Knowledge Base Question Answering by Case-based Reasoning over Subgraphs

Rajarshi Das*[1], Ameya Godbole*[2], Ankita Naik[1], Elliot Tower[1], Robin Jia[2]
Manzil Zaheer[3], Hannaneh Hajishirzi[4], Andrew McCallum[1]

[1]University of Massachusetts Amherst, [2]University of Southern California,
[3]Google DeepMind, [4]University of Washington

Abstract

Question answering (QA) over real-world knowledge bases (KBs) is challenging because of the diverse (essentially unbounded) types of reasoning patterns needed. However, we hypothesize in a large KB, reasoning patterns required to answer a query type reoccur for various entities in their respective subgraph neigborhoods. Leveraging this structural similarity between local neighborhoods of different subgraphs, we introduce a semiparametric model with (i) a nonparametric component that for each query, dynamically retrieves other similar k-nearest neighbor (KNN) training queries along with query-specific subgraphs and (ii) a parametric component that is trained to identify the (latent) reasoning patterns from the subgraphs of KNN queries and then apply it to the subgraph of the target query. We also propose a novel algorithm to select a query-specific compact subgraph from within the massive knowledge graph (KG), allowing us to scale to full Freebase KG containing billions of edges. We show that our model answers queries requiring complex reasoning patterns more effectively than existing KG completion algorithms. The proposed model outperforms or performs competitively with state-of-the-art models on several KBQA benchmarks.

所需的推理模式多种多样（基本上是无界的）

1 Introduction

Knowledge bases (KBs) store massive amounts of rich symbolic facts about real-world entities in the form of relation triples—(e_1, r, e_2), where e_1, e_2 denote entities and r denotes a semantic relation. KBs can be naturally described as a graph where the entities are nodes and the relations are labelled edges. An effective and user-friendly way of accessing the information stored in a KB is by issuing queries to it. Such queries can be structured (e.g.

Corresponding author emails: `rajarshi@cs.umass.edu` and `ameyagod@usc.edu`. Refer to `https://github.com/rajarshd/CBR-SUBG` for code, models and subgraphs.

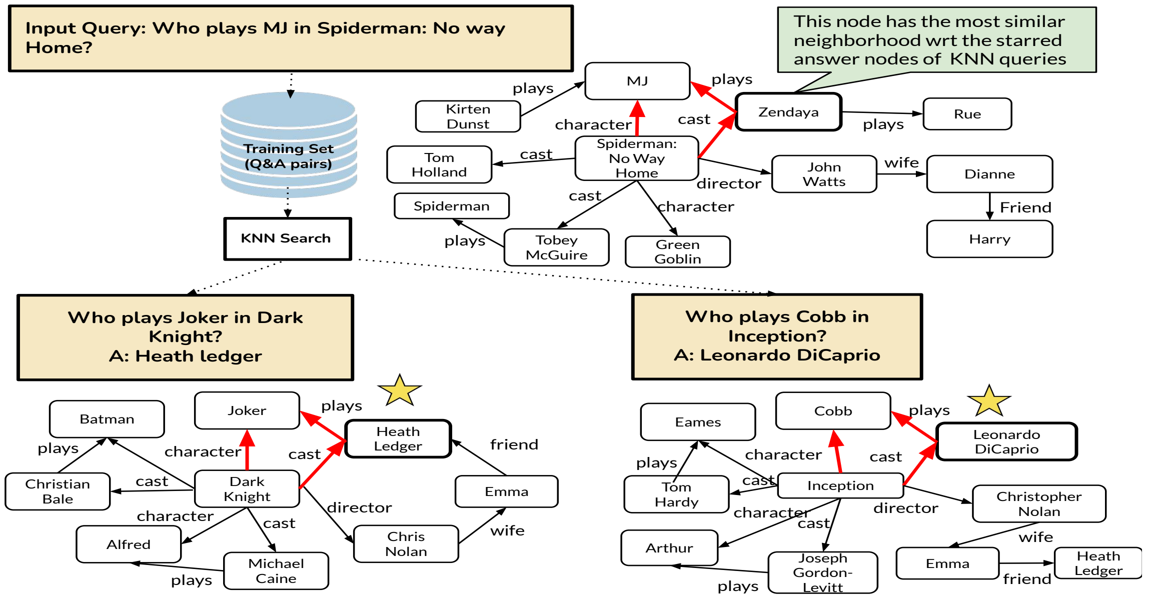


Figure 1: Figure shows an input query and two queries in the training set that are similar to the input query. The relevant subgraph for each query is also shown. Note that the "reasoning patterns" required to answer the queries (edges in red) repeats in the subgraphs of each query. Also note, the corresponding answer nodes (marked as 'star') are analogously located within the reasoning patterns in each subgraph. Thus the answer node can be found by identifying the node in the query subgraph that is most similar to the answer nodes in the subgraph of KNN queries.

queries for booking flights) or unstructured (e.g. natural language queries). The set of KB facts useful for answering a query induce a reasoning pattern—e.g. a chain of KB facts forming a path or more generally a subgraph in the knowledge graph (KG) (set of red edges in Figure 1). It is very laborious to annotate the reasoning patterns for each query at scale and hence it is important to develop weakly-supervised knowledge base question answering (KBQA) models that do not depend on the availability of the annotated reasoning patterns.

We are interested in developing models that can answer queries requiring complex subgraph reasoning patterns. Many previous works in KBQA [31, 48, 8] reason over individual relational paths. However, many queries require a model to reason over a set of facts *jointly*. For example, the query in Figure 1 cannot be answered by considering individual paths. Furthermore, a model has to learn a large number of reasoning patterns because of the diverse nature of possible questions. Moreover, a model may encounter very few examples of a particular pattern during training, making it challenging for the models to learn and encode the patterns entirely in its parameters. A possible solution to

模型在训练期间可能会遇到很少的特定模式示例，这使得模型很难完全在其参数中学习和编码模式。

this challenge might lie in a classical AI paradigm proposed decades back—Case-based Reasoning (CBR) [37]. In a CBR framework, a new problem is solved by retrieving other *similar* problems and reusing their solution to derive a solution for the given problem. In other words, models, instead of memorizing patterns in its parameters, can instead reuse the reasoning patterns of other similar queries, retrieved dynamically during inference. Recently, CBR was successfully used for KB completion by Das et al. [9].

