspect of the past pricing information of a financial asset. These heuristics have been shown to provide useful signals when predicting the future profitability of financial assets [28], and are widely used when training price prediction models [27,28,40]. Technical indicators do not commonly analyse the whole time series before a given time t . Instead, they usually summarize a fixed period of time, given by a window Δt . Examples of technical indicators include the average price of a stock during a period of time, the return on investment of the stock in the last month, or the volatility of the asset price (standard deviation of the stock price between two points in time).

We can mathematically define a financial technical indicator as a function $f : \mathcal{S} \times \mathbb{R}^+ \times \mathbb{R}^+ \rightarrow \mathbb{R}$. This function receives as an argument (a) the price time series of an asset $s \in \mathcal{S}$, (b) the time t and (c) the time window Δt . The technical indicator studies then the time series $s[t - \Delta t, t]$ (i.e. the pricing information of the asset between $t - \Delta t$

and t) and provides a numerical value. In our framework, we define a technical indicator vector as:

$$s_{TI}(t) = \{f_i(s, t, (\Delta t)_i)\}_{i=1}^n \quad (3)$$

where $\{f_i\}_{i=1}^n$ represents the technical indicators to use and $(\Delta t)_i$ represents the time window chosen for the technical indicator f_i . We allow the use of different time windows as the value and meaning of an indicator might differ according to Δt . For instance, the technical indicator [return on investment (ROI)] for the prior 1 week may tell us something different than [return on investment (ROI)] for the prior year. The set of technical indicators used are summarized later in Table 4.

Asset entity embeddings Knowledge graph embedding (KGE) models aim to encode the information in the knowledge graph into a low dimensional space, with each entity in the graph represented by an embedding vector, while preserving the important properties of the graph [8]. Entity embeddings are computed considering not only the information about each entity, but also that entity’s relationship with other entities in the graph (thereby encoding information about related assets and relations within the embedding). In this work, we experiment with a range of methods for generating knowledge graph embeddings for our financial assets that we use downstream as features for profitability prediction.

An important consideration here is that entities and relations might appear and disappear over time, and, if we are predicting the price of stocks at a given point in time t , we should not have any information in the knowledge graph that appeared after time t (i.e. we need to avoid including information from the future). For instance, if we predict the profitability of NASDAQ stocks in 2003, we should not have any information about Meta or Twitter (which were founded, respectively, in 2004 and 2006). Using the properties of the entities and relations, we can remove future information from the graph. We define the knowledge graph containing only information before time t as $\mathcal{G}(t)$.

Formally, we want to produce a representation of an asset s as a vector $s_{KG}(t)$ containing the embedding of the entity related to that asset in the knowledge graph. We can formulate a KGE model as a function $g : \mathcal{G} \rightarrow \mathcal{E} \times \mathbb{R}^d$ where \mathcal{G} is a knowledge graph, \mathcal{E} is the set of entities and d is the dimension of the embedding space. If we denote the entity associated to asset s in the knowledge graph $\mathcal{G}(t)$ as e_s , we can define $s_{KG}(t) \in \mathbb{R}^d$ as the vector embedding of e_s in the knowledge graph $\mathcal{G}(t)$:

$$s_{KG}(t) = g(\mathcal{G}(t)) [e_s] \quad (4)$$

In order to generate those embeddings, KGE models establish an scoring function $g_s : \mathcal{E} \times \mathcal{L} \times \mathcal{E} \rightarrow \mathbb{R}$ for every possible triplet (e_h, l, e_t) , denoting the plausibility of every relation. The learning process of these models preserves the structure of the graph by maximizing that scoring function for the existing relationships \mathcal{R} .

There are many knowledge graph embeddings models that have previously been proposed, as discussed previously in Section 2.2. However, to-date these models have not been compared on the profitability prediction task. Hence, we select a representative set of both popular and state-of-the-art models from the literature and experiment with them our later experiments. While exhaustively describing each embedding model

is out-of-the-scope of this work, we provide a brief formulation of each KGE model we use below to highlight the differences between them, where $|\mathcal{E}|$ denotes the number of entities, $|\mathcal{R}|$ denotes the number of relations, d denotes the dimension of the embeddings and, to simplify the notation, e_h and e_t represent here, respectively, the embeddings of the head and tail entities:

Translation-based embeddings

- **TransE [6]**: this method represents relations as translations between entities in the same embedding space. The tail entity vector e_t should be near to the head entity vector e_h plus the relation vector l , i.e., $e_h + l \approx e_t$. Hence, we use as scoring function g_s to optimize the negative L1-norm distance between $e_h + l$ and e_t :

$$g_s(e_h, l, e_t) = -\|e_h + l - e_t\|_1$$

- **TransH [52]**: TransH extends TransE by representing relations as vectors in a $d-1$ -dimensional hyperplane with normal vector w_l . This method projects the entity vectors e_h and e_t , respectively, onto the hyperplane as:

$$\begin{aligned} e_h^\perp &= e_h - w_l^\perp e_h w_l \\ e_t^\perp &= e_t - w_l^\perp e_t w_l \end{aligned}$$

Hence, the relation vector l is thus regarded as the connection between e_h^\perp and e_t^\perp in the hyperplane, and the scoring function g_s to optimize is then defined as:

$$g_s(e_h, l, e_t) = -\|e_h^\perp + l - e_t^\perp\|_2^2$$

- **TransR [22]**: TransR breaks the restrictive assumption of TransE and TransH that entities and relations live in the same semantic space. Instead, it represents them in distinct vector spaces by projecting entities ($e_h, e_t \in \mathbb{R}^d$) to relation space using a projection matrix $M_l \in \mathbb{R}^{k \times d}$.

$$\begin{aligned} e_h^l &= e_h M_l \\ e_t^l &= e_t M_l \end{aligned}$$

With $l \in \mathbb{R}^k$, the scoring function g_s to optimize is defined as:

$$g_s(e_h, l, e_t) = -\|e_h^l + l - e_t^l\|_1^2$$

- **RotatE [43]**: RotatE represent each relation as an element-wise rotation from the head entity to the tail entity. This method expects that

$$e_t \sim e_h \circ l, \quad \text{where } \|l_i\| = 1 \forall i \in \{1, \dots, d\}$$

where \circ represents the Hadamard product and relations are represented as vectors $l \in \mathbb{C}^d$ in the complex d -dimensional sphere of radius equal to 1. The scoring function g_s for each relation is then defined as:

$$g_s(e_h, l, e_t) = -\|e_h \circ l - e_t\|$$

Factorization-based embeddings

- **RESCAL [31]**: RESCAL employs tensor factorization to model latent entity representations and their interactions. Entities are represented as vectors, while every relation l is represented as an asymmetric matrix $L \in \mathbb{R}^{d \times d}$. The scoring function is defined as:

$$g_s(e_h, l, e_t) = e_h^T \cdot L \cdot e_t$$

- **HolE [30]**: HolE combines the advantages of RESCAL and TransE and improves efficiency by using the circular correlation operator $*$: $\mathbb{R}_d \times \mathbb{R}_d \rightarrow \mathbb{R}_d$, which can be seen as a compression of the tensor product that does not increase the dimensionality of the representation. The scoring function is then defined as:

$$g_s(e_h, l, e_t) = \sigma(l^T(e_h * e_t))$$

where l is a vector in \mathbb{R}^d representing the relation.

