CrunchQA: A Synthetic Dataset for Question Answering over Crunchbase Knowledge Graph

Lifan Yu
NYU Shanghai
Shanghai, China
ly1164@nyu.edu

Nadya Abdel Madjid
NYU Abu Dhabi
Abu Dhabi, UAE
na3112@nyu.edu

Djellel Difallah
NYU Abu Dhabi
Abu Dhabi, UAE
dd115@nyu.edu

***Abstract*—The digital transformation in the finance and enterprise sector has been driven by the advances made in big data and artificial intelligence technologies. For instance, data integration enables businesses to make better decisions by consolidating and mining heterogeneous data repositories. In particular, knowledge graphs (KGs) are used to facilitate the integration of disparate data sources and can be utilized to answer complex queries. This work proposes a new dataset for question-answering on knowledge graphs (KGQA) to reflect the challenges we identified in real-world applications which are not covered by existing benchmarks, namely, multi-hop constraints, numeric and literal embeddings, ranking, reification, and hyper-relations. To build the dataset, we create a new Knowledge Graph from the Crunchbase database using a lightweight schema to support high-quality entity embeddings in large graphs. Next, we create a Question Answering dataset based on natural language question generation using predefined multiple-hop templates and paraphrasing. Finally, we conduct extensive experiments with state-of-the-art KGQA models and compare their performance on CrunchQA. The results show that the existing models do not perform well, for example, on multi-hop constrained queries. Hence, CrunchQA can be used as a challenging benchmark dataset for future KGQA reasoning models. The dataset and scripts are available on the project repository.** [1]

***Index Terms*—Knowledge Graph, Graph Embedding, Question Answering, Enterprise KB**

I. Introduction

A Knowledge Graph is a data organization model that uses graph vertices and edges to represent real-world entities and their relationships. This representation can facilitate data integration from various datasets by leveraging common and machine-readable ontologies. Beyond data integration, KGs can be used as a source of factual information (e.g., in Google Knowledge Panel, Wikipedia Infoboxes), to infer missing (latent) facts [1], [2], and to answer complex multi-hop queries. Motivated by their flexibility and the opportunities they offer, KGs have been advocated and embraced within several industries [3]. End-users can interact with the a KG using structured language (such as SPARQL) or natural language queries. The latter is a task commonly referred to as Question Answering over Knowledge Graphs (KGQA) [4].

In this paper, we start by reviewing existing KGQA datasets and their limitations, highlighting missing features critical to real-world applications (Section II). To fill this gap, we take the enterprise and finance domain as a use-case and create a new dataset that reflects the challenges we identified. This choice is motivated by the sundry of applications leveraging KGs [5], [6], for example, fraud detection [7], stock prediction [8], information extraction and linking [9], content wikification [10], and automated KG construction [11].

Similar to our effort, Zehra et al. [12] constructed a KG from structured/unstructured data and implemented a system for resolution of intelligent financial queries on top of the KG. The system extracts target entities from a query, converts it into a graph querying statement and extracts the results from the graph. Zhiyu et al. [13] created a financial QA dataset through scanning the financial reports and composing the questions, which require first retrieving the relevant information and then performing numerical reasoning.

In this work, we create a KGQA dataset based on a knowledge graph we derive from the Crunchbase, which contains information about companies ranging from early-stage startups to large companies. The questions are automatically generated from a set of carefully designed templates and paraphrases that capture several complexity levels. In summary, we make the following key contributions:

1) We build a Knowledge Graph from the Crunchbase database using a lightweight schema. (Section III-A)
2) We build a Question Answering dataset generator using multi-hop templates and question paraphrasing. Unlike other datasets, the templates have an ordinal facet relevant for fintech and enterprise related questions. (Section III-C)
3) We propose a simple clustering-based method to add numerical literals support to existing embedding-based methods. (Section IV)
4) We conduct extensive experiments to evaluate state-of-the-art KGQA models on CrunchQA, their shortcomings, and highlight future area of research. (Section V)

II. Background

A knowledge graph $\mathcal{G}$ is a directed labeled graph where the nodes represent entities $\mathcal{V}$ and the edges connecting these nodes represent relations $\mathcal{R}$ between the entities. Mathematically, we define the knowledge graph $\mathcal{G}$ as a set of triplets $(h, r, t) \in \mathcal{E} \subseteq \mathcal{V} \times \mathcal{R} \times \mathcal{V}$ with each triplet representing a directed *relation* r between a head entity h and a tail entity t.

[1] https://github.com/colab-nyuad/CrunchQA

978-1-6654-8045-1/22/$31.00 ©2022 IEEE

KGQA. The task of question answering over a knowledge graph $\mathcal{G}$ involves taking as input a question q (in its textual form) and a head-entity h (main subject of the question). The goal is producing the correct set of answer entities $\mathcal{A}^q \subset \mathcal{V}$. An effective KGQA system must be capable of multi-hop reasoning and handling questions in the presence of missing links in the KG. This scenario can be simulated by truncating edges from the KG.

