Project Weave: Startup Knowledge Graph

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

Abstract

The venture capital (VC) firms and startups generate significant amounts of data. Ventech, a proprietary technology built at Vela Partners, collects data sets relevant to the startups, investors, and founders, and interprets them to drive robust investment decisions. The majority of models built so far uses Euclidean data which does not capture the relationships among entities. Project Weave builds a pipeline to convert tabular market data into a knowledge base (KB), a database that captures the relations among entities, and a knowledge graph (KG) that visualizes and connects entity relationships in a non-Euclidean space. The KB and KG essentially weave the relationships among entities within the data sets while keeping all the context. Additionally, Project Weave explores counter-intuitive relationships by applying novel state-of-the-art Graph Machine Learning (ML) models, which provide future directions for this project.

Keywords: Knowledge Base; Knowledge Graph;Graph Machine Learning; Ventech; Startups

1 Motivation & Overview

The VC industry is increasingly relying on artificial intelligence (AI) enabled models that utilize the vast amount of data available on markets, companies and other components of the venture ecosystem to make smarter and more robust investment decisions. The insights generated by novel AI-enabled and data-driven models can be leveraged to predict variables relevant to the VC market such as a startup's success or growth potential, a founder's likelihood of success, and the behavior of a VC. Significant amounts of market data are available to fuel these models by leveraging various signals such as characteristics of startups, founders, employees and VCs. The existing ML models often focus on a subset of this data and do not utilize relations among entities. For instance, the number of investments that an investor made (investor count) has been used as a feature in various Ventech's investment decision-based ML models at Vela Partners but these models have been rigid and not flexible. Each feature has its predictive powers and more insights can be extracted. For example, by providing investor count as input to a relationship graph, we can get a list of the most successful investors who invests at a certain frequency (e.g. high conviction or spray and pray).

Recent developments in ML allowed well-known models and techniques to be extended to non-Euclidean spaces such as graphs. A KG is a multi-relational graph comprising entities and relations among entities[6]. These nodes and edges are stored in a KB in the form of triples specifying the entities and their relationships. By building a KB out of tabular data, and constructing the KG, the relationships among the entities can be utilized when performing machine learning tasks. The insights driven by ML models using a KG may generate more powerful insights than interrogating tabular data.

Project Weave provides a pipeline to build a KB and generates a KG using tabular data and entity relationships. It also explores learning over the KG to perform inductive and predictive graph ML (GML) tasks. The following sections detail the data sets, methodology for the KB and the KG pipelines as well as the experimental GML algorithms.

2 Data Preparation & Feature Engineering

The KB created for this project uses a subset of Moneyball 2.0 database, a proprietary database of Vela Partners. It contains enriched founder profiles through the use of web scraping methods and investor profiles. Most of the columns in the data sets are used. Only a couple of them were dropped due to redundancy or not being a good fit for the scope of this project. The raw data used for this project and its columns are provided in Section 2.1. Additionally, three features were built to convert certain variables from continuous to discrete so that the graph can be more readable and intuitive. The features will be further explained in Section 2.2. Finally, the Moneyball 2.0 data set has a text column that stores the company descriptions. Each description was converted into a knowledge graph using a natural language processing (NLP) algorithm called Named Entity Extraction. The KGs generated for each description were then added to the global KG. Since this process entails building a KG and not data preparation or processing, the details are provided in Section 3.

2.1 Data Sets

Moneyball 2.0 database This database has two data sets containing entities and relationships of 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 the company
- org_name: Name of the company
- status: Specifies if the company is acquired, operating, IPO or closed
- founded_on: The date that the company was founded
- category_group_list: A list of categories that the company operates in
- city: The city that the company was listed
- country_code: The country code of where the company was listed
- long_description: Long description of the company's business
- founder_linkedin_url: Linkedin profile links of the company founders
- investment_type: Specifies if the funding round is angel, pre-seed, seed or series a.
- raised_amount_usd: The total investment amount raised in US dollars.
- investor_count: Number of investors.
- investor_name: Unique key and name of the company.
- investor_uuids: Unique key of the company.