- **TuckER [5]**: Based on Tucker decomposition[46], TuckER decomposes the binary tensor representation of the knowledge graph into two factor matrices ($E \in \mathbb{R}^{|\mathcal{E}| \times d}$, $L \in \mathbb{R}^{|\mathcal{L}| \times d_l}$) and a core tensor $W \in \mathbb{R}^{d \times d_l \times d}$. E represents the head/tail entity embedding matrix, L represents the relation embedding matrix, and W indicates the level of interaction between the different factors and contributes to the parameter-sharing ability. Its scoring function is defined as:

$$g_s(e_h, l, e_t) = W \times_1 e_h \times_2 l \times_3 e_t$$

where e_h, l, e_t are the rows of corresponding factor matrices, and \times_i denotes the mode- i tensor product.

Neural network-based embeddings

- **ConvE [12]**: ConvE takes advantage of convolutional neural networks (CNNs) for computing the embeddings. This method models the complex interactions between subject entities and relations by using convolutional and fully-connected layers. It first applies a two-dimensional convolution layer with filters ω on reshaped and concatenated head entity and relation embeddings, \bar{e}_h and \bar{l} denote a 2D reshaping of e_h and l , respectively. The resulting feature map tensor is then vectorized and projected into d -dimensional entity space using a linear transformation by W and matched with the e_t via an inner product. The scoring function is defined as:

$$g_s(e_h, l, e_t) = f(\text{vec}(f([\bar{e}_h; \bar{l}] * \omega)))W e_t$$

- **RGCN [41]**: RGCN is an extension of a graph convolutional network (GCN) [21] that applies relation-specific transformations by replacing the weighting scheme in which all edges have the same value with a weight that varies depending on the relation type. It consists of the RGCN encoder, which computes entity representations, and the DistMult[54] factorization scoring function. Scores for each triple are defined as:

$$g_s(e_h, l, e_t) = e_h^W \cdot L \cdot e_t^W$$

where W is the final level of the encoder, and $L \in \mathbb{R}^{d \times d}$ is a diagonal relation matrix.

4.2 Profitability prediction

Once we have the vector representation for the assets at time t , we train a regression model to estimate the future profitability of the assets. Formally, the regression model acts as a function $h : \mathbb{R}^D \rightarrow \mathbb{R}$, which receives the feature vector of an asset s at time t and estimates the return of investment of the asset at time $t + \Delta t$. (where Δt is the time window we want to predict for). Notably, any regression model can be used here: from simple methods like linear regression or random forest to more complex approaches like long short-term memory networks (LSTMs) or gated recurrent units (GRUs). Finally, as most downstream use-cases for price prediction involve finding the most profitable assets, we rank all assets by their predicted profitability (in descending order).

As we are in a temporal scenario where we can only train using data prior to the time we are to produce predictions for, given a prediction time t , we train a model using asset feature vectors for different time points preceding t (with the end of the training period referred to as t_{train}). For example, given a training vector for an asset s computed for the training time point $t - 1y$ (1 year before the prediction time), its regression target is the return on investment (ROI) between $t - 1y$ and $t - 1y + \Delta t$, i.e. the percentage change in closing price over the Δt period starting at $t - 1y$. Note that training time points t_{train} must be less than or equal to $t - \Delta t$ to avoid requiring information from the future when evaluating. Return on investment is calculated as follows:

$$\text{ROI}(s, t, \Delta t) = \frac{\text{Close}(s, t + \Delta t) - \text{Close}(s, t)}{\text{Close}(s, t)} \quad (5)$$

where $\text{Close}(s, t)$ represents the closing price of $s \in \mathcal{S}$ at time t . As the loss function for our regression algorithms, we use the squared error:

$$\mathcal{L} = \sum_{t=t_0}^{t_{train}-\Delta t} \sum_{s \in \mathcal{S}} (\text{ROI}(s, t, \Delta t) - h(s(t)))^2 \quad (6)$$

where t_0 is the initial time.

5 Experimental setup

To assess the effectiveness of the two constructed knowledge graphs in predicting stock market profitability, we carry out experiments using U.S. stock market data. Here, we detail the dataset and the experimental setup used for our analysis.

5.1 Dataset

To conduct our experiments, we collected a dataset from three major U.S. stock exchanges: NASDAQ, NYSE, and AMEX.

Pricing data: We collect daily pricing data from Yahoo! Finance⁵ including open, close, high, low, and volume prices for 5,823 assets from January 2018 to September 2022.

⁵ <http://finance.yahoo.com>

Wikidata graph: Using the approach described in Section 3.1, we collected 3,370 assets entities from Wikidata, resulting in more than 100k entities and 450k relations crawled for our knowledge graph. Table 3 summarises the total properties of the crawled Wiki graph.

| Property | Wikidata | 10K |
|--------------------------|----------|--------|
| Number of entities | 102,739 | 8,380 |
| Number of relation types | 114 | 450 |
| Number of links | 457,758 | 36,973 |

Table 3: Graph Properties

10K graph: We successfully retrieved 2,264 assets with 10K reports⁶ based on the linked assets in the Wikidata graph, resulting in 5,399 filings from 2017-2022. We incorporated an additional year of data prior to 2018 to ensure a rich dataset for constructing the initial knowledge graph. Table 3 summarizes the global 10K graph properties.

Dataset split: For each time point, technical indicators and knowledge graph versions from prior dates are used as input to predict price changes over a six-month horizon (Δt). For our experiments, time points on or before the 31st of December 2019 are used for training and the six months following the 30st of June 2020 are used for testing, maintaining a six-month gap between those sets to avoid data leakage. Within the dataset, time points are spaced 1 week apart, selecting the Monday of each week as t . In total, the training set includes 73 time points, while the test set contains 25.

Dataset post-processing: To ensure data consistency and reduce discrepancies, we take the intersection of assets listed in the pricing data, Wikidata entities and 10K reports. This results in 2,042 assets for our training set and 2,096 assets for our testing set. We also exclude 421 upper outliers from our test set when profits exceed 1.5 times the interquartile range above the third quartile, which indicate unusually high profits. These outliers include penny stocks, companies coming back from bankruptcy, and phenomena like the 2021 meme stock trading (e.g., GameStop). Including these assets leads to unstable evaluations, as their presence among the top-ranked assets can significantly skew metrics like ROI@10.

5.2 Metrics

In order to evaluate our predictions, we consider two different metrics: 1) a ranking-oriented metric, monthly return on investment (ROI), and 2) a global error metric, root mean squared error (RMSE). We summarise each below:

- **Monthly return on investment (ROI@k):** we analyse the average monthly return on investment over the top k ranked assets. In our experiments, we take $k = 10$ and compute the ROI over the 6 months following the date of the recommendations.

⁶ <https://sec-api.io/>

Table 4: Summary of financial technical indicators

| Indicator (financial days) | Time period Δt |
|---------------------------------------|------------------------|
| Average price | 28, 63, 126 |
| Return on investment | 28, 63, 126 |
| Volatility | 28, 63, 126 |
| Momentum | 14, 21, 28 |
| Moving average convergence divergence | 26 |
| Rate of change | 14, 21, 28 |
| Relative strength index | 14 |
| Detrended close oscillator | 22 |
| Force index | 1 |
| Minimum | 14, 21, 28 |
| Maximum | 14, 21, 28 |
| Chaikin oscillator | 10 |
| Average true range | 14 |
| Average directional index | 14 |
| Vortex indicator | 14 |

- **Root mean-squared error (RMSE):** To understand the model accuracy, we compute the square root of the average squared difference between predicted and real ROI.