Datasets. To build a KGQA model, a QA dataset $\mathcal{D}$ is provided with a set of tuples (q, h, a), where $a \in \mathcal{A}^q$. Each tuple may have an *inferential chain*, that is, the subset of relations connecting the head h entity and the answer entity a, in the context of the question q. We summarize the properties of the three common datasets used to evaluate KGQA methods:

1) METAQA [14] is a large-scale multi-hop QA dataset containing over 400k movie questions. It includes 1-hop, 2-hop and 3-hop questions with answers, question templates, and the underlying knowledge graph. The questions do not contain constraints. Some answers return *year* as a literal. The underlying graph contains 43K entities and only 10 relation types.
2) WEBQUESTIONSSP (WQSP) [15] is a smaller QA dataset consisting of 4,737 natural language questions that are answerable through 1-hop and 2-hop reasoning over a subset of Freebase KG. It contains roughly 2M entities and 550 relation types. Some questions have constraints including gender, city or date. Some questions have *year* as answers.
3) COMPLEXWEBQUESTIONS (CWQ) [16] is built on top of WebQuestionsSP dataset with additional constraints to the existing questions. The dataset contains four types of constraints: composition (45%), conjunction (45%), comparative (5%), and superlative (5%) and requires up to 3-hops and 4-hops of reasoning over Freebase KG to reach the answers.

Performance of KGQA Models. Table I shows the performance of existing methods on these datasets. We observe that the methods based on graph embeddings and path reasoning, such as TransferNet, solve MetaQA in multi-hop questions. A manual inspection reveals that the remaining gap in MetaQA's 1-hop questions is mainly due to errors in the dataset. The same methods have improved the results in WQSP and CWQ; however, these datasets are still challenging to solve fully.

TABLE I
HITS@1 RESULTS ON KGQA DATASETS. THE METHODS ARE ORDERED BY ASCENDING PUBLICATION YEAR. (RESULTS FROM [17])

Model	MetaQA			WQSP	CWQ
	1-hop	2-hop	3-hop		
KVMemNN [18]	95.8	25.1	10.1	46.7	21.1
GraftNet [19]	97.0	94.8	77.7	66.4	32.8
PullNet [20]	97.0	99.9	91.4	68.1	47.2
VRN [14]	**97.5**	89.9	62.5	–	–
SRN [21]	97.0	95.1	75.2	–	–
ReifKB [22]	96.2	81.1	72.3	52.7	–
EmbedKGQA [23]	**97.5**	98.8	94.8	66.6	–
HyperKGQA [24]	97.0	99.4	88.5	**64.1**	–
TransferNet [17]	**97.5**	**100**	**100**	71.4	48.6

Nevertheless, none of these datasets contain literal, numeric, or date attributes; nor do they contain ranges and other ordinal-based queries, for example, *"How tall is the Eiffel tower?"*, *"List the 5 latest EdTech companies IPOs"*, *"Who were the presidents of the United States during the 1960s?"*.

In this work, we propose to fill this gap by developing CrunchQA. We follow the MetaQA approach, i.e., the questions are generated automatically from a predefined set of templates. Moreover, we add the challenges that WQSP and CWQ offer: having a large KG and constraints. Additionally, since all of the existing datasets have a unique form per question pattern, which poses the issue of overfitting, we tackle this issue by creating multiple template paraphrases.

III. CRUNCHQA DATASET

In this section, we describe the building blocks used by our method to construct a Knowledge Graph from the Crunchbase database dump[2] as well as the generation of QA templates and the QA dataset.

A. CrunchKG: Knowledge Graph Construction

The Crunchbase data dump comprises 17 relational tables[3] with primary and foreign keys linking tables together. In the following, we describe our approach to building CrunchKG, the underlying knowledge graph of our dataset.

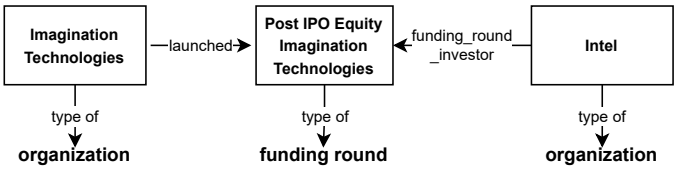

Fig. 1. Triples with main entities and identified relations.

