Project Weave

Ipek Sahbazoglu[1], Yigit Ihlamur[2]
[1] University of Oxford, UK
[2] Vela Partners, US

Abstract

The investment industry continously generates significant amounts of data. Ventech allows this data relating to the markets, companies, investors and founders to be used to drive robust decisions. The majority of models built on this data uses Euclidean data which does not capture the relations between the data. Project Weave builds a builds a pipeline to convert tabular market data on start-ups, investors and founders into a knowledge base (KB), a database that captures the relations between the data, and a knowledge graph, visualising and connecting all data in a non-Euclidean space. The knowledge base and knowledge graph essentially weave the vast amount of data available without loosing the context. Project Weave also explores the potential information that can be extracted from these data structures by applying novel Graph Machine Learning technologies, providing future directions.
Keywords: Knowledge Base; Knowledge Graph;Graph Machine Learning; VenTech; Start-Up

1 Motivation & Project Overview

The investment industry is increasingly relying on AI-enabled models that utilise the vast amount of data available on markets, companies and other components of the venture ecosystem to make smarter and robust investment decisions. The insights generated by novel AI-enabled and data-driven models can be leveraged to predict variables pertaining to the market, start-up success/growth potential, behaviour of other VCs and more. Significant amounts of market data are available to fuel these models, relating to a variety of attributes of a given start-up company, investor company, founders etc. The existing ML models for investment recommendations often focus on a subset of this data, and do not utilise relations between the data available. For example, the investor count may be used for an ML model but the fact that the information is the investor count is omitted. Recent developments ML allowed the well-known models and techniques to be extended to non-Euclidean spaces such as graphs. A knowledge graph (KG) is a multi-relational graph comprising various entities and different types of relations among entities[6]. These nodes and edges are stored in a knowledge base (KB) in the form of triples specifying the entities and their relation. By building a KB out of tabular data, and constructing the KG the relations among the data can also be utilised when performing machine learning tasks. The insights driven from machine learning over knowledge graphs can have a powerful competitive edge, by the inclusion of relations, over insights driven from tabular data. Project Weave provides a pipeline to build a KB and generate a KG given tabular data. It also explores learning over knowledge graphs to perform inductive and predictive GML tasks. The following sections detail the data used, the methodology for the KB and KG pipeline as well as the experimentation with GML over the graphs generated.

2 Data Structure & Feature Engineering

The knowledge base created for this project uses data from Moneyball 2.0 database, enriched founder Linkedin data set (+webscraping data) and investor profile data set. Most of the information in the data sets are used only a couple columns were dropped due to redundancy or not being fit for this project. The raw data used for this project is detailed by data set and its columns in Section 2.1. Additionally, three features were built to convert certain variables from continuous to discrete, in efforts to make the graph more readable and intuitive. The features will be further explained in Section 2.2. Finally, the Moneyball data set has a text column that stores the company description. Each description text will be lifted into a knowledge graph using the NLP task: Named Entity Extraction. The KGs generated for each company description will then be merged with the global KG. Since this process concerns building a KG and not data pre-processing, it will be detailed in Section 3

2.1 Data sets

Moneyball 2.0 Database This database has two data sets pertaining to successful and unsuccessful companies. Both data sets have the same columns, and the following columns were used for this project:

- org_uuid: Unique key of company
- org_name: Name of company
- status: Specifies if company is acquired, operating, ipo or closed
- founded_on: The date the company was founded
- category_group_list: A list of categories/domains the company product/service is in.
- city: The city the company is registered in.
- country_code: The country code of where the company is registered.
- long_description: Description of the company
- founder_linkedin_url: Linkedin profile links of company founders
- investment_type: Specifies of the investment is pre-seed, seed, series a or angel.
- raised_amount_usd: Total amount raised from investments in US dollars.
- investor_count: Number of investors.
- investor_name: Unique key and name of company.
- investor_uuids: Unique key of company.

Founder Linkedin Data Set and Webscraping The enriched founder dataset contains information extracted from the Linkedin profiles of the company founders. The webscraping script is to extract and update the data as needed. The columns of the founder data used for this project are as follows:

- org_uuid: Unique key of company
- founder_name: Name of Founder
- founder_school: List of academic institutions the founder attended.
- founder_degree: List of degrees completed at the academic institutions listed.
- founder_exp: List of current/past companies the founder worked at.
- founder_roles: List of roles undertaken at the companies listed.