This paper introduces a semiparametric CBR-based model (CBR-SUBG) for QA over KBs with a nonparametric component, that for each query, dynamically retrieves other similar k-nearest neighbor (KNN) queries from the training set. To retrieve similar queries, we use masked sentence representation of the query [39] obtained from pre-trained language models.

本质是：如何找出最相似的query?

We hypothesize that the reasoning patterns required for answering similar queries reoccur within the subgraph neighborhood of entities present in those queries (Figure 1). The answer nodes for each query are also analogously nestled within the reasoning patterns (marked as 'starred nodes' in Figure 1) of the query subgraphs, i.e. they have similar neighborhoods. However, we do not have annotated reasoning patterns that could be used to search for the answer node. Moreover, a subgraph can have tens of thousands of entity nodes. How do we still identify the answer nodes in the query subgraph?

To identify the answer nodes, our model has a parametric component comprising of a graph neural network (GNN) that is trained to identify the (latent) reasoning patterns from the subgraphs of KNN queries and apply it to the subgraph of the target query. GNNs have been shown to be effective in encoding structural properties of local neighborhoods in the node representations [11, 26]. We leverage node representations obtained from GNNs for finding answer nodes. Specifically, the answer nodes are identified by performing a nearest neighbor search for finding the most similar nodes in the query subgraph w.r.t the representation of answer nodes in the KNN subgraph. The parametric model is trained via contrastive learning (§3.3) [6, 16].

A practical challenge for KBQA models is to select a compact subgraph for a query. The goal is to ensure that the subgraph has high recall and is small enough to fit into GPU memory for gradient-based learning. Many KBQA methods usually consider few hops of edges around entities as the query subgraph [31, 35] leading to query-independent and (often) large subgraphs, because of the presence of hub nodes in large KBs. We propose an *adaptive* subgraph collection method tailored for each query where we use our nonparametric approach of retrieving KNN queries to help gather the query subgraph leading to compact subgraphs with higher recall of reasoning patterns (§ 3.2).

An important property of nonparametric models is its ablility to grow and reason with new data. Being true to its nonparametric design, CBR-SUBG uses sparse representations of entities that makes it easy to represent new entities. Moreover, we also demonstrate that the performance of CBR-SUBG improves as more evidence is retrieved, suggesting that

CBR-SUBG can reason with new evidence.

Contributions. To summarize, this paper introduces CBR-SUBG, a semiparametric model for weakly-supervised KBQA that retrieves similar queries and utilizes the similarities in graph structure of local subgraphs to answer a query (§3.3). We also propose a practical algorithm for gathering query-specific subgraphs that utilizes the retrieved KNN queries to produce compact query-specific subgraphs (§3.2). We show that CBR-SUBG can model (latent) subgraph reasoning patterns (§4.1), more effectively than parametric models; can reason with new entities (§4.1) and new evidence (§4.3). Lastly, we perform competitively with state-of-the-art KBQA models on multiple benchmarks. For example, on the FreebaseQA dataset [22], CBR-SUBG outperform most competitive baseline by 14.45 points.

2 Related Work

CBR-based models for KB reasoning. Recently, Das et al. [9] proposed a CBR-based technique for KB completion. However, their work has several limitations. Firstly, it can only model simple linear chains. Secondly, it uses exact symbolic matching to find similarities in patterns between cases and the query. Lastly, it cannot handle natural language queries and only works with structured slot-filling queries. In contrast, CBR-SUBG can model arbitrary reasoning patterns; uses soft-matching by comparing representations of answer nodes and can handle natural language queries. Lastly, our method outperforms Das et al. [9] on various benchmarks. A follow up work of Das et al. [10] proposed a CBR model that can handle natural language queries, however that work requires the availability of annotated reasoning patterns for training, an important distinction from our work that does not need any annotation of reasoning patterns.

Semiparametric models for KBQA. GraftNet [41] and PullNet [42] are two semiparametric models for KBQA where they like us, provide both a mechanism of collecting a query-subgraph and reasoning over them. For their nonparametric component, these work employ a retrieval process where a parametric model classifies which edges would be relevant to the query. Being parametric, these models cannot generalize to new type of questions without re-training the model parameters. However, our nonparametric approach will work as it retrieves similar queries on-the-fly. For their reasoning model, both works use a graph convolution model and treat the answer prediction as a node classification task. However, unlike us they do not reason with subgraphs of similar KNN queries. Lastly, we compare extensively with them and outperform them on multiple benchmarks. Our approach also has similarities with retriever-reader architecture for open-domain QA over text [4] where a retriever selects evidence specific to the query and the reader reasons with them to produce the answer.

Graph representations using contrastive learning. Recently, there has been a lot of work on learning graph representations via contrastive learning [19, 52, 40, 32, 59] where they create two views of the same graph by randomly dropping edges and nodes. Next, the two views of subgraphs are treated as positive pair and their representations are contrasted wrt other negative samples. Our work differs from them because we do not create different views of the same graph, rather we follow the CBR hypothesis and make node representations of answer nodes of two query-specific subgraphs similar provided the queries have relational similarity.

Semantic parsing. Classic works in semantic parsing [54, 56, 55] were among the early works to use statistical learning to convert queries to executable logical forms. However, these work needed annotated logical forms during training. Follow up work [1, 27] learned semantic parser directly from question answer pairs. However, their model relied on simple hand crafted features. Recent approaches to semantic parsing [35, 20] uses powerful neural models and achieve strong performance. However unlike us, these parametric models learn dense representation of entities and hence will not generalize to unseen entities like our apprach.

Inductive KB reasoning. Our model is also related to Teru et al. [44] as it explores KB reasoning in an inductive setting. They also have a sparse representation of entities. However, the task they consider is predicting a KB relation between two nodes which is an easier task than performing KBQA using natural language queries.