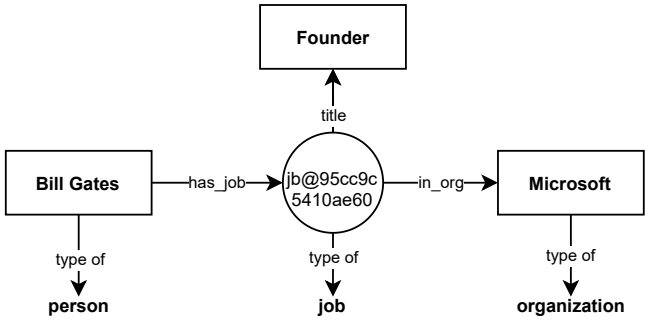

Fig. 2. Reification node JOB.

First, we create new entities for each main entity type, namely, ORGANIZATION, PERSON, FUND, EVENT, and FUNDING ROUND and identify the relations between them. Figure 1 shows an example where entities of types ORGANIZATION and FUNDING ROUND are connected by the relation *launched*, since the company "Imagination Technologies" launched a

[2]The database schema description is available at: https://data.crunchbase.com/docs

[3]The database dump was obtained on December 2021.

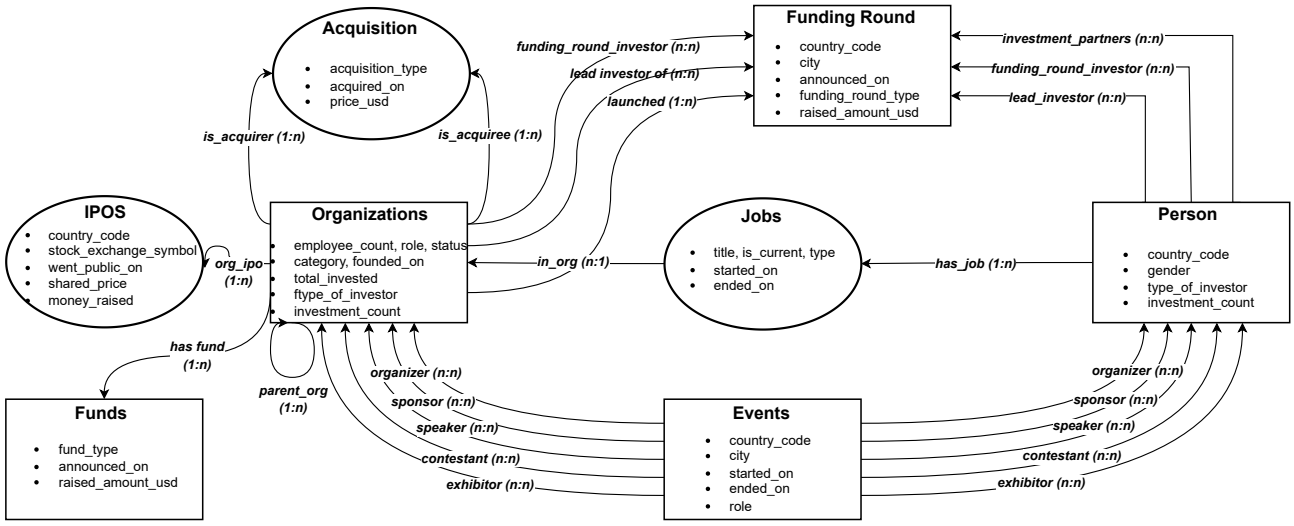

Fig. 3. Diagram of the constructed Knowledge Graph showing main entities, reification nodes and relations with cardinality. Elliptical nodes represent reification nodes.

funding round and by the relation *funding_round_investor* due to the Intel's investment in this funding round.

Next, we use reification nodes JOB, ACQUISITION and IPO to map the relationship between base entities. Each reification node instance has a unique ID and may contain additional information about the relationship between entities. For example, instances of JOB are reification nodes that link entities of types ORGANIZATION, PERSON and add additional information about the position a person occupies or occupied, e.g, start and end dates of employment, position title (refer to Figure 2 for more details).

Finally, we map additional triples extracted from the entities' respective tables and, at the same time, we exclude fields of various corrupted or insignificant metadata irrelevant to our task. Beyond entities, CrunchKG triples support the following attribute types:

- *Literal Attributes.* A textual form which describes attributes of the head entity. For example, a job title, or description.
- *Numeric Attributes.* reflect financial indicators such as invested amount, IPO price, raised amount of money during funding rounds, etc.
- *Date Attributes.* to reflect temporal information.

CrunchKG. The knowledge graph we constructed includes 3.2 million entities, 31 relations, and 17.6 million triples. Figure 3 shows the overall structure of our Knowledge graph, where rectangular nodes denote the main entities and elliptical nodes denote the reification nodes. A detailed description of all the entities and relation types is available on the dataset's website. Since the Crunchbase dataset is subject to licensing, we provide a script to process a dump and reconstruct the KG at a given timestamp.