Investor Profiles Data Set The investor dataset contains detailed information on the investors that invested in the companies in the Moneyball database. The columns used are as follows:

- investor_uuid: Unique key of company
- investor_name: Name of Founder
- investor_types: Specifies a list of types for the investor (incubator, vencture capital, investment bank etc.).
- investment_count: Number of investments made by the investor.
- founded_on: The date the investor company was founded.

Table 1: Statistics

	min	max	mean	standard deviation
investor_count	1	98	8	8
investment_count	1	4.6e3	24	92
investment_amount	3.1e5	1.1e9	3.2e7	6.4e7

2.2 Feature Generation

As mentioned in the introduction, a knowledge base contains information in the form of triplets:{entity_1, relation, entity_2} For the knowledge graph constructed from this base to be useful, intuitive and readable these entities need to be discrete and descriptive variables. Most of the data in the data sets detailed above in Section 2.1 are categorical and require no further pre-processing apart from filling/dropping missing values. There were only 3 continuous variables that were discretised to yield categorical features: (i) raised_amount_usd, (ii) investor_count and (iii) investment_count. Normalisation was applied to all of the raw data to understand and visualise the underlying distribution. The relevant statistics of the variables are tabularised in Table 1. The specific normalisation method used is mean normalisation follows the following formula:

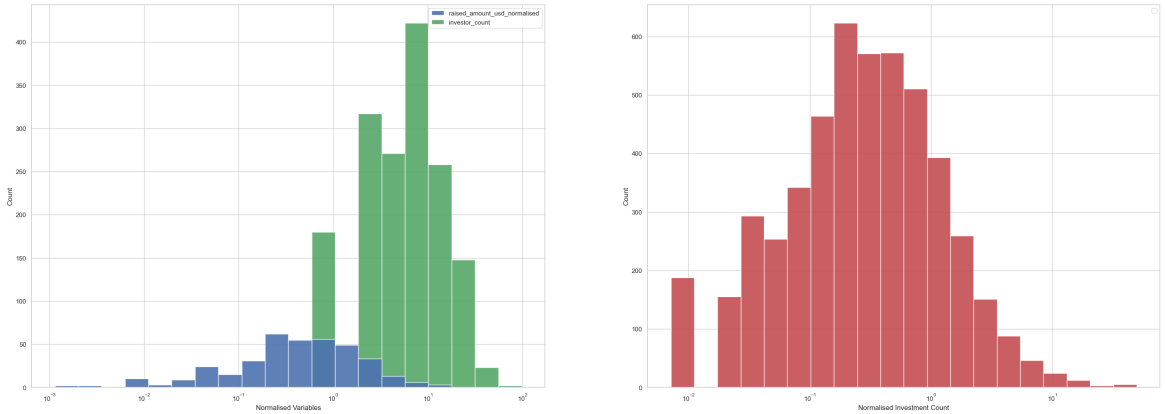
$$\tilde{x} = \frac{x - \mu}{\sigma} \tag{1}$$

$$x, \mu, \sigma = \text{data, mean, standard deviation}$$

2.2.1 Investment Size

This feature is the categorical variable pertaining to the amount of investment raised by the company. As detailed above the Moneyball 2.0 data provides the amount invested in a company per investment type. The purpose of this feature is to categorise the companies according to the total amount of investment they received. To this end, the raised_amount_usd column was grouped by the org_uuid. Figure1(a) shows the histogram for the distribution of the total amount of investment received per company. The investment amount distribution were clustered around the lower and higher ends of the range, with most of the data towards the lower end. The x-axis has logarithmic scale to account for this distribution and provide better visualisations. The distribution was analysed to conclude discretising the data into three categories effectively capture the data and the normalised/raw boundaries were set as follows:

$$\begin{cases} low & \tilde{x} \leq 0.1 \\ mid & 0.1 < \tilde{x} \leq 2 \\ high & \tilde{x} > 2 \end{cases} \qquad \begin{cases} low & x \leq 3.8e7 \\ mid & 3.8e7 < x \leq 1.6e8 \\ high & x > 1.6e8 \end{cases}$$