Founder Linkedin Data Set and Web Scraping The enriched founder data set contains information extracted from the Linkedin profiles of the company founders. Since the extraction was not recent, the web scraping script was used to update the data whenever needed. The columns are as follows:

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

Investor Profiles Data Set The investor data set contains detailed information about the investors that invested in the companies in the Moneyball 2.0 database. The columns are as follows:

- investor_uuid: Unique key of the company
- investor_name: Name of the founder
- investor_types: The investor types such as incubator, venture capital and family office
- investment_count: Number of investments made by the investor.
- founded_on: The date that the investor 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 Engineering

As mentioned in the prior sections, a KB contains information in the form of triplets:{entity_1, relation, entity_2}. In order to make the KG useful, intuitive and readable, these entities need to be discrete and descriptive values and variables. Most of the columns in the data sets detailed above in Section 2.1 are categorical and require no further processing and wrangling apart from filling and dropping missing values. There were only three continuous variables that were discretized to yield categorical features: (i) raised_amount_usd, (ii) investor_count and (iii) investment_count. Normalization was applied to raw data in order to interpret and visualize the underlying distribution. The relevant statistics of the variables are shared in Table 1. Mean normalization was used and the formula is provided below:

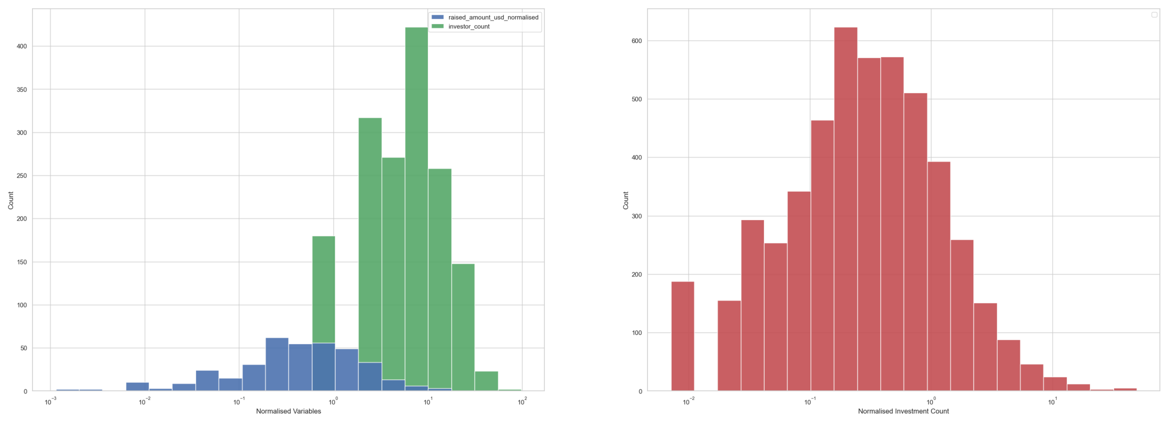
$$\tilde{x} = \frac{x - \mu}{\sigma} \tag{1}$$

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

2.2.1 Investment Size

This feature is a variable to categorize the amount of investment raised by the company. As detailed above, the Moneyball 2.0 database provides the amount invested in a company per investment type. The purpose of this feature is to categorize companies according to the total amount of investment they received. The raised_amount_usd column was grouped by org_uuid. Figure1(a) shows the histogram for the distribution of the total amount of investment received per company. The investment amount distribution was clustered around the lower and higher ends of the range, and most of the data is skewed toward the lower end. The x-axis has a logarithmic scale to account for this distribution and provide better visualization. The analysis concludes that three categories can effectively capture the attributes of this feature. For that reason, the normalized 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 that the number of investments received by the company. Similar to the investment size, this variable was separated according to the investment type and it was grouped to get the aggregated values per company. The distribution is clustered around the extremes. 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 provided in normalized and raw formats below:

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

2.2.3 Investment Count

This feature specifies the number of investments made by the investor. Figure1(b) shows the histogram for this variable on a logarithmic scale and an analysis of the distribution concluded four suitable categories. The need for an additional category was required because the distribution range for this continuous variable is higher compared to the other two variables described earlier. The boundaries for the four categories are provided in normalized and raw formats below:

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

3 Methodology

KGs consist of nodes and edges which specify entities and relations respectively. The entities refer to objects, organizations, locations and people, e.g. car, United Nations, London, Ada Lovelace. The edges represent the semantic relation among two entities. A triple (also defined as [head, link, tail]) is the smallest connected element in a KG, representing the relationship between two entities. KGs structurally represent all triples contained in a set of entities and relations. A KB is an intelligent database to manage, store and retrieve structured and unstructured data [7]. For this project, the 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 relationships. The entities are all the unique entries extracted from the data sets. Relations contain the triples in an array of dictionaries with the keys as head, link and tail. The KG is constructed from this KB to interpret the entities as nodes, and relations as the edge list. This section details the methodology for constructing the KB (3.1) and generating the KG (3.2).

3.1 Knowledge Base

As detailed in Section 2, the tabular data contains categorical and discrete information for startups, investors and founders. Converting this data into triples required three different processes for the (i) company data, (ii) founder data, and (iii) text descriptions. The KB class written for this project contains methods 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 is useful when enriching founder and investor data sets. A representative schema for the KB and KG structure can be found in Figure 2.

3.1.1 Company Data

Extracting relations from the startup and 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 and column takes the general form of [company name, column name, data stored]. More specifically this can be [Apple, founded_on, 1976]. The function written for this extraction returns a KB which contains all triples and unique entities.

3.1.2 Founder Data

Similar to the company data sets, the founder data can be converted into a KB 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 issue, an additional step was added to the algorithm. This algorithm generates a triple with the general form [company name, founder, founder name] after the founder triplet is created. This step indirectly

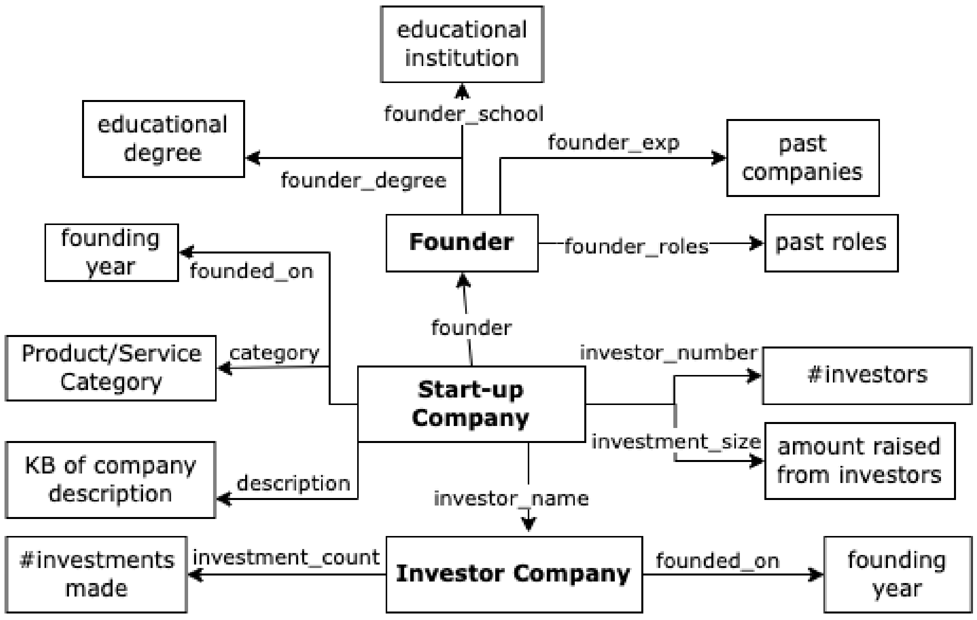


Figure 2: Knowledge Base/Graph Schema

connects the founder and the company. The knowledge base returned by this function contains the triples related to the founders, and triples connecting 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