B. Question Templates

To obtain an initial set of templates, we construct a supplementary bidirectional graph $\mathcal{G}_s$ with main class entities and reification nodes as vertices, and relations as edges. The graph $\mathcal{G}_s$ reflects the scheme of the Knowledge graph $\mathcal{G}$. At the next stage, we traverse the graph $\mathcal{G}_s$ from every vertex of main class entities and record all possible paths and vertices we can reach within 1 and 2 edges. In the context of templates, the result of this stage is the list of potential inferential chains I, where the source vertex points to the head entity and destination vertex to the answer entity. Table II shows examples of inferential chains generated by traversing the KG from the head entity "Google" of type ORGANIZATION within 1 and 2 edges.

TABLE II
TRAVERSAL EXAMPLES

Destination t	**Inferential Chain** I	**Question** q
1-hop traversal		
Funding round	org-funding_round_investor-funding round	In which funding rounds did Google invest?
Funding round	org-launched-funding round	Which funding rounds did Google launch?
2-hop traversal		
Organization	org-is acquirer-acquisition-is acquiree-org	What companies did Google acquire?

At the next stage, after obtaining all potential inferential chains, the redundant and unreasonable chains are filtered out through manual screening. Then, the remaining inferential chains are manually examined for the possibility of attaching constraints. Each constraint consists of a constraint inferential path, which can be up to 2-hops since some constraints involve reification nodes. To retrieve the set of possible constraints that can be attached, we apply a similar approach as in the case of retrieving inferential chains. For an inferential chain I^t, we traverse the graph G_s from each node entity in I^t, so that the head entity of the constraint is in the main inferential chain. This will give us a set of possible constraints that can be attached to the inferential chain I^t. After filtering, meaningful

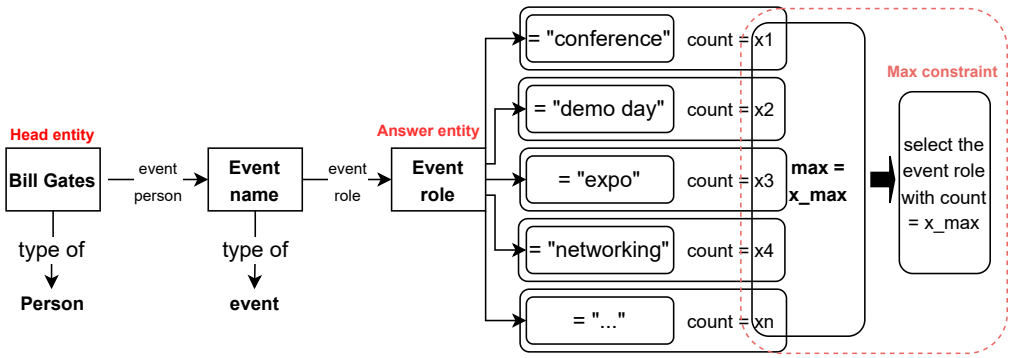

Fig. 4. Grouping, counting and selecting process for maximum constraint specified in the template for the question *"What types of events did Bill Gates mostly participate in?"*

combinations of constraints result in a set of different templates.

The types of constraints used in this paper are entity, temporal, maximum and numeric. These specific types and the format of constraints are based on our inspiration to cover the most informative insights a user can infer from the KG. After this stage, each template t is represented by an inferential chain I^t and the set of constraints C^t.

The templates are classified into 3 categories: **1-hop** (inferential chain of length 1 and at most 1 constraint), **2-hop** (inferential chain of length 2 and at most 1 constraint), and the challenging templates are classified as **Advanced**. We populate the advanced templates with the most popular business inquires we find on forums and Crunchbase FAQ.[4] For advanced templates, to ensure diversity in lexicalization of questions, each template has several paraphrases for a question query. When generating questions we randomly select which query form to use.

In the following, we provide the description and format of each constraint type.

1) Entity constraint: requires the tail entity of the constraint chain to be equal to a certain value, e.g., for queries asking about financial service companies in Abu Dhabi, the head entity is "financial services" and the constraint can be specified as city = 'Abu Dhabi'.

2) Temporal constraint: requires the date to be within a specified time range. A sufficient set of questions that we surfed require a time range, and not necessary an implicit timestamp. Some of the questions are aimed to see the dynamics attached to a specific period (e.g., pandemic), which can be reflected through a temporal constraint. For instance, from the beginning of COVID-19 can specified as {"after": 2019} in the template. The format of the constraint is: {"before":year}, {"after":year} or {"between": [year1, year2]}. If time range is not specified, we sample from the records without filtering by time.