(a) Company Data

(b) Investor Data

Figure 1: Histograms

2.2.2 Investor Number

This feature is the categorical variable pertaining to the number of investment received by the company. Similar to investment size, this variable was separated according to investment type and it was grouped to get the aggregated values per company. Again the distribution clustered around the extremes so Figure1(a) shows the histogram with a logarithmic scale. The data can be accurately captured with three categories and the bounds for the categories are below (normalised and raw respectively):

$$\begin{cases} low & \tilde{x} \leq 0.1 \\ mid & 0.1 < \tilde{x} \leq 1.5 \\ high & \tilde{x} > 2 \end{cases} \qquad \begin{cases} low & x \leq 9 \\ mid & 9 < x \leq 20 \\ high & x > 20 \end{cases}$$

2.2.3 Investment Count

This feature specifies the number of investments made by the investor company. Figure1(b) shows the histogram for this variable on a logarithmic scale and an analysis of the distribution concluded that four categories for investment size is suitable. The need for an additional category is expected as the distribution range for investment count is higher compared to the other two continuous variables. The bounds for the the four categories are below (normalised/raw):

$$\begin{cases} A & \tilde{x} > 10 \\ B & 1 < \tilde{x} \leq 10 \\ C & 0.1 < \tilde{x} \leq 1 \\ D & \tilde{x} \leq 0.1 \end{cases} \qquad \begin{cases} A & x > 947 \\ B & 1 < x \leq 947 \\ C & 33 < x \leq 116 \\ D & x \leq 33 \end{cases}$$

3 Methodology

Knowledge graphs consist of nodes and edges which specify entities and relations respectively. The entities refer to objects, organisations, locations and people: e.g. car, United Nations, London, Ada Lovelace. The edges represent the semantic relation between two entities. A triple (also defined as [head, link, tail]) is the smallest connected element in a KG, representing the relation between two entities. Knowledge graphs structurally represent all triples contained in a set of entities and relations. A knowledge base (KB) is an intelligent database to manage, store and retrieve structured/unstructured data [7]. For this project KB class was created to store triples generated from the tabular data detailed in Section2 and this database was used to generate the KG. The KB stores the entities and the relations. The entities are all the unique entries we extract from our data. Relations contains the triples in an array of dictionaries with the keys head, link, tail. The knowledge graph is constructed from this knowledge base by interpreting the entities as nodes, and relations as the edge list. This section details the methodology for constructing the knowledge base (3.1) and generating the knowledge graph (3.2)

3.1 Knowledge Base

As detailed in Section 2, the tabular data contains categorical and discrete information for start-up/investor companies and the founders. Converting this data into triples required three different processes for (i) company data, (ii) founder data, and (iii) text descriptions. The knowledge base class written for this project contains method that allow for adding triples, merging KBs and adding indirect triples which are entities that relate to the company through another entity. The latter feature useful when enriching founder/investor data. A representative schema for the knowledge base/graph structure can be found in Figure 2.

3.1.1 Company Data

Extracting relations from the start-up/investor data sets is an intuitive process. Each row corresponds to a specific company, and each column represents an attribute of the company. Considering this, the [head-link-tail] triple generated from a specific row, column takes the general form [company name, column name, data stored]. More specifically this can be [Apple, founded_on, 1976]. The function written for this extraction returns a knowledge base which contains all triples and unique entities.