5.3 Model Configuration

Technical indicators: In our experiments, we use 16 different KPIs derived from the pricing time series as technical indicators, summarized in Table 4. In order to generalise the comparison of two knowledge graphs, and reduce the influence of KPIs on the comparison result, we have chosen two groups of technical indicators:

- **BasicKPIs:** average price, return on investment, and volatility.
- **AdvKPIs:** all KPIs in Table 4.

Knowledge graph embeddings: We select a set of both popular and state-of-art KGE models for the experiment, specifically, 9 KGE models are tested:

- **Translation-based embeddings:** TransE [6], TransH [52], TransR [22] and RotatE [43]
- **Factorization-based embeddings:** RESCAL [31], HolE [30] and Tucker [5]
- **Neural network-based embeddings:** ConvE [12] and RGCN [41]

We use the PyKeen library [1] to generate 50-dimensional embeddings for entities (companies) associated with each asset. We repeat embeddings generation 5 different times to analyse the variability across the runs. However, we use just one random seed for the RGCN model due to its high computational cost.

Regression Model We opt to use a Random Forest regression algorithm with 100 trees as our prediction model.

| | | BasicKPIs | | | | AdvKPIs | | | |
|----------------------------|---------------|----------------|----------|---------------------|----------------------------|-----------|----------------|---------------------|----------------------------|
| | | ROI@10 | | RMSE | | ROI@10 | | RMSE | |
| Group | Algorithm | 10K Graph | Wikidata | 10K Graph | Wikidata | 10K Graph | Wikidata | 10K Graph | Wikidata |
| Baseline (Only KPIs) | | 0.010 | | 0.4574 | | 0.0466 | | 0.4299 | |
| Translation-based models | KPIs + TransE | 0.0454* | 0.0368 | 0.4439 [†] | 0.4415 [†] | 0.0474* | 0.0393 | 0.4277 [†] | 0.4289 |
| | KPIs + TransH | 0.0409 | 0.0419 | 0.4408 [†] | 0.4383 [†] | 0.0467 | 0.0434 | 0.4257 [†] | 0.4275 |
| | KPIs + TransR | 0.0435 | 0.0422 | 0.4442 [†] | 0.4385 ^{†*} | 0.0442 | 0.0454 | 0.4276 [†] | 0.4259 [†] |
| | KPIs + RotatE | 0.0427* | 0.0385 | 0.4423 [†] | 0.4371 [†] | 0.0472* | 0.0403 | 0.4254 [†] | 0.4256 [†] |
| Factorization-based models | KPIs + RESCAL | 0.0418 | 0.0418 | 0.4474 [†] | 0.4439 ^{†*} | 0.0451 | 0.0447 | 0.4329 | 0.4243 ^{†*} |
| | KPIs + HolE | 0.0418 | 0.0407 | 0.4448 [†] | 0.4353 ^{†*} | 0.0441 | 0.0442 | 0.4285 | 0.4239 ^{†*} |
| | KPIs + TuckER | 0.0400 | 0.0409 | 0.4442 [†] | 0.4334 ^{†*} | 0.0412 | 0.0445* | 0.4298 | 0.4226 ^{†*} |
| Neural network models | KPIs + ConvE | 0.0407 | 0.0450* | 0.4433 [†] | 0.4304 ^{†*} | 0.0435 | 0.0451 | 0.4269 [†] | 0.4207 ^{†*} |
| | KPIs + RGCN | 0.0423 | 0.0447 | 0.4479 [†] | 0.4192^{†*} | 0.0423 | 0.0502* | 0.4322 | 0.4126^{†*} |
| Market | | 0.0345 | | - | - | 0.0345 | | - | - |
| S&P 500 | | 0.0266 | | - | - | 0.0266 | | - | - |

Table 5: Performance of random forest regression methods with assets embeddings derived from two knowledge graphs, when predicting six months into the future. The best value for each algorithm and metric is highlighted in bold. [†] denotes significant improvements (Wilcoxon test $p < 0.05$) with respect to the only KPIs baseline. * indicates significant improvements compared to the corresponding graph model for the other graph.

6 Results

In this section we compare the impact of incorporating two distinct knowledge graphs sourced from Wikidata and 10K reports on the prediction of asset profitability. In particular, we investigate the following research questions:

- **RQ1:** How does the use of the Wikidata and 10K graphs affect the effectiveness of profitability prediction?
- **RQ2:** How different are the profitable assets recommended for each knowledge graph?

6.1 RQ1: Graph Performance Comparison

We begin by examining the core question posed in this work: how do knowledge graphs derived from financial reports vs. a knowledge base affect financial asset recommendation (FAR) effectiveness? In particular, we would like to know whether one knowledge graph provides more useful information than the other, and whether the approach used to embed the graph for each company impacts performance.

To answer this question, we compare FAR approaches with and without knowledge graph embeddings. In particular, we start with a price-prediction-based baseline referred as *Baseline (Only KPIs)*, which uses past pricing data to predict the future price of an asset. For a day, all assets are ranked by their predicted return-on-investment (ROI) after 6 months. To evaluate performance, we report both error between the prediction and actual ROI (RMSE, lower is better) and the actual (monthly) ROI of the top 10 recommendations (ROI@10, higher is better). We have two baseline variants, denoted BasicKPIs and AdvKPIs, where the latter includes more technical indicators. As we can see from

Table 5, the baseline models achieve an ROI@10 of 4.1% (BasicKPIs) to 4.66% (AdvKPIs), which is higher than both the market average (Market) and S&P 500 (a common index benchmark) for the same period (also reported at the bottom of Table 5).

Having established our baseline, we now contrast this baseline to the same model when augmented with the embeddings derived from our two knowledge graphs. In Table 5, for each metric, we include two columns (10K Graph and WikiData) reporting performance when the baseline is augmented by each knowledge graph. As there are a range of possible graph embedding techniques (see Section 4.1), we include one row for each embedding technique tested, denoted $KPIs + \langle KGE \rangle$ (where $\langle KGE \rangle$ is a knowledge graph embedding approach, e.g. TransE). The best metric values are highlighted in bold, and statistically significant increases (pairwise Wilcoxon test at $p < 0.05$) in comparison to the Baseline (Only KPIs) model is denoted \dagger . We also highlight significance differences between the application of the same model on the two knowledge graphs as $*$.

The first observation is that integrating KGE for profitability prediction generally results in RMSE reductions with respect to the baselines (33/36 times). In 30 cases, this reduction is significant, thus showcasing the capability of knowledge graph information to generate more accurate predictions. When comparing both graphs, the Wikidata KG obtains lower errors in 15 out of 18 cases (with 11 of them showing a significant difference) – therefore showing that this graph provides more accurate results than the 10K graph.

We observe a different pattern when we study the return on investment over the top-10 ranked results however: even when most methods using KGE reduce the prediction error, this fact does not necessarily result in more profitable recommendation rankings. This is particularly notable for the methods using the larger set of indicators, where only four models beat the baseline (TransE, TransH and RotatE for the 10k graph and RGCN for the Wikidata graph). However, for both baselines, it is possible to find at least one model for each graph improving its profitability. In the case of the BasicKPIs baseline, the best models are TransE for the 10K graph (4.54% ROI@10) and ConvE for the Wikidata graph (4.47% ROI@10). For AdvKPIs, TransE is again the best for the 10K graph (4.74% ROI@10), whereas RGCN is the best for Wikidata (5.02% ROI@10). This illustrates that both knowledge graphs are capable of providing a useful profitability signal for the task.