3) Maximum constraint: is introduced to reflect key words as "top", "at most", "the highest", etc. Maximum is always computed within a group, e.g., if we want to know *"which software companies have the highest IPO share price"*, the constraint first specifies grouping by the category 'software' and then selects the maximum among share prices. Another setting the maximum constraint supports is first counting over edges and then selecting maximum, e.g., *"companies acquired by Meta mostly come from which industry"*. In this example, the number of companies that Meta acquired in each industry is counted and the industry with the highest count is selected. For this setting we need to specify three fields (i) *"count_group_by"*: the list of columns to group by, (ii) *"count_over"*: the column we count over while grouping and from which we will select the maximum, and (iii) *"max_group_by"*: the entity around which the question is centered. For example, in the query asking *"what types of events did Bill Gates mostly participate in?"*, we need to filter on 'person' and group by 'event_type'. While grouping, we count how many times Bill Gates participated in event type, which is equivalent to counting over the column 'events'. After grouping, we select the maximum from the count for Bill Gates. In the generic question templates, we choose the maximum of count for the central entity (person). Figure 4 visualizes the described grouping, counting, and selecting maximum.

4) Numeric constraint: reflects the key words "more than", "less than", "at least". This constraint implies counting over edges, e.g., for a question *"list companies with acquired more than 50 companies"*, the constraint first specifies grouping by organization acquisitions where it is an acquirer, then counts acquirees and selects a company based on a condition for a number of acquirees > 50. The numeric condition can be specified in the format ["", "> | = | <", number].

5) Multi-entity/relation: type is introduced to cover questions which can refer to multiple entities or relations, e.g., if we ask about investors, both companies and people can make investments, or if a question is about participating in an event without specifying a specific role, we should encounter all types of relations, i.e., sponsor, speaker, organizer, contestant and exhibitor.

To conclude, question-constraint parameterization allows the flexibility in creating templates and lets us decompose questions and present them as a combination of constraints. Figure 5 shows an example of such a template, which is a combination of three constraints: two entities and one numeric.

[4]https://www.crunchbase.com/featured

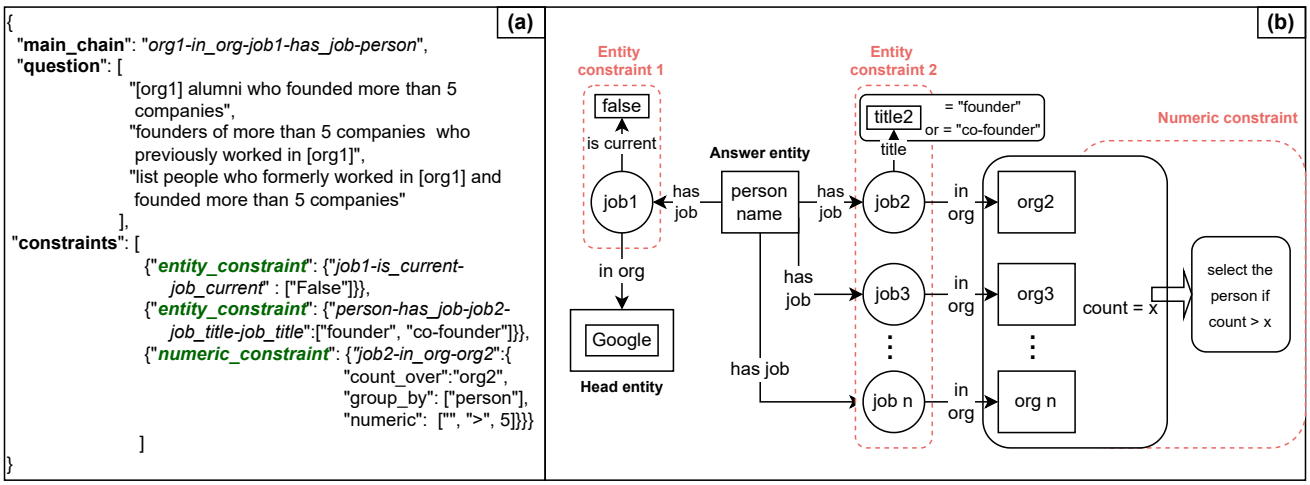

Fig. 5. Visualization of an advanced question template with 2-hop main chain, two entity constraints and one numeric constraint: (a) the template in json format, (b) shows how the constraints are applied

C. Generating Questions from Templates

Giving a set of templates T, we generate N questions per template, where N is a parameter. Algorithm 1 depicts the overall procedure of question generation from templates. Each constraint chain set is merged with the set formed by the main chain through inner join. Before sampling from the final set of records, we group by question to accumulate answers. After grouping, the questions with more than 20 answers are dropped to remove questions with unspecific answers. Finally, we sample N questions and improve the readability of the questions. In algorithm 1, *make_readable()* is a human-computation function to verify the lexicalization of the final questions [25]. While this operation can be executed using crowdsourcing, we were able to screen most of the anomalies manually. We further split the questions into three categories based on the data type of their answers: TEMPORAL questions return date information, NUMERIC questions return numbers. The rest of the questions are ENTITY questions, i.e., they return regular entities.