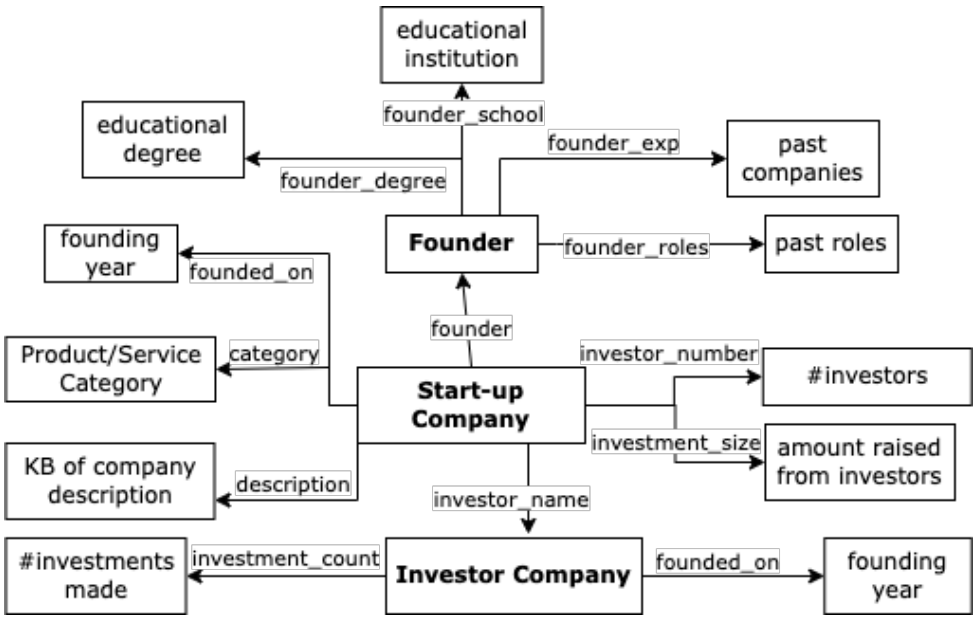


Figure 2: Knowledge Base/Graph Schema

3.1.2 Founder Data

Similar to the company data sets, the founder data can be converted into a knowledge base by relating the founders to the data stored in the columns by the column name. The triple generated from this generally takes the form [founder name, column name, data]. Since the founder data set is separate from the company data set the triple generated from the founder data set is not connected to the company itself. To resolve this an additional step was added to the algorithm, that generates a triple with the general form [company name, founder, founder name] after the generation of the founder triplet. This step essentially indirectly connects the information pertaining to the founder with the company. The knowledge base returned by this function contains the triples relating to the founders, and triples relating the founders to the companies.

Table 2: Semantic Triples

head	link	tail	text
Aledia	product produced	light-emitting diode	Aledia develops and manufactures innovative light-emitting diodes based on a unique 3D architecture using gallium-nitride on-silicon microwires...enabling the production of LED chips at 25 percent of the cost of traditional planar LED chips...
planar LED	subclass of	light-emitting diode	Same as above
Airtable	instance of	cloud-based software	Airtable is a cloud-based software company that offers an easy-to-use online platform for creating and sharing
Airtable	use	relational database	Same as above

3.1.3 Company Descriptions

To extract information from the descriptive text in a format suitable to append to the knowledge base. Natural language texts like this can be expressed in a computer-processable form by extracting [subject, predicate, object] triples. Considering this a two-stage process for embedding the company description to the knowledge base was designed. First, a semantic knowledge base consisting of triples with the general format defined above will be generated for each company description. To this end we will use

and end-to-end model called REBEL[3], capable of extracting more than 200 different relation types. It is a seq2seq model pretrained on BART[4],a denoising autoencoder.Sample results for triples generated from company description excerpt are in Table 2.

Although the information extracted are parallel to those in the company/founder data sets, these triples are not redundant. The critical difference is that this information is derived from how the company describes itself, which captures the characteristics and culture of the company/founders. Adding these triples will result in companies with similar company description triples to be more densely connected. After the triples are generated for each description, the triples are checked to see if they contain the company name. If not, the triple is indirectly connected to the company by generating a triple of the form [company name, description, head of unconnected triple] similar to section 3.1.2. Multiple functions are written to implement this methodology. The main ones involve performing relation extraction + creating KBs for each description, and merging the description KBs with the global one.

3.2 Knowledge Graph

After storing our tabular data in the knowledge base format, generating the graph is a straightforward process. The entities are the nodes and the relations are the named-edge list for our graph. This project uses the PyVis[2] library to generate the graph, however any common graph creation, manipulation and visualisation library (e.g. NetworkX) is also suitable for this purpose. The knowledge graph generated from the successful set of data, excluding the description triples is shown in Figure 3. As the entity size grows significantly with additional attributes, the class to generate and visualise the graphs allow for the user to specify which types of links should be visualised. These graphs on their own are very powerful tools in understanding the inter-connectedness of the data.