Graph neural networks. A model like CBR-SUBG is possible for KBQA because of tremendous progress made in graph representation learning by message passing neural networks [26, 45, 38, inter-alia]. CBR-SUBG is not dependent on any specific message passing scheme and can work with any GNN architecture that can operate over heterogenous knowledge graphs. For our experiments, we use the widely used relational-GCN [38]. Further related work is included in the appendix (§F).

3 Model

This section describes the nonparametric and parametric components of CBR-SUBG. In CBR, a case is defined as an abstract representation of a problem along with its solution. In our KBQA setting, a case is a natural language query along with its answer. Note in KBQA, answers are entities in the KB (or nodes in KG).

Task description. Let q be a natural language query and let $\mathcal{K}$ be a symbolic KG that needs to be queried to retrieve an answer list $\mathcal{A}$ containing the answer(s) for q. We assume access to a training set $\mathcal{D} = \{(q_1, \mathcal{A}_1), (q_2, \mathcal{A}_2), \ldots (q_N, \mathcal{A}_N)\}$ of query-answer pairs where $\mathcal{A}_i$ denote the list of answer nodes for q_i. The training set $\mathcal{D}$ forms the 'case-base'. The reasoning pattern (a.k.a. logical form) required to answer a query are the set of KG edges required to deduce the answer to q. Let P_q denote this set of edges. For example, in Figure 1, for q = "Who plays 'MJ' in No Way Home?", P_q = {(MJ, played_by, Zendaya), (No Way Home, has_character, MJ), (No Way Home, cast, Zendaya)}. We define reasoning pattern 'type' for a pattern as the set of edges where the entities have been replaced by free variables. For example, $T(P_q)$ = {(M, played_by, Z), (S, has_character, M), (S, cast, Z)}. It should be noted that CBR-SUBG does not assume access to annotated P_q.

Method overview. For input q and $\mathcal{K}$, CBR-SUBG first retrieves k similar query-answer(s) w.r.t q from $\mathcal{D}$ (§ 3.1). Denote this retrieved set as $\text{kNN}_q \subset \mathcal{D}$. Next, CBR-SUBG finds query-specific subgraphs $\mathcal{K}_{q_i}$ for each query in $\{q\} \cup \text{kNN}_q$ (§ 3.2). According to the CBR hypothesis, the reasoning required to solve a new problem will be similar to the reasoning required to solve other similar problems. Similarly for our KBQA setting, we hypothesize that the reasoning pattern type, $T(P_q)$ repeats across the neighborhood of query subgraphs of kNN_q. Next, CBR-SUBG uses GNNs to encode local structure into node representations (§ 3.3). Now, if the CBR hypothesis holds true, then the representation of the answer nodes in each query subgraphs will be similar as the local structure around them share similarities. Hence, the answer node of the given query q can be identified by searching for the node that has the most similar representations to the answer nodes in the query subgraphs of kNN_q.

3.1 Retrieval of Similar Cases

Given the input query q, CBR-SUBG first retrieves other similar cases from the training set. We represent the query by embeddings obtained from large pre-trained language models [28]. Inspired by the recent advances in neural dense passage retrieval [23], we use a ROBERTA-base encoder to encode each question independently. A single representation is obtained by mean pooling over the token-level representations.

Generally, we want to retrieve questions that express similar relations rather than retrieving questions that are about similar entities. For example, for the query, 'Who played Natalie Portman in Star Wars?', we would like to retrieve queries such as 'Who played Ken Barlow in Coronation St.?' instead of 'What sports does Natalie Portman like to play?'. To obtain entity-agnostic representation, we replace the entity spans with a special '[MASK]' token in the query, i.e. the original query becomes 'Who played [MASK] in [MASK]?'. The entity masking strategy has previously been successfully used in learning entity-independent relational representations [39]. The similarity score between two queries is given by the inner product between their normalized vector representations (cosine similarity). We pre-compute the representations of queries in the train set. During inference, the most similar query representations are obtained by doing a nearest neighbor search over the representations stored in the case-base.

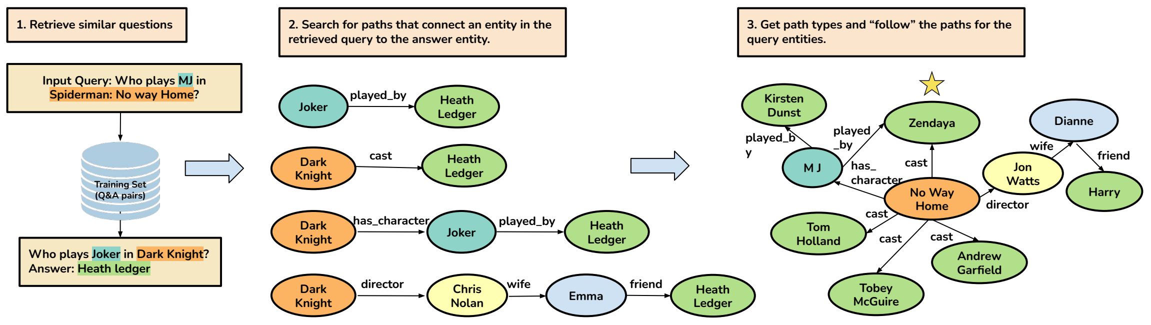


Figure 2: Figure shows the query-subgraph selection procedure with 1-nearest neighbor retrieved query. Graph paths connecting the entities in the retrieved query and its answer are collected. Next the sequence of relations (path types) are gathered and are then followed starting from the entity in the given query. All the edges spanned by this process are collected to form the query-specific subgraph. This process is repeated for each of the k retrieved queries.

3.2 Query-subgraph Selection

A practical challenge for KBQA models is to select a subgraph around the entities present in the query. The goal is to ensure that the necessary reasoning patterns and answers are included while producing a graph small enough to fit into GPU memory for gradient-based learning. A naïve strategy to select a subgraph is to consider all edges in 2-3 hops around the query entities. This strategy leads to subgraphs which are independent of the query. Moreover, in large KGs like Freebase [2], considering the full 2 or 3-hop subgraph often leads to accumulation of millions of edges because of the presence of hub nodes.