Algorithm 1 Generation of QA dataset

Require: $T, \mathcal{G}, N$
  $Q = []$ #list of questions
  for t in T **do**
    main_set = *extract*(t.main_inferential_chain)
    for c in C^t **do**
      sub_set = *extract*(c.inferential_chain)
      inner_join(main_set, sub_set)
      group_by_question(main_set)
    end for
    samples = *select_samples*(main_set, N)
    questions = *make_readable*(samples)
    append(Q, questions)
  end for
  return Q

D. CrunchQA: Question Answering Dataset

Table III shows some samples of the questions generated from different categories of templates (1,2-hop and advanced) and different constraints types. We highlight in bold the head entity (topic of the question) and keywords, which are mapped to a constraint in the template. For instance, in the question *"List people who formerly worked in Chaotic Moon Studios and founded more than 3 companies?"*, the main entity is the company where employees worked (Chaotic Moon Studios) and the keywords "more than three" indicate that there is a numerical comparison which links to a numerical constraint.

Table IV shows the statistics of the QA dataset with total 29,367 questions generated. We anticipate that the size of CrunchQA dataset is sufficient to train new effective KGQA systems and yet will be a challenging benchmark dataset for the QA systems research community not due to the limited size or unbalanced data but rather because of semantically and logically complex questions.

IV. BASELINE MODELS

We focus our evaluation on EmbedKGQA [23], an approach that combines graph embeddings and reasoning using graph traversal. We select this approach because it proposes a general framework that captures the main ideas of the state-of-the-art techniques while allowing for experimenting with different backend KG embedding models. The EmbedKGQA framework has two training stages: (i) train the knowledge graph embedding on the link prediction task; (ii) train the model for QA by leveraging the pretrained embeddings. The model aims to project the question into the space of KG's embedding. To learn the question representation, the model uses a pretrained sentence-transformer [26] followed by two linear layers. To learn KG embeddings, we use the PyKEEN [27] library. We conduct experiments with three KG embedding models setting.

TABLE III
QA TEMPLATES: OVERALL THE DATASET CONTAINS 243 TEMPLATES. NOTE, ADVANCED NO CONSTRAINT INDICATE MULTI-ENTITY/RELATION QUESTIONS; SINGLE-ENTITY ADVANCED QUESTIONS BELONG TO SEVERAL CONSTRAINTS, SO THE PERCENTAGES DO NOT SUM UP TO 100.

Type	Constraint category	Example Question	Percentage
1hop	no constraint	**Seed Round - Celtra** was launched in which country	11.5%
	entity constraint	list **conferences** that took place in **Abu Dhabi**	10.2%
	temporal constraint	which funding round did **Sweetch** launch in **July 2021**	21.8%
	max constraint	what types of event did **Meta most frequently** participate in	0.4%
2hop	no constraint	from which industries are the investors of **Series A - Healthera**	25.9%
	entity constraint	**Meta** invested in which **U.S.** companies	2.9%
advanced	no constraint	who was the lead investor of **Seed Round - QuotaPath**	4.5%
	entity constraint	list companies in the **Home Decor** sector that released their initial public offering on **sse**	13.5%
	temporal constraint	which Ad Targeting companies were founded **after 2019**	8.2%
	max constraint	which Rental Property company has **the highest** ipo share price	2%
	numeric constraint	list people who formerly worked in **Chaotic Moon Studios** and founded **more than 3** companies	2.5%

TABLE IV
CRUNCHQA: QA DATASET STATISTICS

	Train	**Valid**	**Test**
Entity	13485 (65.6%)	1847 (62.9%)	3792 (64.6%)
Temporal	6394 (31.1%)	969 (33%)	1878 (32%)
Numeric	679 (3.3%)	120 (4.1%)	203 (3.4%)
Total	20558 (70%)	2936 (10%)	5873 (20%)

1-hop	**2-hop**	**Advanced**	**Total questions**
10515 (35.8%)	6732 (22.9%)	12120 (41.3%)	29367

TABLE V
THE EVALUATION OF EMBEDKGQA WITH SEVERAL KG EMBEDDINGS MODELS. WE REPORT HITS@1 ON CRUNCHQA.

KG Model	Entity	Temporal	Numeric	Overall
TransE	17%	-	-	-
ComplEx	12%	-	-	-
DistMult	12.7%	-	-	-
ComplExLiteral	13.8%	-	-	-
DistMultGatedLiteral	10.6%	-	-	-
TransE (clustering)	**18.2%**	8.1%	**37.4%**	**15.6%**
ComplEx (clustering)	13.2%	8.8%	33.5%	12.4%
DistMult (clustering)	11.1%	**8.9%**	8.8%	10.2%

A. KG Embedding Models

This evaluation setting uses compositional and translational embedings models to answer entity type questions. Specifically, our experiments utilize TransE model [1], which uses a simple addition operation as a scoring function, ComplEx model [28], which represents entity vectors and relation matrices in the complex space and bi-linear product based model DistMult [29].