When we compare the effectiveness of the graphs in terms of ROI@10, we also see that there is a different relationship between the complexity of the embedding approach and ROI gain across the two graphs. Specifically, the 10k graph yields higher ROI for the translation-based algorithms (particularly the simpler TransE and RotatE models) that perform poorly when applied on Wikidata. Meanwhile, for the most complex of tested algorithms (TuckER, and both neural network approaches, ConvE and RGCN), the Wikidata graph provides a stronger profitability signal. According to Table 3, the Wikidata graph contains approximately ten times the number of entities and links as the 10K graph, indicating a greater complexity and graph size. Although the simple knowledge graph embedding models are capable of providing useful summaries of the 10k graph information, we hypothesize that the more complex knowledge graph embedding models (specially those based on neural networks) need a much larger number of links

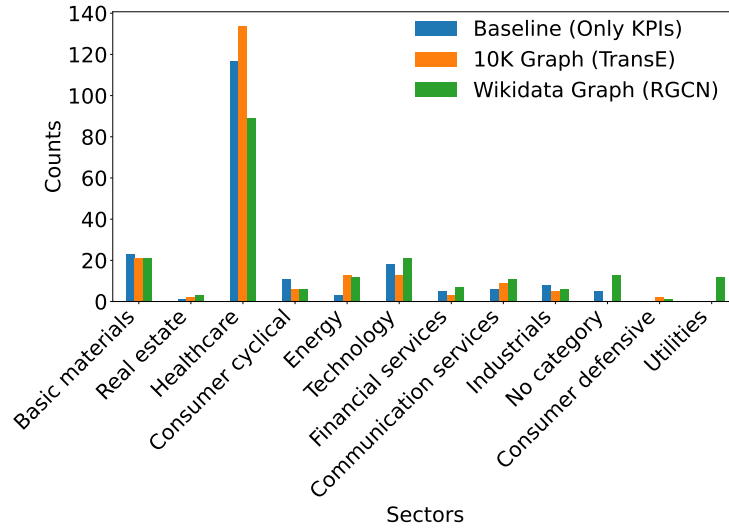


Fig. 4: Distribution of profitable assets in the top-10 recommendation rankings across sectors.

to learn how to extract stronger profitability signals from knowledge graphs – hence why RGCN performs well on Wikidata but not the 10K filings.

To answer RQ1: *Both knowledge graphs are capable of enhancing the accuracy of the predictions – with Wikidata achieving better results. If we look at returns, however, it highly depends on the embedding method used. The simpler translation-based methods favour the use of the smaller 10K graph, whereas the most complex neural-based methods require more information to work, which they can obtain from the Wikidata graph.*

6.2 RQ2: Profitable Asset Sector Analysis

Besides raw algorithm performance, we hypothesize that different knowledge graph construction methodologies lead to the promotion of specific types of assets in the recommendations. For instance, general knowledge graphs might promote well-known companies as they have more information about them. As studying these differences is important to understand the inner workings of these methods, we provide a preliminary analysis where we study the distribution of recommended profitable assets across sectors.

To perform this analysis, we identify assets with positive ROI in the top-10 of the asset rankings and count how many times each sector is represented. We compare two top performing models using the basic indicators: TransE for the 10K graph, and RGCN for the Wikidata graph. Although ConvE provides slightly better performance when using the basic KPIs for the Wikidata graph, we choose RGCN as it is the best overall method for this KG. Both BasicKPIs + TransE (10k) and BasicKPIs + RGCN (WikiData) provide similar ROI@10 values (4.54% vs. 4.47%), but we hypothesise that source of that profitability might be different.

Figure 4 displays the results of our experiment, where the x axis shows the different sectors and the y axis shows the number of profitable assets for each algorithm and sector. In the plot, we also include the baseline using only technical indicators as features, for comparison. For many of the sectors (basic materials, consumer cyclical, energy), similar numbers of profitable assets are selected by both graphs, but there are sectors which highlight the differences between both knowledge bases.

The most important is the healthcare sector. In our data, the studied test period (June-December 2020) runs during the Covid-19 pandemic. Due to the pandemic, the value of healthcare in this period rose. This is observable in our results, as it is the sector counting the biggest number of profitable assets for the three compared models. However, it is the models using the 10K graph that recommends more assets from this sector. Considering that 10K filings contain company projection information, in this case, they enable the model to capture company outlooks on the Covid-19 pandemic and exploit them – something that the Wikidata graph, containing more general information, does not, as it even reduces the number of profitable healthcare recommended assets with respect to the baseline. Instead, the Wikidata graph takes its improvements from other sectors, like technology or utilities, for which the graph might contain more data.

To answer RQ2: *The knowledge graph construction strategy markedly impacts the types of assets recommended and this appears to be driven by the types of relationships and properties captured within each graph, although further investigation will be needed to conclusively show this.*

7 Adaptive Graph Selection

The differences on sector-level performance across knowledge graphs observed in Section 6.2 suggest that each knowledge graph may be better suited to predict the profitability of specific types of stocks. Motivated by this observation, we investigate whether adaptively selecting the suitable graph for each stock can improve profitability. To analyze this, we propose an ensemble method that switches between knowledge graph-based models depending on the stock.

7.1 Algorithm description

We propose a switching ensemble model [7] for profitability prediction, that adaptively selects a previously trained ROI prediction algorithm for each stock. We show the architecture of the ensemble in Figure 5. The ensemble chooses among the possible models following their performance on the training set. We consider two possible strategies for choosing the model for each individual stock:

- **Stock-level strategy:** Given a stock, we choose the profitability prediction algorithm that minimizes RMSE during the training period.
- **Sector-level strategy:** All stocks within a sector are assigned to the same profitability prediction model – the model minimizing the average RMSE of the sector stocks during the training period.

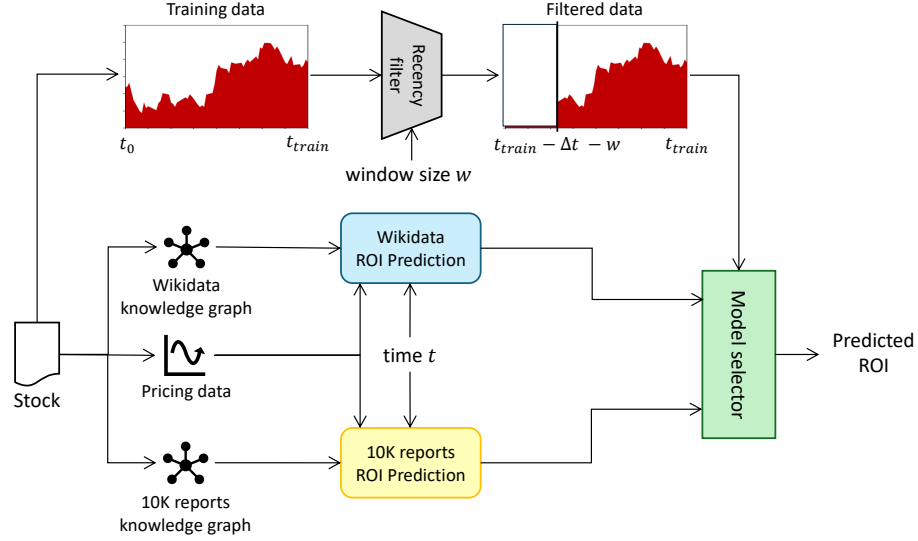


Fig. 5: Architecture of the adaptive ensemble.