In order to make company descriptions suitable for the KB structure, further work had to be completed. Natural language texts can be expressed in a computer-processable form by extracting [subject, predicate, object] triples. A two-stage process was formed in order to embed company descriptions to the KB. First, a semantic KB consisting of triples with the general format defined above was generated for each company description. For that reason, an end-to-end model called REBEL[3], which is capable of extracting more than 200 different relation types, was used. It is a seq2seq model pre-trained on BART[4], a denoising auto-encoder. Sample results for triples generated from the company description data set are shown in Table 2.

Although the information extracted is parallel to those in the company and founder data sets, these triples are not necessarily redundant. The important nuance is that this information is derived from how the company describes itself, which captures the characteristics and culture of the company and the founders. Adding these triples resulted in companies with similar company description triples being 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 were written to implement this methodology. The main ones involve performing relation extraction and 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 KB format, generating the graph was 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, but any common graph creation, manipulation and visualization library (e.g. NetworkX) can also be suitable for this purpose. The KG generated from the successful data set, excluding the description triples, is shown in Figure 3. The entity size grows significantly with additional attributes and gets more complex. For that reason, a function was created to generate and visualize the graph by enabling a user to specify the type of relationship links. These graphs on their own are immensely powerful tools to understand the inter-connectedness of the data sets.

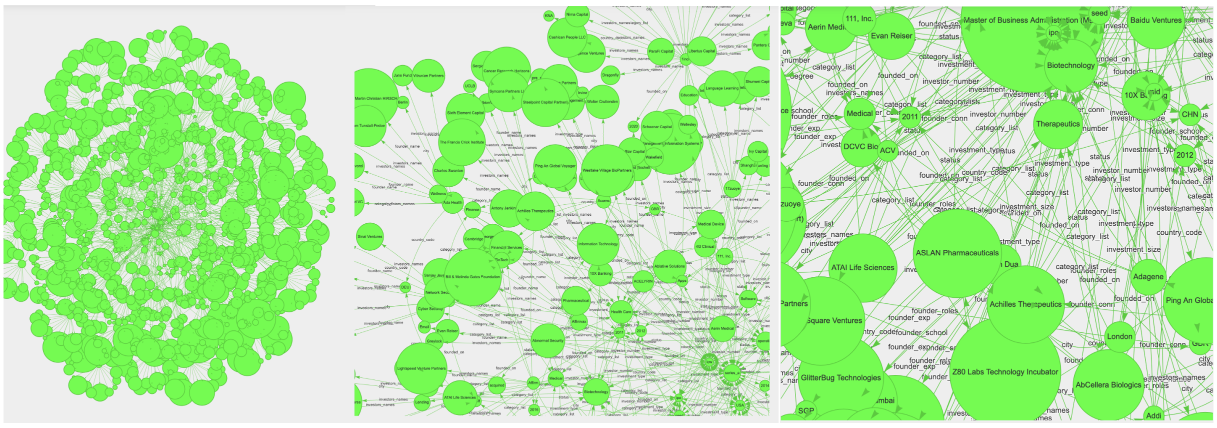


Figure 3: Knowledge Graph

3.3 Applying Machine Learning to 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 KG compatible with machine learning models. Every model has a score function that measures the distance among the embeddings, which quantifies the plausibility of a given triple. These scores are used to generate predictions. For this project, we used the CompGCN[5], an embedding model a novel Graph Convolutional Networks 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 giving two items of a triplet and predicting the remaining. Intuitively this task can be interpreted as a search engine function, as it is extremely flexible and it allows for a variety of queries. For example, given our data, we can query the triple [?, investors_names, Tencent] which ideally would return the most likely companies Tencent should invest in. Alternatively, we can query which investors may invest in a 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 total investment that a company may receive 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 using the KGEs.