We propose a nonparametric approach of query subgraph collection that utilizes the retrieved cases, kNN$_q$, from the last step (Figure 2). For each of the retrieved case, chains of edges (or paths) in the graph that connect the entity in the retrieved query to its answers are collected by doing a depth-first search. (Note, since the retrieved queries are from the training set, we know the answer to them). Next, the sequence of relations are collected from each chain and they are followed starting from the entities of the input query. If a chain of relations do not exist from the query, then they are simply ignored. This process is repeated for each of the retrieved cases. Note that, not all chains collected from the nearest neighbors are meaningful for the query. For example, the last (3-hop) chain collected in Figure 2 is not relevant for answering the query and even though it ended at the answer for the retrieved query, the same is not the case for the input query. Such paths are often referred to as spurious paths or evidence [20]. All the edges gathered by following this process form the subgraph for the input query. The underlying idea behind the subgraph selection procedure is simple — since the paths connect queries and answers of similar queries, they should also be relevant for answering the given query.

Our subgraph selection procedure is similar to the approach proposed in [9]. However, Das et al. [9] do not use this approach to collect a query-specific subgraph. Rather it uses each of the paths to *independently* predict the answer to a query. In contrast, we collect all edges in the path to form a subgraph and then reason jointly over the subgraph of the query as well as the subgraph of retrieved cases as detailed in the next sub-section.

3.3 Reasoning over Multiple Subgraphs

This section describes how CBR-SUBG reasons across the subgraphs of the given query and the subgraphs of the retrieved cases. We use GNNs [36] to encode the local structure into the node representations of each subgraph. During training, the answer node representations of different subgraphs are made more similar to each other in comparison to other non-answer nodes. Inference reduces to searching for the most similar node in the query subgraph w.r.t the answer nodes in the KNN-subgraphs.

Modern GNNs employ a neighborhood aggregation strategy (message passing) where a node representation is iteratively updated by aggregating representations from its neighbors [14]. Let $\mathcal{G}_q = (V_q, E_q)$ represent the subgraph for a query q obtained from (§3.2). Let $\boldsymbol{X}_v$ denote the node feature vectors for each $v \in V_q$.

Input node representations. A property of nonparametric models is its ability to represent, reason and grow with new data. Knowledge graphs store facts about the world and as the world evolves, new entities and facts are added to the KG. Models developed for KG reasoning [3, 38, 43, inter-alia] learn dense representations of a fixed vocabulary of entities and are hence unable to handle new entities added to the KG. Following our nonparametric design principles, each entity node is represented as a sparse vector of its outogoing edge (relation) types, i.e. $\boldsymbol{x}_v \in \{0,1\}^{|\mathcal{R}|}$ where $\mathcal{R}$ denotes the set of relation types in the KG. If entity x_v has m distinct outgoing edge types, then the dimension corresponding to those types are set to 1. This is an extremely simple and flexible way of representing entities which also expresses the local structural information around each node. Also note that, as new entities are added or new facts are updated about an entity, the sparse representation makes it very easy to represent new entities or update existing embeddings.

Relative distance embedding. Each query-specific subgraph $\mathcal{G}_q$ has a few special entities—the entities present in the input query. This is because the reasoning pattern is usually in the immediate subgraph surrounding the query entity. We treat the query entities as 'center' entities and append a relative distance embedding to every other node in $\mathcal{G}_q$ [57, 44]. Specifically, for each node, the representation $\boldsymbol{x}_v$ is appended with an one-hot distance embedding $\boldsymbol{x}_{\boldsymbol{d}} \in \{0,1\}^{|d|}$ where the component corresponding to the shortest distance from the query entity is set to 1. In practice, we consider subgraphs upto 3-hops from the query entities, i.e. $d = 4$. For queries with multiple query entities, the minimum distance is considered.

Message passing. Our GNN uses the graph structure and the sparse input node features $\boldsymbol{X}_v$ to learn intermediate node features capturing the local structure within them. We follow the general message-passing scheme where a node representation is iteratively updated by combining it with aggregation of it's neighbors' representation [49]. In particular, the l^{th} layer of a GNN is,

$$\boldsymbol{a}_v^l = \text{AGGREGATE}^l \left(\left\{ \boldsymbol{h}_s^{l-1} : s \in \mathcal{N}(v) \right\}, \boldsymbol{h}_v^{l-1} \right), \tag{1}$$

$$\boldsymbol{h}_v^l = \text{COMBINE}^l \left(\boldsymbol{h}_v^{l-1}, \boldsymbol{a}_v^l \right), \tag{2}$$

where, $\boldsymbol{a}_v^l$ denote the aggregated message from the neighbors of node v, $\boldsymbol{h}_v^l$ denote the node representation of node v in the l-th layer and $\mathcal{N}(v)$ denotes the neighboring nodes of v. Since KGs are heterogenous graphs with labelled edges, we adopt the widely used multi-relational R-GCN model model [38] which defines the aggregate step as: $\boldsymbol{a}_v^l = \sum_{r=1}^{\mathcal{R}} \sum_{s \in \mathcal{N}_r(v)} \frac{1}{|\mathcal{N}_r(v)|} W_r^l h_s^{l-1}$ and the combine step as $h_v^l = \text{ReLU}(W_{\text{self}}^l h_v^{l-1} + \boldsymbol{a}_v^l)$. For each answer node, we consider the representation obtained from the last layer.

Training. Let a_i, a_j be an answer node in the corresponding query-subgraphs of q_i and q_j (i.e. $\mathcal{G}_{q_i}$, $\mathcal{G}_{q_j}$) respectively. Let $\text{sim}(\boldsymbol{a}_i, \boldsymbol{a}_j) = \boldsymbol{a}_i^\top \boldsymbol{a}_j / \|\boldsymbol{a}_i\| \|\boldsymbol{a}_j\|$ denote the inner product between ℓ_2 normalized answer representations (i.e. cosine similarity). In general there can be multiple answer nodes for a query. Let $\mathcal{A}_j$ denote the set of all answer nodes for query q_j in its subgraph $\mathcal{G}_{q_j}$. Let $\text{sim}(\boldsymbol{a}_i, \mathcal{A}_j) = \frac{1}{|\mathcal{A}_j|} \sum_{a_j \in \mathcal{A}_j} \text{sim}(\boldsymbol{a}_i, \boldsymbol{a}_j)$, i.e. $\text{sim}(\boldsymbol{a}_i, \mathcal{A}_j)$ represents the mean of the scores between a_{q_i} and all answer nodes in $\mathcal{G}_{q_j}$. We aggregate the similarity score from all retrieved queries kNN_{q_i} for the current query q_i.