While using the complete training data for the model selector is a possibility, in the financial sector, the time of the input information matters. We hypothesize that, by choosing the most recent training points, the performance of our ensemble model should increase. To deal with this, we apply a recency filter to our training data. This filter receives as input a window size w (measured in weeks). If we define t_{train} as the end of the training period, this recency filter keeps the training time points in the $(t_{train} - \Delta t - w, t_{train})$ period.⁷

7.2 Experiments

We compare our adaptive method under the same experimental setup defined in Section 5. To build our ensemble, we use the best-performing KPIs + <KGE> model for each knowledge graph as the baseline: TransE for the 10K graph and RGCN for the Wikidata graph. We also include these two models as baselines in our experiment.

For our ensembles, we experiment with multiple window sizes, ranging from 1 to 78 weeks (corresponding to 73 data points, as some holidays were excluded). This window selection allows us to assess how the length of historical windows for graph selection affects recommendation effectiveness.

⁷ As the last time point of the training period is at $t_{train} - \Delta t$, we take the starting point of the windowed period as $t_{train} - \Delta t - w$ to take all the examples within a period of length w .

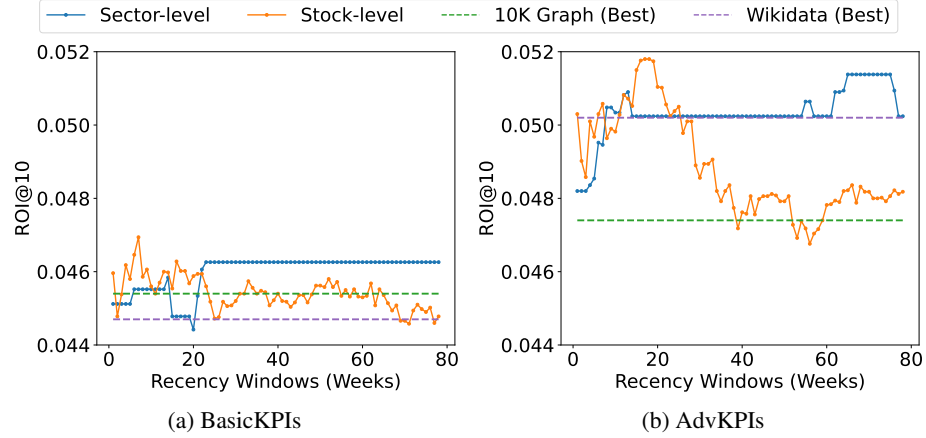


Fig. 6: Comparison of ROI@10 Across Different Graph Selection Strategies

7.3 Results

In this section, we examine the influence of adaptive graph selection strategy on asset recommendation performance. Specifically, we focus on the following research questions:

- **RQ3:** How does the adaptive combination of two knowledge graphs impact the effectiveness of investment profitability?
- **RQ4:** How does the choice of recency window length affect the proportion of each graph selected?

RQ3: Dynamic Selection Performance We begin by addressing the key question in the section: whether dynamically choosing the predicted best graph for each stock or sector can enhance profitability in asset recommendation. Thus, we compare the performance of both stock-level and sector-level strategies across all possible recency windows to examine the overall performance against the performance of the best KPIs + <KGE> models (that use a single knowledge graph as input). Figure 6 illustrates the ROI@10 performance of the tested methods for different KPIs – BasicKPIs on Figure 6(a), on the left, and AdvKPIs on Figure 6(b), on the right. Here, the x-axis represents the length of the recency window in weeks (as we go towards the right part of each graph, we are using more – and older – examples to choose the best model for each stock). Figure 6 shows, in orange, the adaptive stock-level model and, in blue, the adaptive sector-level model. We include, as reference the best model for the 10K graph (KPIs + TransE) and the Wikidata graph (KPIs + RGCN), respectively, as green and purple dashed lines.

We observe distinct patterns achieved by the two adaptive strategies. The stock-level approach exhibits volatile performance and outperforms the single-graph baseline over shorter recency windows—up to 20 weeks for BasicKPIs and 15–25 weeks for

AdvKPIs. The highest improvement reaches 3.39% (0.04694 vs. 0.0454) for BasicKPIs and 3.19% (0.05180 vs. 0.0502) for AdvKPIs, compared to the best-performing single-graph method. In contrast, the sector-level method demonstrates more stable performance over longer recency windows and consistently outperforms or performs comparably to the best-performing baseline when the window size exceeds 20 weeks, with the highest improvement of 1.89% (0.04626 vs. 0.0454) and 2.35% (0.05138 vs. 0.0502) for BasicKPIs and AdvKPIs, respectively. This suggests that shorter recency windows are more suitable for fine-grained selection, whereas longer windows are better suited for coarse-grained sector-level strategies that benefit from broader temporal aggregation.

To answer RQ3: *An adaptive graph selection method that predicts and assigns a graph to each asset can lead to improved recommendation ROI. However, our findings suggest that the recency window has a significant impact on the performance of different adaptive selection strategies: shorter windows benefit stock-level strategy by emphasizing recent fluctuations, while longer windows support sector-level methods through stable temporal smoothing.*

RQ4: Effect of Recency Window Length As the recency window sizes plays an important role in the way of dynamically combine different graphs, we hypothesize that different varying window lengths promote the use of different graphs. In particular, we investigate how the length of the recency window influences the graph selection across both the top-10 recommended stocks and the entire stock set. Figure 7 and Figure 8 show the variation in the percentage of 10K Graph usage across different recency windows (x-axis) for BasicKPIs and AdvKPIs, respectively. For each KPI set, results are presented for both stock-level and sector-level strategies, with proportions shown separately for the top-10 recommended stocks (Figures 7 (a) and 8(a)) and for the full stock set (Figures 7(b) and 8(b)).

We first compare the graph selection with the profitability of our adaptive approaches. A first observation from Figures 7 and 8 is that, as the recency window increases, we observe a trend toward more balanced and stable use of the graph across the entire stock set, in both stock and sector-level configurations. This is particularly noteworthy in the sector level strategy, which barely modifies the stock allocation to each method on recency windows bigger than 20 weeks – explaining the stability of the ROI performance for this method in Figure 6. Indeed, the improvement in performance under long recency windows in Figure 6(b) can be explained by the increase of usage of the 10K graph when we use those long recency windows. The performance of the stock-level strategy, however, it is not so easily explainable by changes in graph usage: while the usage of different graphs tends to stabilize as we consider longer time periods, the performance does not. This implies that, although the proportion of stocks selected for each graph is the same, the stocks are not – making the stock selection strategy very sensitive to changes in the recency window.