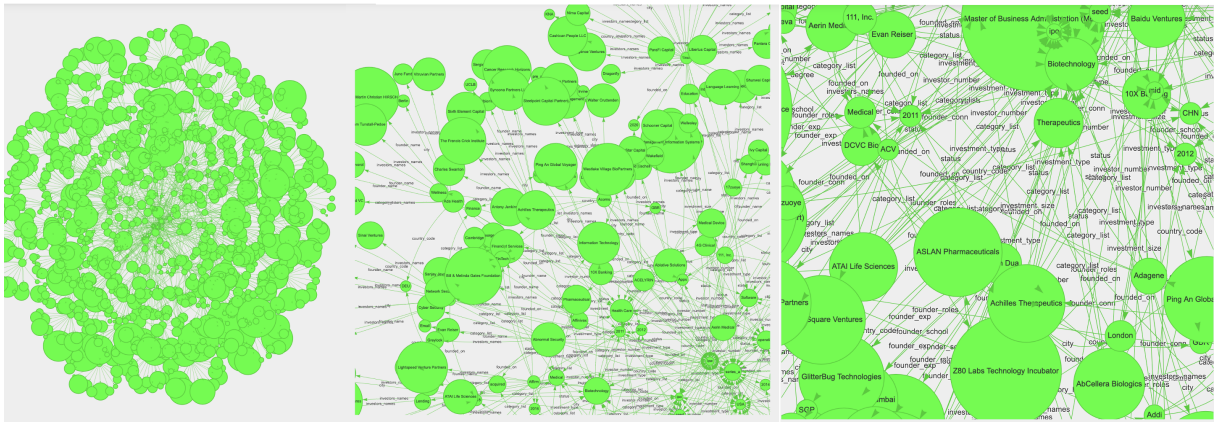


Figure 3: Knowledge Graph

3.3 Machine Learning on Knowledge Graphs

Knowledge graph embeddings (KGEs) are low-dimensional representations of the entities and relations in a knowledge graph. They provide a generalizable context about the overall KG that can be used to infer relations. It is essential in making knowledge graph compatible with machine learning models. Every model has a score function that measures the distance between the embeddings, quantifying the plausibility of a given triple. These scores are used to generate predictions. For this project we use the CompGCN[5], an embedding model a novel Graph Convolutional framework which jointly embeds both nodes and relations in a relational graph.

Once the embeddings are generated, they can be used to generate a variety of predictions. The most straightforward prediction task is given two items of a triplet, predicting the remaining. Intuitively this task can be interpreted as a search engine function, as it is extremely flexible and allows for a variety of queries. For example, given our data we can query the triple [?, investors_names, Tencent] which ideally will return the most likely companies Tencent should invest in. Alternatively we can query which investors will invest in the company. We can also query the investment amount a company will receive by querying [Company Name, investment_size, ?]. Another way of querying is to check if a triple is likely to exist, for example the investment size of a company can also be checked by comparing the relative scores of the

triples [Company Name, investment_size, low],[Company Name, investment_size, mid],[Company Name, investment_size, high].These and many more queries can be written and interpreted with the KGEs.

4 Implementation & Results

4.1 Implementation

The implementation of the methodologies detailed above were used to explore the predictive potential of learning over the knowledge graph by using a sample of 1000 successful and failed companies. The KB, KG and KGE of the successful and failed companies are separated as they will exhibit different structures, but the difference in their graph structure can uncover insight into what makes a start-up successful. This is left for future work. For this implementation The training, testing and evaluation for the embedding generation is implemented using the PyKeen Python package[1], and run on Google Colab. The model trained on the GPU with 1000 epochs, batch size of 512 and default parameters. Hyper-parameter tuning is left for future work.

4.2 Results

After training the model and generating the scores, a variety of questions were posed. Firstly the triple [?, investors_names, Tencent] was queried. Every possible triple is generated and scored by replacing the missing head value with each entity in our knowledge graph. The triples with the highest scores are returned as the most plausible values for the missing head. With this specific triple query we pose the question: which entity in our knowledge graph is most likely to be connected to Tencent by the investors_names link? Figure 4 shows the results generated. The top 3 results (Blockstream, Enflame

	head_id	head_label	score
888	888	Blockstream	7.587943
1411	1411	Enflame	7.278859
3065	3065	Satellogic	7.043642
1865	1865	Imply	4.494715
3660	3660	Wealthsimple	3.810984