The loss function we use is,

$$-\log \frac{\sum_{a_i \in \mathcal{A}_i} \exp(\sum_{q_j \in \text{KNN}_{q_i}} \text{sim}(\boldsymbol{a}_i, \mathcal{A}_j)/\tau)}{\sum_{x_i \in \mathcal{V}(\mathcal{G}_{q_i})} \exp(\sum_{q_j \in \text{KNN}_{q_i}} \text{sim}(\boldsymbol{x}_i, \mathcal{A}_j)/\tau)}, \tag{3}$$

where, $\mathcal{A}_j$ denotes the set of all answer nodes in $\mathcal{G}_{q_j}$ for a $q_j \in \text{kNN}_{q_i}$, x_i goes over all nodes in query-subgraph $\mathcal{G}_{q_i}$ and τ denotes a temperature parameter. In other words, the loss encourages the answer nodes in $\mathcal{G}_{q_i}$ to be scored higher than all other nodes in $\mathcal{G}_{q_i}$ w.r.t the answer nodes in the retrieved query subgraphs. This loss is an extension of the the normalized temperature-scaled cross entropy loss (*NT-Xent*) used in Chen et al. [5].

Inference. During inference, message passing is run over each of the query-subgraph $\mathcal{G}_{q_i}$ and the subgraphs $\mathcal{G}_{q_j}$ of its k retrieved queries $q_j \in \text{kNN}_{q_i}$ to obtain the node representations and the highest scoring node in $\mathcal{G}_{q_i}$ w.r.t all the answer nodes in the retrieved query sub-graphs is returned as the answer.

$$\boldsymbol{a}_i = \arg\max_{x_i} \sum_{x_i \in \mathcal{V}(\mathcal{G}_{q_i})} \exp\left(\sum_{q_j \in \mathrm{kNN}_{q_i}} \mathrm{sim}(\boldsymbol{x}_i, \mathcal{A}_j) \right) \tag{4}$$

4 Experiments

In this section, we demonstrate the effectiveness of the semiparametric approach of CBR-SUBG and show that the nonparametric and parametric component offer complementary strengths. For example, we show that the model performance improves as more evidence is dynamically retrieved by the nonparametric component (§4.3). Similarly, CBR-SUBG can handle queries requiring reasoning patterns more complex than simple chains (i.e. subgraphs) because of the inductive bias provided by GNNs (§4.1). It can handle new and unseen entities because of the sparse entity input features as a part of its design (§4.1). We also show that the nonparametric subgraph selection of CBR-SUBG allows us to operate over a massive real-world KG (full Freebase KG) and obtain very competitive performance on several benchmark datasets including WebQuestionsSP [51], FreebaseQA [22] and MetaQA [58].

4.1 Reasoning over Complex Patterns

We want to test whether CBR-SUBG can answer queries requiring complex reasoning patterns. Note that, the reasoning patterns are always latent to the model, i.e. the model has to answer a given query from the query-subgraph and the retrieved KNN-subgraphs *without* any knowledge of the structure of the pattern.

To test the model capacity to identify reasoning patterns, we devise a controlled setting in which the model has to infer reasoning patterns of various shapes (Figure 3), inspired by Ren et al. [34]. Note that in their work, the task was to execute the input structured query on an incomplete KB, i.e, the shape of the input patterns are known to the model. In contrast, in our setting, the model has to find the answer node (marked ●), which is nestled in each of the structured pattern without the knowledge of the pattern structure. Also note, there can be multiple nodes of the same type as the answer type, so the task cannot be completed by solving the easier task of determining entity types. Instead the model has to identify the specific ● nodes which are at the end of the reasoning patterns (there can be multiple ● nodes in the graph).

Data generation process. We first define a type system with a set of entity types $\mathcal{E}$ and relation types $\mathcal{R}$. The type-system also specifies a set of 'allowed relation types' between different pairs of entity types. For example, an 'employee' KB relation is defined between an 'organization' and 'people' entity types. Entities (or nodes $\mathcal{V}$) are then generated

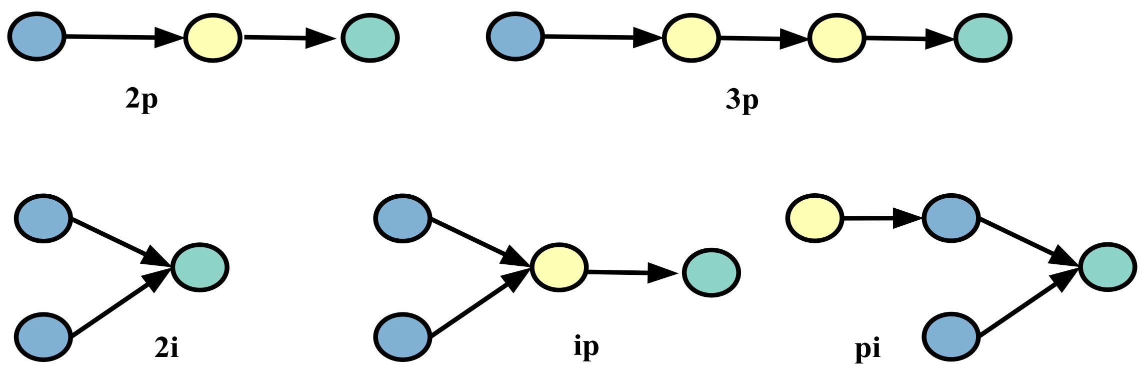


Figure 3: Various shapes of reasoning pattern types considered.

uniformly from the set of entity types $\mathcal{E}$. Next, relation edges (uniformly sampled from the allowed types) are joined between a pair of nodes with a probability p following the Erdős-Rényi model [12] of random graph generation. To ensure that models only rely on the graph structure, each graph has a 'unique' set of entities and no two graphs share entities. This also effectively tests how much the nonparametric property of CBR-SUBG can reason with unseen entities. More details regarding the hyperparameters $\mathcal{E}, \mathcal{R}, \mathcal{V}$ are included in the appendix B.