B. KG Embedding Models with Literals

Literals models incorporate literals information to compute entity embeddings. However, the computed entities embedding matrix does not contain embeddings for literals. Since the KGQA model can generate as the answer only vertices with embeddings, the model cannot answer temporal and numeric questions in this setting. In our experiments we use two literals models, i.e., ComplExLiteral and DistMultGatedLiteral [30].

C. KG Embedding Models with Clusters

To answer the temporal and numeric question, we propose as a simple baseline a clustering-based variant of the models. In this approach, dates and numeric values are clustered and treated as entities; thus the model can compute their embeddings. To achieve this, dates are clustered into *years*, omitting days and month information. We perform K-means clustering among the categories for numeric values as 'share_price' or 'investment_count'.

V. EVALUATION

Table V shows the quantitative results for the described setups where we report Hits@1. Among vanilla settings, TransE model achieves the best performance as in the case of clustered data for entity and numeric questions and as a result in the overall performance. For ComplEx model, link prediction Hits@1 after incorporating literals information increased almost two times, which results in rise of 1.86% for QA. On contrary, for DistMult model after incorporating the literals information the performance for link prediction degraded, which also led to decrease in the QA performance, i.e., by 2.1% in comparison with the entity setting. All three clustering models demonstrate similar low results for temporal questions (average around 8.5%), however, for numeric questions, TransE and ComplEx models are tangibly superior to DistMult.

VI. CONCLUSION AND FUTURE WORK

In this work, we created CruhchQA, a new dataset for the question-answering task. The dataset is based on the Crunchbase database and template-based question generation. The proposed dataset can be used as a benchmark for future KGQA models and promotes designing new question-answering models that can handle questions with multi-hop constraints, numeric and temporal information, ranking, reification, and hyper-relations. As a future work, we will pursue the following directions (i) developing new path-based KGQA models that can handle ranking and selection of maximum, and (ii) adapting embedding-based KGQA models to literals.

REFERENCES

[1] A. Bordes, N. Usunier, A. Garcia-Durán, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data." Red Hook, NY, USA: Curran Associates Inc., 2013.

[2] M. Luggen, J. Audiffren, D. Difallah, and P. Cudré-Mauroux, "Wiki2prop: A multimodal approach for predicting wikidata properties from wikipedia," in *Proceedings of the Web Conference 2021*, 2021, pp. 2357–2366.

[3] J. Shinavier, K. Branson, W. Zhang, S. Dastgheib, Y. Gao, B. Arsintescu, F. Özcan, and E. Meij, "Panel: Knowledge graph industry applications," in *Companion Proceedings of The 2019 World Wide Web Conference*, 2019, pp. 676–676.

[4] L. Hirschman and R. J. Gaizauskas, "Natural language question answering: the view from here," *Nat. Lang. Eng.*, vol. 7, no. 4, pp. 275–300, 2001. [Online]. Available: https://doi.org/10.1017/S1351324901002807

[5] J. P. McCrae, P. Mohanty, S. Narayanan, B. Pereira, P. Buitelaar, S. Karmakar, and R. Sarkar, "Conversation concepts: Understanding topics and building taxonomies for financial services," *Information*, vol. 12, no. 4, 2021.

[6] Y. Wang, J. Zhao, F. Li, and M. Yu, "Construction of knowledge graph for internal control of financial enterprises," in *2020 IEEE 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, 2020, pp. 418–425.

[7] Q. Zhan and H. Yin, "A loan application fraud detection method based on knowledge graph and neural network," ser. ICIAI '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 111–115.

[8] Y. Liu, Q. Zeng, J. Ordieres Meré, H. Yang, and B. M. Tabak, "Anticipating stock market of the renowned companies: A knowledge graph approach," *Complex.*, vol. 2019, jan 2019.

[9] Z. Wang, X. Zhang, and Y. Hu, "A semantic path based approach to match subgraphs from large financial knowledge graph," in *Proceedings of the 2019 International Conference on Mathematics, Big Data Analysis and Simulation and Modelling (MBDASM 2019)*, 2019/10, pp. 86–92.

[10] M. Gerlach, M. Miller, R. Ho, K. Harlan, and D. Difallah, "Multilingual entity linking system for wikipedia with a machine-in-the-loop approach," in *CIKM '21: The 30th ACM International Conference on Information and Knowledge Management*. ACM, 2021, pp. 3818–3827.