Interestingly, there are changes in the stock selection trend when we look at the top-10 selection (Figures 7 and 8(a)) and the complete set of stocks (Figures 7 and 8 (b)). When we look at the full set of stocks, there is a different dominant graph for BasicKPIs and AdvKPIs – specially when we use longer recency windows. The BasicKPIs model

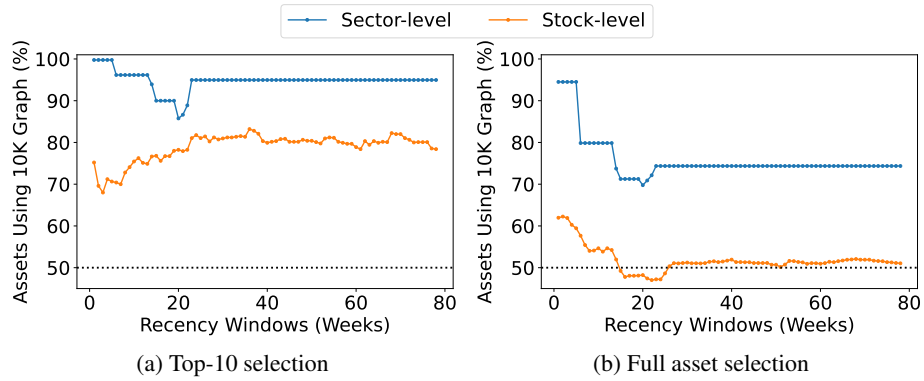


Fig. 7: A comparison of the proportion of 10K graphs selected (a) in the top 10 recommended stocks and (b) across all stocks under the BasicKpis setting

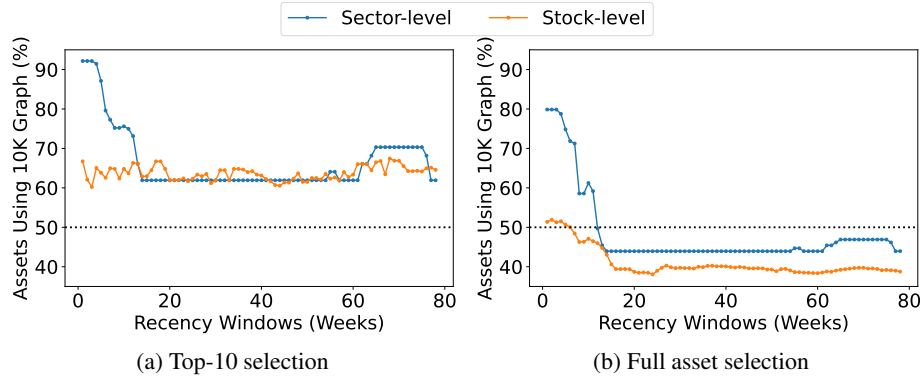


Fig. 8: A comparison of the proportion of 10K graphs selected (a) in the top 10 recommended stocks and (b) across all stocks under the AdvKpis setting

tends to favor the 10K graph – which is used to predict the profitability of slightly more than 50% of the stocks predicted by the stock-level model, and between 70-95% of the stocks predicted by the sector-level model. On the other hand, the AdvKPIs model tends to favor the Wikidata graph – limiting the usage of the 10K graph to 40-50% of the stocks. Considering the overall ROI@10 performance reported in Section 6.1, where the best BasicKPIs model uses the 10K graph, and the best AdvKPIs model uses the Wikidata graph, this is what we expected for both the top-10 and the complete selection. However, for both BasicKPIs and AdvKPIs, in the top-10 assets, the selection is dominated by the 10K graph (predicting between 60-99% of the stocks in the top-10, depending on the recency window).

Following Figure 4, this mismatch between the top-10 and the complete set of stocks might be explained by the preference of the 10K graph to promote healthcare stocks. These stocks were highly important during the Covid-19 pandemic period on which the test period runs – and their selection on the top 10 explains the preference of the 10K

graph over the Wikidata graph for the top-10 stocks. However, both sector and stock-based adaptive models can benefit from a mixture of the two graphs – as it allows a greater diversification of assets in the recommendation and greater performance.

To answer RQ4: *The length of recency window has differing effects on graph usage patterns and asset selection for investment – with longer recency windows leading to more stable graph usage selections. While overall graph selection appears relatively balanced across the full stock set, the top-10 recommended assets consistently favor the 10K Graph across most settings.*

8 Conclusion & Future Work

In this work, we have explored the impact that two KG construction strategies have when predicting the future returns of U.S. stocks. For this, we collected a Wikidata subgraph and a built a graph by automatically extracting factoids from annual 10K filings. We have compared these methods under a unified FAR model that estimates the profitability of stocks. This method integrates price technical indicators with asset KG vectors extracted from the graphs by 9 different knowledge graph embedding models.

Our findings show that both graph types can improve the profitability of recommendations with respect to only using price information by up-to 10.7%. However, different graphs favour different embedding strategies: graphs extracted from financial reports tend to be smaller, and therefore benefit from translation-based models like TransE [6] or RotatE [43], whereas the bigger Wikidata graph favours complex neural network models like RGCN [41].

We have also analysed the distribution of the profitable assets recommended by the models across sectors, showing that different knowledge graph construction strategies might present biases towards certain types of assets. In our experiments, the 10K graph has been able to leverage the information regarding global events (in particular, the Covid-19 pandemic) available in the reports to promote profitable healthcare stocks, while the more static Wikidata graph has identified profitable assets in sectors like utilities. Based on this observation, we further investigated the effectiveness of adaptive graph selection strategies at stock and sector levels. Our findings indicate that dynamically selecting graphs based on past performance can enhance profitability, supporting that different knowledge graphs indeed offer complementary strengths for different asset types. However, different strategies exhibit varying preferences for recency window sizes, reflecting their sensitivity to temporal granularity. Moreover, while longer windows generally result in more balanced graph usage, the top-10 assets consistently favor the 10K Graph, largely due to its alignment with high-performing healthcare sectors during the Covid-19 pandemic.

As future work, we aim to compare these knowledge graphs with others that include other types of financial information, such as news or press releases. We also aim to further investigate more fine-grained adaptive methods that integrate additional properties of different knowledge graphs for improved stock recommendation. Finally, as in this work we have only used random forests, we aim to test other FAR algorithms, including those directly targeting asset ranking [2].

References

1. Ali, M., Berrendorf, M., Hoyt, C.T., Vermue, L., Sharifzadeh, S., Tresp, V., Lehmann, J.: PyKEEN 1.0: A Python Library for Training and Evaluating Knowledge Graph Embeddings. *Journal of Machine Learning Research* **22**(82), 1–6 (2021)
2. Alsulmi, M.: From Ranking Search Results to Managing Investment Portfolios: Exploring Rank-Based Approaches for Portfolio Stock Selection. *Electronics* **11**(23), 4019 (2022). <https://doi.org/10.3390/electronics11234019>
3. Alzaman, C.: Deep learning in stock portfolio selection and predictions. *Expert Systems with Applications* **237**, 121404 (2024). <https://doi.org/10.1016/j.eswa.2023.121404>
4. Bach, N., Badaskar, S.: A review of relation extraction. *Literature review for Language and Statistics II* **2**, 1–15 (2007)
5. Balazevic, I., Allen, C., Hospedales, T.: TuckER: Tensor Factorization for Knowledge Graph Completion. In: 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP 2019). pp. 5185–5194. Association for Computational Linguistics, Hong Kong, China (2019). <https://doi.org/10.18653/v1/D19-1522>
6. Bordes, A., Usunier, N., Garcia-Durán, A., Weston, J., Yakhnenko, O.: Translating Embeddings for Modeling Multi-relational Data. In: 27th Conference on Neural Information Processing Systems (NeurIPS 2013). Curran Associates, Inc., Stateline, Nevada, USA (2013)
7. Burke, R.: Hybrid Web Recommender Systems, pp. 377–408. Springer Berlin Heidelberg, Berlin, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72079-9_12, https://doi.org/10.1007/978-3-540-72079-9_12
8. Cai, H., Zheng, V.W., Chang, K.C.C.: A Comprehensive Survey of Graph Embedding: Problems, Techniques, and Applications. *IEEE Transactions on Knowledge and Data Engineering* **30**(9), 1616–1637 (2018). <https://doi.org/10.1109/TKDE.2018.2807452>
9. Chen, Q.: Stock movement prediction with financial news using contextualized embedding from BERT. *CoRR abs/2107.08721* (2021)
10. Cheng, D., Yang, F., Wang, X., Zhang, Y., Zhang, L.: Knowledge Graph-based Event Embedding Framework for Financial Quantitative Investments. In: 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2020). pp. 2221–2230. ACM, Online, China (2020). <https://doi.org/10.1145/3397271.3401427>
11. Deng, S., Zhang, N., Zhang, W., Chen, J., Pan, J.Z., Chen, H.: Knowledge-Driven Stock Trend Prediction and Explanation via Temporal Convolutional Network. In: The Web Conference 2019 (WWW 2019 Companion). pp. 678–685. ACM, San Francisco, CA, USA (2019). <https://doi.org/10.1145/3308560.3317701>
12. Dettmers, T., Minervini, P., Stenetorp, P., Riedel, S.: Convolutional 2D Knowledge Graph Embeddings. In: 32nd of the AAAI Conference on Artificial Intelligence (AAAI 2018). pp. 1811–1818. AAAI Press, New Orleans, LA, USA (2018). <https://doi.org/10.1609/aaai.v32i1.11573>
13. Elhammadi, S., V.S. Lakshmanan, L., Ng, R., Simpson, M., Huai, B., Wang, Z., Wang, L.: A High Precision Pipeline for Financial Knowledge Graph Construction. In: 28th International Conference on Computational Linguistics (COLING 2020). pp. 967–977. International Committee on Computational Linguistics, Online, Barcelona, Spain (2020). <https://doi.org/10.18653/v1/2020.coling-main.84>
14. Feng, F., He, X., Wang, X., Luo, C., Liu, Y., Chua, T.S.: Temporal Relational Ranking for Stock Prediction. *ACM Transactions on Information Systems* **37**(2), 1–30 (2019). <https://doi.org/10.1145/3309547>