Pattern generation. A pattern is next sampled from the set of shapes shown in Figure 3. The sampled pattern merely suggests the structure of the desired pattern. '*Grounding*' a pattern shape involves assigning each nodes with an entity present in a generated graph. Similarly each edge of the pattern type is assigned a relation from the set of allowed relation types. After grounding the pattern, we "insert" the pattern in the graph. Since the nodes of the grounded pattern already exists in the graph, inserting a pattern in the graph amounts to adding the edges of the grounded pattern to the graph that already did not exist in it. We also define a 'pattern type' — that refers to a pattern whose edges have been assigned relation types but the nodes have not been assigned to specific entities (bottom-left corner in Figure 4). Each pattern type is assigned an identifier and queries with the same pattern type are grouped together.

We generate 1000 graphs in each of the training, validation and test sets. We generate 200 pattern types whose shapes are uniformly sampled from the 5 shapes shown in Figure 3. Therefore, there are around 40 examples of each pattern shape and around 5 examples of each pattern type. This is consistent with real-world setting where a model will encounter a reasoning pattern only very few times during training. For a query with a particular pattern type, other training queries with the same pattern type form its nearest neighbors.

Baselines. Because of the inductive nature of this task where only new entities are seen

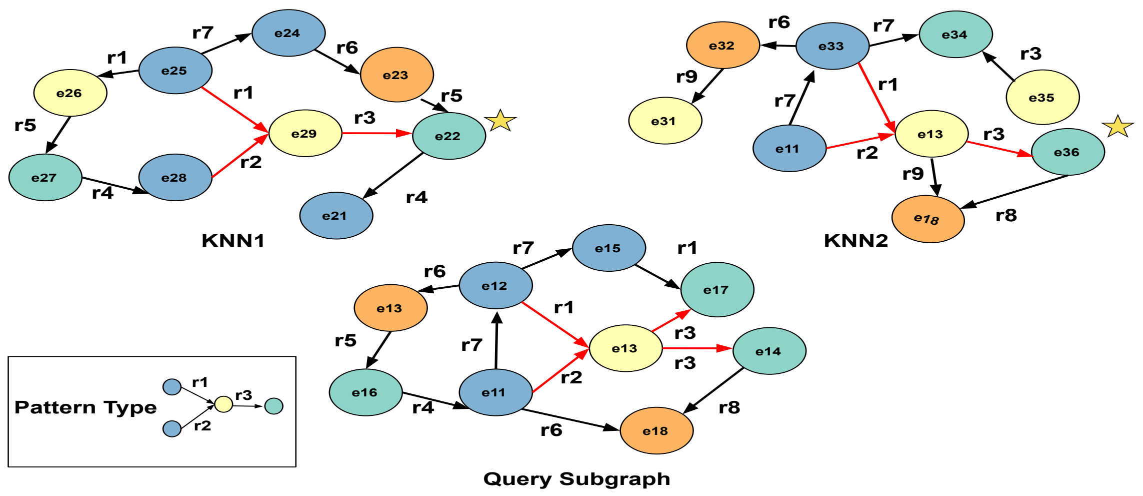


Figure 4: Setup for reasoning over subgraph patterns. Note that the same 'pattern type' repeats across subgraphs. Also note, that the query graph (bottom) has two answer nodes (e14, e17). Since every subgraph has its own set of unique entities, hence a model has to reason over the similarities in graph structures to find the answer node in the query-subgraph.

at test time, most parametric KG reasoning algorithms [3, 50, 43] will not work out of the box. We extend the widely used KG reasoning model — TransE [3] to work in the inductive setting. Specifically, instead of having a fixed vocabulary of entities, the objective function is computed on the dense representation obtained from the output of the GNN layers. This also makes the comparision with CBR-SUBG fair since it also operates on the same representations albeit with a contrastive loss. KG completion algorithms also need a query-relation as input. Each pattern type for a query serves as the query relation. Apart from the parametric baseline, we also test another nonparametric baseline that operates on individual paths rather than subgraphs similar to the approach proposed by Das et al. [9] (CBR-Path in Table 1). Comparing CBR-SUBG to CBR will help us understand the importance of modeling subgraph patterns rather than simple chains.

Does CBR-SUBG have the right inductive bias? The first research question that we try to answer is, if CBR-SUBG has the right inductive bias for this task. We test CBR-SUBG that has undergone *no training*, i.e. the parameters of the GNN are randomly initialized. Note the model still takes as input the sparse representation of entities. This experiment will help us answer if the node representations actually capture the local structure around them and whether the answer node can be found by doing a search w.r.t the answer nodes in the KNN-query subgraphs.

Table 1 reports the strict hits@1 on this task, i.e. to score a query correctly, a model

Model	**2p**	**3p**	**2i**	**ip**	**pi**	**avg.**
CBR-SUBG (No training)	68.56	84.35	23.00	34.85	35.35	47.28
GNN + TransE	80.03	74.49	80.00	52.67	81.53	72.69
CBR-Path	69.71	54.39	**100.00**	69.12	51.24	71.09
CBR-SUBG	**96.64**	**88.43**	90.46	**70.02**	**86.81**	**85.68**

Table 1: Hits1(%) for predicting *all* the answer nodes correctly. CBR-SUBG without any training performs decently suggesting that it has the right inductive bias for the task. On training, the performance improves on all subgraph patterns.