[11] S. Elhammadi, L. V.S. Lakshmanan, R. Ng, M. Simpson, B. Huai, Z. Wang, and L. Wang, "A high precision pipeline for financial knowledge graph construction," in *Proceedings of the 28th International Conference on Computational Linguistics*. Barcelona, Spain (Online): International Committee on Computational Linguistics, Dec. 2020, pp. 967–977.

[12] S. Zehra, S. F. M. Mohsin, S. Wasi, S. I. Jami, M. S. Siddiqui, and S. M. K. Raazi, "Financial knowledge graph based financial report query system," *IEEE Access*, vol. 9, pp. 69 766–69 782, 2021. [Online]. Available: https://doi.org/10.1109/ACCESS.2021.3077916

[13] Z. Chen, W. Chen, C. Smiley, S. Shah, I. Borova, D. Langdon, R. Moussa, M. Beane, T. Huang, B. R. Routledge, and W. Y. Wang, "Finqa: A dataset of numerical reasoning over financial data," in *EMNLP (1)*. Association for Computational Linguistics, 2021, pp. 3697–3711.

[14] Y. Zhang, H. Dai, Z. Kozareva, A. J. Smola, and L. Song, "Variational reasoning for question answering with knowledge graph," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18)*. AAAI Press, 2018, pp. 6069–6076.

[15] W. Yih, M. Chang, X. He, and J. Gao, "Semantic parsing via staged query graph generation: Question answering with knowledge base," in *ACL (1)*. The Association for Computer Linguistics, 2015, pp. 1321–1331.

[16] A. Talmor and J. Berant, "The web as a knowledge-base for answering complex questions," in *NAACL-HLT*. Association for Computational Linguistics, 2018, pp. 641–651.

[17] J. Shi, S. Cao, L. Hou, J. Li, and H. Zhang, "Transfernet: An effective and transparent framework for multi-hop question answering over relation graph," in *EMNLP (1)*. Association for Computational Linguistics, 2021, pp. 4149–4158.

[18] A. Miller, A. Fisch, J. Dodge, A.-H. Karimi, A. Bordes, and J. Weston, "Key-value memory networks for directly reading documents," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 1400–1409.

[19] H. Sun, B. Dhingra, M. Zaheer, K. Mazaitis, R. Salakhutdinov, and W. W. Cohen, "Open domain question answering using early fusion of knowledge bases and text," in *EMNLP*. Association for Computational Linguistics, 2018, pp. 4231–4242.

[20] H. Sun, T. Bedrax-Weiss, and W. Cohen, "PullNet: Open domain question answering with iterative retrieval on knowledge bases and text," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 2380–2390.

[21] Y. Qiu, Y. Wang, X. Jin, and K. Zhang, *Stepwise Reasoning for Multi-Relation Question Answering over Knowledge Graph with Weak Supervision*. New York, NY, USA: Association for Computing Machinery, 2020, p. 474–482.

[22] W. W. Cohen, H. Sun, R. A. Hofer, and M. Siegler, "Scalable neural methods for reasoning with a symbolic knowledge base," in *International Conference on Learning Representations*, 2020.

[23] A. Saxena, A. Tripathi, and P. Talukdar, "Improving multi-hop question answering over knowledge graphs using knowledge base embeddings," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020, pp. 4498–4507.

[24] N. Abdel Madjid, O. El Kahtib, S. Gao, and D. Difallah, "Hyperkgqa: Question answering over knowledge graphs using hyperbolic representation learning," in *The IEEE International Conference on Data Mining (ICDM)*. IEEE, 2022.

[25] G. Demartini, D. E. Difallah, U. Gadiraju, and M. Catasta, "An introduction to hybrid human-machine information systems," *Found. Trends Web Sci.*, vol. 7, no. 1, pp. 1–87, 2017.

[26] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 3982–3992.

[27] M. Ali, M. Berrendorf, C. T. Hoyt, L. Vermue, S. Sharifzadeh, V. Tresp, and J. Lehmann, "PyKEEN 1.0: A Python Library for Training and Evaluating Knowledge Graph Embeddings," *Journal of Machine Learning Research*, vol. 22, no. 82, pp. 1–6, 2021.

[28] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard, "Complex embeddings for simple link prediction," in *ICML*, ser. JMLR Workshop and Conference Proceedings, vol. 48. JMLR.org, 2016, pp. 2071–2080.

[29] Z. Huang, B. Li, and J. Yin, "Knowledge graph embedding via multiplicative interaction," ser. ICIAI '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 138–142. [Online]. Available: https://doi.org/10.1145/3194206.3194227

[30] A. Kristiadi, M. A. Khan, D. Lukovnikov, J. Lehmann, and A. Fischer, "Incorporating literals into knowledge graph embeddings," in *ISWC (1)*, ser. Lecture Notes in Computer Science, vol. 11778. Springer, 2019, pp. 347–363.