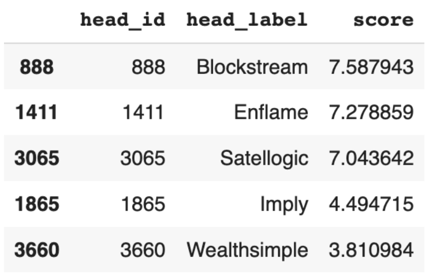
4 Implementation & Results

4.1 Implementation

The implementation of the methodologies detailed above was 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 would exhibit structural differences. The difference in their graph structure may uncover insight into which startups may be successful. This is left for future work. For the purpose of our implementation, the training, testing and evaluation of the embedding generation is implemented using the PyKeen Python package[1], and run on Google Colab. The model was trained on a GPU with 1000 epochs and a batch size of 512 using default parameters. Further optimization tasks such as hyper-parameter tuning are left for future work.

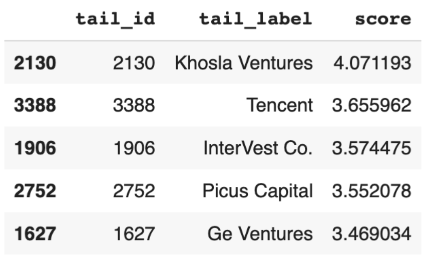
4.2 Results

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, a question was posed: 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.



	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

The top 3 results (Blockstream, Enflame and Satellogic) are companies Tencent invested in which validates the accuracy of the CompGCN model. Wealthsimple and Imply were not invested by Tencent. Since they scored comparatively high, this indicates that the model identified them as investment prospects. Comparing the attributes yield interesting results. The invested companies contain software and data analytics in their category list which matches Imply's attributes. Interestingly, Blockstream and Imply have a common investor: Khosla Ventures. This might indicate a similarity between the two investors. On the other hand, 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. More importantly, Khosla Ventures is retrieved at the top. This 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 of an affinity between Khosla Ventures and Tencent.

5 Discussion & Future Directions

Project Weave builds a framework to apply ML techniques to multi-relational data sets. Modules and scripts to 'weave' and store the data, build inter-relations of the KB, construct and visualize the KG, generate the knowledge graph embeddings and utilize state-of-the-art graph convolutional network models are provided in the project's GitHub repository.

Independently without the use of other components of this project, company data KG proved to be a powerful tool to understand the inter-connectedness of all the elements. The true power of the KGs was showcased in the Results section, which scratched the surface of the inductive and predictive capabilities of learning over them. In the Results section, the queries were posed around investor and company relationships. Even with training over a small subset of 1000 companies, the results proved that this framework can easily be used to generate investment recommendations for investors or provide insight to companies which investors they should reach out to. 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 set. We believe that a search engine can be built on top of this framework, which can extract a query triple from a natural language question, run this model and return an answer.

Alternatively, the global structure of different graphs (e.g. industry-specific graphs, successful vs. failed startup graphs) can be compared to drive important insights on characteristic attributes and relations. A potential application could be to generate insights to reduce risk for investors when investing in different industries. For instance, one can answer which founder characteristics or co-investors can be attractive while investing in a certain industry. Additionally, companies can be classified by fitting their individual KG to a global KG. For example, one might be check if the company KG matches a successful or failed company KG to assess its success potential.

In conclusion, 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 posed here can easily be improved and extended with activities such as adding more data and tuning the hyper-parameters. The suggestion is to experiment more, try different embedding methods and implement other graph ML techniques used in different industries. This direction most definitely may spark new ideas for more applications and extend the potential 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.