Model	**MetaQA**			**WebQSP**
	1-hop	**2-hop**	**3-hop**	
KVMemNN [29]	95.8	25.1	10.1	46.7
GraftNet [41]	97.0	94.8	77.7	66.4
PullNet [42]	97.0	99.9	91.4	68.1
SRN [33]	97.0	95.1	75.2	-
ReifKB [7]	96.2	81.1	72.3	52.7
EmbedKGQA [35]	**97.5**	98.8	94.8	66.6
NSM [20]	97.2	99.9	98.9	**74.3**
CBR-SUBG (Ours)	97.1	99.8	**99.3**	72.1

Table 2: Performance on WebQSP and MetaQA benchmarks.

has to identify and rank *all* answer nodes above all other nodes in the graph. The first row of Table 1 shows the results. For comparison, a random performance on this task is $\frac{1}{|\mathcal{V}|} = \frac{1}{120} = 0.83\%$. As it is clear from the results, an un-trained CBR-SUBG achieves performance much higher than random performance. Its quite high for the simple 2p and 3p patterns. For other patterns that need the more complicated intersection operation, the performance degrades, but is still much higher than random.

Our results. On training CBR-SUBG, the performance of the model drastically improves for each pattern type reaching an average performance of 85.68%. The performance on pattern types which are more complex than chains (ip, pi) etc are worse than chain-type patterns (2p, 3p) suggesting that our task is non-trivial[1].

On comparison to parametric model. This experiment helps us understand whether a model can learn to memorize and store patterns effectively (for each query relations) when it has seen few examples of that pattern during training. Row 2 of Table 1 shows the performance of GNN + TransE model. We find that the parametric model performs worse than CBR-SUBG on all the query types reaching an average performance of 13% point below

[1]We will release the code, dataset and data generation pipeline for reproducibility and further research

CBR-SUBG. This shows that a semiparametric model with a nonparametric component that retrieves similar queries at inference can make it easier for the model to reason effectively. In practice, we had to train this model for a much longer time than training CBR-SUBG.

On comparison to path-based model. From Table 1, we can see that CBR-SUBG outperforms CBR-path by more than 14% points suggesting that reasoning over subgraphs is a more powerful approach that reasoning with each paths independently. On the '2i' pattern, CBR-path outperforms CBR-SUBG since '2i' can be seen as 2 independent paths intersecting at one node and CBR-path is able to model that perfectly. However, when the pattern needs composition and intersection and path-traversal, CBR-path struggles and performs much worse.

4.2 Performance on benchmark datasets

Next, we test the performance of CBR-SUBG on various KBQA benchmarks — MetaQA [58], WebQSP [51] and FreebaseQA [22]. MetaQA comes with its own KB. For other datasets, the underlying KB is the full Freebase KB containing over 45 million entities (nodes) and 3 billion facts (edges). Please refer to the appendix for details about each dataset (§C).

Model	Accuracy
KB-only models	
HR-BiLSTM [53]	28.40
KEQA [21]	28.73
KBQA-Adapter [47]	28.78
FOFE [22]	37.00
BuboQA [30]	38.25
CBR-SUBG (Ours)	**52.07**
LM pre-training + KB	
EAE [13]	53.4
FAE [46]	**63.3**

Table 3: Top-1 % accuracy on the FreebaseQA dataset. The top section reports performance of models that operate only on KBs. The bottom section reports performance on models that also use additional knowledge stored in large language models.

Our main baselines are the two semiparametric models that provide both a mechanism to gather query subgraphs for a given query and reason over them to find the answer — GraftNet [41], PullNet [42]. GraftNet uses personalized page rank to determine which edges are relevant for a particular query and PullNet uses a multi-step retriever that at each step, classifies if an edge is relevant to the current representation of the query. For their reasoning model, both works use a graph convolution model and treat the answer prediction as a node classification task. However unlike us, they do not use query-subgraphs of KNN queries. Followup KBQA works [35, 20, inter-alia] use the query-specific graphs provided by GraftNet from their open-source code and do not provide a mechanism to gather query-specific subgraphs. However, for completeness, we report and compare with those methods as well.

Table 2 reports the performance on WebQSP and all three partitions of MetaQA. When compared to GraftNet and PullNet, CBR-SUBG performs much better on an average on both the datasets. On the more challenging 3-hop subset of MetaQA, CBR-SUBG outper-

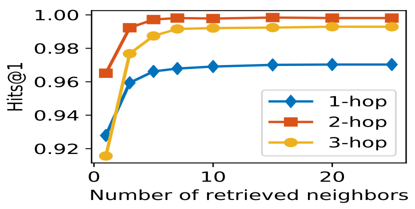


Figure 5: Performance of CBR-SUBG as more nearest neighbors are introduced at test-time. Results on different partitions of MetaQA

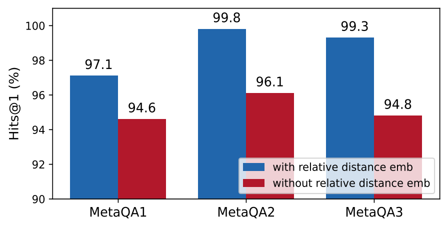


Figure 6: Performance with and without relative distance embedding. The relative distance from the query entity clearly is important for achieving good performance.

forms PullNet by more than 7 points and GraftNet by more than 15 points. This shows that even though these two models use a GNN for reasoning, using information from subgraphs from similar KNN queries leads to much better performance. On WebQSP, we outperform all models except the recently proposed NSM model [20]. But as we noted before, NSM operates on the subgraph created by GraftNet and does not provide any particular mechanism to create its own query-specific subgraph (an important contribution of our model). Moreover NSM is a parametric model and will not have some advantages of nonparametric architectures such as ability to handle new entities and reasoning with more data. Table 3 reports the results on the FreebaseQA dataset, which contains real trivia questions obtained from various trivia competitions. Thus the questions can be challenging in nature. We compare with other KBQA models reported in Han et al. [17]. Most of the models are pipelined KBQA systems that rely on relation extraction to map the query into a KB edge. CBR-SUBG outperforms all the models by a large margin. We also report the performance on two models that use large LMs and large-scale pre-training. CBR-SUBG, which only operates on the KB has a performance very close to the performance of Entity-as-Experts model [13]. We leave the integration of large LMs into our parametric reasoning component as future work.