15. Griffin, P.A.: Got information? investor response to form 10-k and form 10-q edgar filings. *Review of Accounting Studies* **8**, 433–460 (2003)
16. Hogan, A., Blomqvist, E., Cochez, M., D’amato, C., Melo, G.D., Gutierrez, C., Kirrane, S., Gayo, J.E.L., Navigli, R., Neumaier, S., Ngomo, A.C.N., Polleres, A., Rashid, S.M., Rula, A., Schmelzeisen, L., Sequeda, J., Staab, S., Zimmermann, A.: Knowledge Graphs. *ACM Computing Surveys* **54**(4), 71:1–71:37 (2021). <https://doi.org/10.1145/3447772>
17. Hu, X., Wen, L., Xu, Y., Zhang, C., Yu, P.: SelfORE: Self-supervised relational feature learning for open relation extraction. In: Webber, B., Cohn, T., He, Y., Liu, Y. (eds.) 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP 2020). pp. 3673–3682. Association for Computational Linguistics, Online (Nov 2020). <https://doi.org/10.18653/v1/2020.emnlp-main.299>, <https://aclanthology.org/2020.emnlp-main.299>
18. Hu, Z., Liu, W., Bian, J., Liu, X., Liu, T.Y.: Listening to Chaotic Whispers: A Deep Learning Framework for News-oriented Stock Trend Prediction. In: 11th ACM International Conference on Web Search and Data Mining (WSDM 2018). pp. 261–269. ACM, Los Angeles, CA, USA (2018). <https://doi.org/10.1145/3159652.3159690>
19. Kaur, S., Smiley, C., Gupta, A., Sain, J., Wang, D., Siddagangappa, S., Aguda, T., Shah, S.: REFinD: Relation Extraction Financial Dataset. In: the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2023). pp. 3054–3063. ACM, Taipei, Taiwan (2023)
20. Kertkeidkachorn, N., Nararatwong, R., Xu, Z., Ichise, R.: FinKG: A Core Financial Knowledge Graph for Financial Analysis. In: 17th IEEE International Conference on Semantic Computing (ICSC 2023). pp. 90–93. IEEE, Laguna Hills, CA, USA (2023). <https://doi.org/10.1109/ICSC56153.2023.00020>
21. Kipf, T.N., Welling, M.: Semi-Supervised Classification with Graph Convolutional Networks. In: 10th International Conference on Learning Representations (ICLR 2017). Open-Review, Toulon, France (2017)
22. Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X.: Learning Entity and Relation Embeddings for Knowledge Graph Completion. In: 29th AAAI Conference on Artificial Intelligence (AAAI 2015). pp. 2181–2187. AAAI Press, Austin, TX, USA (2015). <https://doi.org/10.1609/aaai.v29i1.9491>
23. Long, J., Chen, Z., He, W., Wu, T., Ren, J.: An integrated framework of deep learning and knowledge graph for prediction of stock price trend: An application in Chinese stock exchange market. *Applied Soft Computing* **91**, 106205 (2020). <https://doi.org/10.1016/j.asoc.2020.106205>
24. McCreadie, R., Perakis, K., Srikrishna, M., Droukas, N., Pitsios, S., Prokopaki, G., Perdikiouri, E., Macdonald, C., Ounis, I.: Next-Generation Personalized Investment Recommendations. In: Soldatos, J., Kyriazis, D. (eds.) *Big Data and Artificial Intelligence in Digital Finance: Increasing Personalization and Trust in Digital Finance using Big Data and AI*, pp. 171–198. Springer (2022). https://doi.org/10.1007/978-3-030-94590-9_10
25. Mendes, P.N., Jakob, M., García-Silva, A., Bizer, C.: DBpedia Spotlight: Shedding Light on the Web of Documents. In: 7th International Conference on Semantic Systems (I-Semantics 2011). p. 1–8. ACM, Graz, Austria (2011). <https://doi.org/10.1145/2063518.2063519>
26. Murtagh, F., Legendre, P.: Ward’s hierarchical agglomerative clustering method: which algorithms implement ward’s criterion? *Journal of classification* **31**, 274–295 (2014)
27. Naik, N., Mohan, B.R.: Stock Price Movements Classification Using Machine and Deep Learning Techniques-The Case Study of Indian Stock Market. In: 20th International