(a) Query 1

	tail_id	tail_label	score
2130	2130	Khosla Ventures	4.071193
3388	3388	Tencent	3.655962
1906	1906	InterVest Co.	3.574475
2752	2752	Picus Capital	3.552078
1627	1627	Ge Ventures	3.469034

(b) Query 2

Figure 4: Query 1 Results

and Satellogic) are companies Tencent invested in which validates the accuracy of the CompGCN model. Wealthsimple and Imply are not invested in by Tencent, but since they are scored comparatively it indicates the model identified them as potential investees. Comparing the attributes of the latter companies with the former yields interesting results. The invested companies contain software and data analytics in their category list which matches with Imply's attributes. Interestingly Blockstream and Imply have a common investor: Khosla Venture. This might indicate a potential similarity between the two investor companies. How Wealthsimple relates to the rest of the companies is less apparent. Investigating further by querying ['Wealthsimple', investors_names, ?], i.e. which investors are the most plausible for Wealthsimple yielded an interesting result. The output for this query, shown in Figure 4(b), expectedly has Tencent, but most critically Khosla Ventures is the top score for the investors. These results can be interpreted in a variety of ways. Khosla Ventures can take this result as a recommendation to invest in Wealthsimple. It can also be a strong indicator for an affinity between Khosla Ventures and Tencent.

5 Discussion & Future Directions

Project Weave builds a framework for applying machine learning techniques and architectures to multi-relational data. Modules and scripts to 'weave'/ store the data and its inter-relations within a knowledge base, to construct and visualise a knowledge graph, and to generate knowledge graph embeddings –utilising state of the art graph convolutional network models are provided in the project GitHub.

On their own, knowledge graphs generated for company data proved to be a powerful tool to understand the inter-connectedness of all the elements. The true power of the KGs are showcased in the results section, which scratched the surface of the inductive/predictive capabilities of learning over them. In the results section, the queries were posed around investor/investee relationships. Even with training over a small subset of a 1000 companies, the results generated prove that this framework can easily be used to match investor profiles, generate investment recommendations for investors or even provide insight to companies looking to receive investments from certain investors. As long as the data is available, and the query can be posed in the triple form this framework is able to answer any question relating to the data. Extending on this, a search engine can be built on top of this framework, which can extract a query triple from a natural language question, run the model and return the answer.

Alternatively, the global structure of different graphs (e.g. industry-specific graphs, successful vs. failed start-up graphs) can be compared to drive important insights on characteristic attributes and relations differentiating the graph types. A potential application is driving insights on risk-aversity and other types of behaviours of investors when investing in different industries. Additionally, a comparison of the plausibility of an individual knowledge graph for a company belonging to set of global knowledge graphs can classify companies. An example might be checking if the company KG matches a successful or failed company KG to assess its success potential.

To conclude, this framework is the building block for an extremely flexible, accurate and powerful algorithm that has the potential to generate insights, classifications, and predictions. The accuracy of the model and the variety of the questions asked can easily be improved and extended by adding more data, hyperparameter tuning etc. More experimentation, trying different embedding methods and implementing graph ML techniques used in different industries will most definitely spark ideas for more applications and extend the potentials of this project.

References

[1] Mehdi Ali, Max Berrendorf, Charles Tapley Hoyt, Laurent Vermue, Sahand Sharifzadeh, Volker Tresp, and Jens Lehmann. PyKEEN 1.0: A Python Library for Training and Evaluating Knowledge Graph Embeddings. *Journal of Machine Learning Research*, 22(82):1–6, 2021.

[2] West Health. Pyvis– interactive network visualizations¶.

[3] Pere-Lluís Huguet Cabot and Roberto Navigli. REBEL: Relation extraction by end-to-end language generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2370–2381, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[4] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online, July 2020. Association for Computational Linguistics.

[5] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha P. Talukdar. Composition-based multi-relational graph convolutional networks. *CoRR*, abs/1911.03082, 2019.

[6] Ruiyun Rayna Xu, Hailiang Chen, and J Leon Zhao. Sociolink: Leveraging relational information in knowledge graphs for startup recommendations. *Journal of Management Information Systems forthcoming*, 2022.

[7] Jihong Yan, Chengyu Wang, Wenliang Cheng, Ming Gao, and Aoying Zhou. A retrospective of knowledge graphs. *Frontiers of Computer Science*, 12, 09 2016.