- Conference on Engineering Applications of Neural Networks (EANN 2019). pp. 445–452. Springer, Hersonissons, Crete, Greece (2019). https://doi.org/10.1007/978-3-030-20257-6_38
28. Neely, C.J., Rapach, D.E., Tu, J., Zhou, G.: Forecasting the Equity Risk Premium: The Role of Technical Indicators. *Management Science* **60**(7), 1772–1791 (2014). <https://doi.org/10.1287/mnsc.2013.1838>
 29. Nelson, D.M.Q., Pereira, A.C.M., Oliveira, R.A.: Stock market’s price movement prediction with LSTM neural networks. In: 2017 International Joint Conference on Neural Networks (IJCNN 2017). pp. 1419–1426. IEEE, Anchorage, AK, USA (2017). <https://doi.org/10.1109/IJCNN.2017.7966019>
 30. Nickel, M., Rosasco, L., Poggio, T.: Holographic Embeddings of Knowledge Graphs. In: 30th AAAI Conference on Artificial Intelligence (AAAI 2016). pp. 1955–1961. AAAI Press, Phoenix, AZ, USA (2016). <https://doi.org/10.1609/aaai.v30i1.10314>
 31. Nickel, M., Tresp, V., Kriegel, H.P.: A Three-Way Model for Collective Learning on Multi-Relational Data. In: 28th International Conference on Machine Learning (ICML 2011). pp. 809–816. Omnipress, Bellevue, WA, USA (2011)
 32. Pejić Bach, M., Krstić, Ž., Seljan, S., Turulja, L.: Text mining for big data analysis in financial sector: A literature review. *Sustainability* **11**(5), 1277 (2019). <https://doi.org/10.3390/su11051277>
 33. Pujara, J.: Extracting Knowledge Graphs from Financial Filings: Extended Abstract. In: 3rd International Workshop on Data Science for Macro-Modeling with Financial and Economic Datasets (DSMM 2017), colocated with the 2017 International Conference on Management of Data (SIGMOD/PODS 2017). pp. 5:1–5:2. ACM, Chicago, IL, USA (2017). <https://doi.org/10.1145/3077240.3077246>
 34. Qi, P., Zhang, Y., Zhang, Y., Bolton, J., Manning, C.D.: Stanza: A Python natural language processing toolkit for many human languages. In: 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations (ACL 2020). pp. 101–108. Association for Computational Linguistics, Online (2020). <https://doi.org/10.18653/v1/2020.acl-demos.14>
 35. Rather, A.M., Agarwal, A., Sastry, V.N.: Recurrent Neural Network and a Hybrid Model for Prediction of Stock Returns. *Expert Systems with Applications* **42**(6), 3234–3241 (2015). <https://doi.org/10.1016/j.eswa.2014.12.003>
 36. Reimers, N., Gurevych, I.: Sentence-bert: Sentence embeddings using siamese bert-networks. In: 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP 2019). pp. 3982–3992. Association for Computational Linguistics, Hong Kong, China (2019). <https://doi.org/10.18653/v1/D19-1410>
 37. Rossi, A., Barbosa, D., Firmani, D., Matinata, A., Merialdo, P.: Knowledge Graph Embedding for Link Prediction: A Comparative Analysis. *ACM Transactions on Knowledge Discovery from Data* **15**(2), 14:1–14:49 (2021). <https://doi.org/10.1145/3424672>
 38. Roziere, B., Gehring, J., Gloeckle, F., Sootla, S., Gat, I., Tan, X.E., Adi, Y., Liu, J., Re-meze, T., Rapin, J., et al.: Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950* (2023)
 39. Sainz, O., García-Ferrero, I., Agerri, R., de Lacalle, O.L., Rigau, G., Agirre, E.: GoL-LIE: Annotation guidelines improve zero-shot information-extraction. In: The 12th International Conference on Learning Representations (ICLR 2024). OpenReview, Vienna, Austria (2024), <https://openreview.net/forum?id=Y3wpuxd7u9>
 40. Sanz-Cruzado, J., McCreddie, R., Droukas, N., Macdonald, C., Ounis, I.: On Transaction-Based Metrics as a Proxy for Profitability of Financial Asset Recommendations. In: 3rd International Workshop on Personalization & Recommender Systems in Financial Services

- (FinRec 2022), colocated with the 16th ACM Conference on Recommender Systems (RecSys 2022). pp. 1–9. Seattle, WA, USA (2022)
41. Schlichtkrull, M., Kipf, T.N., Bloem, P., van den Berg, R., Titov, I., Welling, M.: Modeling Relational Data with Graph Convolutional Networks. In: 15th European Semantic Web Conference (ESWC 2018). pp. 593–607. Springer, Heraklion, Greece (2018). https://doi.org/10.1007/978-3-319-93417-4_38
 42. Sun, Y., Fang, M., Wang, X.: A Novel Stock Recommendation System Using Guba Sentiment Analysis. *Personal and Ubiquitous Computing* **22**(3), 575–587 (2018). <https://doi.org/10.1007/s00779-018-1121-x>
 43. Sun, Z., Deng, Z.H., Nie, J.Y., Tang, J.: RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. In: 7th International Conference on Learning Representations (ICLR 2019). OpenReview, New Orleans, LA, USA (2019)
 44. Takayanagi, T., Chen, C.C., Izumi, K.: Personalized dynamic recommender system for investors. In: Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval. p. 2246–2250. SIGIR '23, Association for Computing Machinery, New York, NY, USA (2023). <https://doi.org/10.1145/3539618.3592035>
 45. Takayanagi, T., Izumi, K., Kato, A., Tsunedomi, N., Abe, Y.: Personalized stock recommendation with investors' attention and contextual information. In: Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval. p. 3339–3343. SIGIR '23, Association for Computing Machinery, New York, NY, USA (2023). <https://doi.org/10.1145/3539618.3591850>
 46. Tucker, L.R.: The Extension of Factor Analysis to Three-Dimensional Matrices. In: Contributions to Mathematical Psychology, pp. 109–127. Holt, Rinehart and Winston (1964)
 47. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: 6th International Conference on Learning Representations (ICLR 2018). OpenReview, Vancouver, BC, Canada (2018)
 48. Vrandečić, D., Krötzsch, M.: Wikidata: A Free Collaborative Knowledgebase. *Communications of the ACM* **57**(10), 78–85 (2014). <https://doi.org/10.1145/2629489>
 49. Wang, M., Qiu, L., Wang, X.: A Survey on Knowledge Graph Embeddings for Link Prediction. *Symmetry* **13**(3), 485:1–485:29 (2021). <https://doi.org/10.3390/sym13030485>
 50. Wang, T., Guo, J., Shan, Y., Zhang, Y., Peng, B., Wu, Z.: A knowledge graph-GCN-community detection integrated model for large-scale stock price prediction. *Applied Soft Computing* **145**, 110595 (2023). <https://doi.org/10.1016/J.ASOC.2023.110595>
 51. Wang, X., He, X., Cao, Y., Liu, M., Chua, T.S.: KGAT: Knowledge Graph Attention Network for Recommendation. In: 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD 2019). pp. 950–958. ACM, Anchorage, AK, USA (2019). <https://doi.org/10.1145/3292500.3330989>
 52. Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge Graph Embedding by Translating on Hyperplanes. In: 28th AAAI Conference on Artificial Intelligence (AAAI 2014). pp. 1112–1119. AAAI Press, Québec City, Québec, Canada (2014). <https://doi.org/10.1609/aaai.v28i1.8870>
 53. Wu, L., Petroni, F., Josifoski, M., Riedel, S., Zettlemoyer, L.: Zero-shot Entity Linking with Dense Entity Retrieval. In: 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP 2020). pp. 6397–6407. Association for Computational Linguistics, Online (2020)
 54. Yang, B., Yih, W.t., He, X., Gao, J., Deng, L.: Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In: Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015). San Diego, CA, USA (2015)

55. Zhang, L., Aggarwal, C., Qi, G.J.: Stock Price Prediction via Discovering Multi-Frequency Trading Patterns. In: 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2017). pp. 2141–2149. ACM, Halifax, Nova Scotia, Canada (2017). <https://doi.org/10.1145/3097983.3098117>
56. Zhang, Y., Yang, K., Du, W., Xu, W.: Predicting Stock Price Movement Direction with Enterprise Knowledge Graph. In: 22nd Pacific Asia Conference on Information Systems (PACIS 2018). p. 237. Yokohama, Japan (2018)
57. Zhao, Y., Du, H., Liu, Y., Wei, S., Chen, X., Zhuang, F., Li, Q., Kou, G.: Stock Movement Prediction Based on Bi-Typed Hybrid-Relational Market Knowledge Graph via Dual Attention Networks. *IEEE Transactions on Knowledge and Data Engineering* **35**(8), 8559–8571 (2023). <https://doi.org/10.1109/TKDE.2022.3220520>