4.3 Analysis

How effective is our adaptive subgraph collection strategy? Table 4 reports few average graph statistics for the query-subgraphs collected by our graph-collection strategy. We also compare to GraftNet's subgraphs[2]. As can be seen, our adaptive graph collection strategy produces much more compact and smaller graphs *while* increasing recall of answers. We also consistently find that our graph contains relations which is more relevant to the

[2]Code and artifacts for PullNet is not available

Subgraph	#edges	#relations	#entities	coverage(%)
WebQSP				
Graft-net	4306.00	294.69	1447.68	89.93%
CBR-SUBG	2234.35	40.38	1627.37	94.30%
% diff	**-48.11%**	**-86.5%**	+12.4%	**+4.85%**
MetaQA-3				
Graft-net	1153.0	18.00	497.00	99%
CBR-SUBG	89.21	4.72	77.52	99.9%
% diff	**-92.21%**	**-73.78%**	**-84.40%**	**+0.91%**

Table 4: Our adaptive subgraph collection strategy produces a compact subgraph for a query while increasing recall.

questions than the subgraph produced by GraftNet[3]. Table 5 reports the performance of CBR-SUBG when trained and tested on the subgraph obtained from Graftnet and our adaptive procedure, demonstrating the effectiveness of our adaptive subgraph collection method.

Can CBR-SUBG reason with more evidence? A desirable property of nonparametric models is to be able to 'improve' its prediction as more evidence is made available. We test CBR-SUBG by taking a trained model and issuing it an increasing number of nearest neighbor queries. As we see from Figure 5, the performance of CBR-SUBG drastically improves as we increase the number of nearest neighbors from 1 to 7 and then increases at a lower rate and converges at around 10 nearest neighbors. This is because, the model has all the required information it needs from its nearest neighbors.

Subgraph	WebQSP	MetaQA-3
GraftNet	65.61%	96.90%
Adaptive	**71.92%**	**99.30%**

Table 5: Performance of CBR-SUBG with adaptive subgraph and GraftNet subgraph

Are relative distance embeddings important? Figure 6 shows the performance of CBR-SUBG with and without the relative distance embeddings (§3.3). It is clear that capturing the relative distance from the query entities provide serves as a helpful feature for the model.

5 Conclusion

In this work, we explored a semiparametric approach for KBQA. We demonstrated CBR-SUBG poses several desirable properties approach in which nonparametric and parametric

[3]We will also released the collected subgraphs for all the datasets. Refer to Appedix E

component offer complementary strengths. By retrieving similar queries and utilizing the similarities in graph structure of local subgraphs to answer a query, our approach is able to handle complex questions as well as generalize to new types of questions. Exploring different types of parametric models with different reasoning capabilities (LMs, GNNs, etc.) would be an interesting future research direction. Another avenue of potential research would be a never-ending learning type of system where we keeps adding newly discovered facts in the nonparametric part.

Acknowledgments

This work funded in part by the IBM Research AI through the AI Horizons Network and in part by the Chan Zuckerberg Initiative under the project Scientific Knowledge Base Construction and in part by the National Science Foundation under Grant Number IIS-1763618 and in part by the the National Science Foundation under Grant Number IIS-1922090 and in part by the Defense Advanced Research Projects Agency (DARPA) via Contract No. FA8750-17-C-0106 under Subaward No. 89341790 from the University of Southern California and in part by the Office of Naval Research (ONR) via Contract No. N660011924032 under Subaward No. 123875727 from the University of Southern California. The work reported here was performed using high performance computing equipment obtained under a grant from the Collaborative R&D Fund managed by the Massachusetts Technology Collaborative.

References

[1] J. Berant, A. Chou, R. Frostig, and P. Liang. Semantic parsing on freebase from question-answer pairs. In *EMNLP*, 2013.

[2] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. In *ICDM*, 2008.

[3] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In *Neurips*, 2013.

[4] D. Chen, A. Fisch, J. Weston, and A. Bordes. Reading wikipedia to answer open-domain questions. In *ACL*, 2017.

[5] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020.

[6] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*, 2005.

[7] W. W. Cohen, H. Sun, R. A. Hofer, and M. Siegler. Scalable neural methods for reasoning with a symbolic knowledge base. *arXiv preprint arXiv:2002.06115*, 2020.

[8] R. Das, S. Dhuliawala, M. Zaheer, L. Vilnis, I. Durugkar, A. Krishnamurthy, A. Smola, and A. McCallum. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. In *ICLR*, 2018.

[9] R. Das, A. Godbole, S. Dhuliawala, M. Zaheer, and A. McCallum. A simple approach to case-based reasoning in knowledge bases. In *AKBC*, 2020.

[10] R. Das, M. Zaheer, D. Thai, A. Godbole, E. Perez, J.-Y. Lee, L. Tan, L. Polymenakos, and A. McCallum. Case-based reasoning for natural language queries over knowledge bases. In *EMNLP*, 2021.

[11] D. Duvenaud, D. Maclaurin, J. Aguilera-Iparraguirre, R. Gómez-Bombarelli, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams. Convolutional networks on graphs for learning molecular fingerprints. *arXiv preprint arXiv:1509.09292*, 2015.

[12] P. Erdos, A. Rényi, et al. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 1960.

[13] T. Févry, L. B. Soares, N. FitzGerald, E. Choi, and T. Kwiatkowski. Entities as experts: Sparse memory access with entity supervision. In *EMNLP*, 2020.

[14] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. In *ICML*, 2017.

[15] J. Gu, Y. Wang, K. Cho, and V. O. Li. Search engine guided neural machine translation. In *AAAI*, 2018.

[16] M. Gutmann and A. Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *AIStats*, 2010.

[17] N. Han, G. Topic, H. Noji, H. Takamura, and Y. Miyao. An empirical analysis of existing systems and datasets toward general simple question answering. In *CoNLL*, 2020.

[18] T. B. Hashimoto, K. Guu, Y. Oren, and P. Liang. A retrieve-and-edit framework for predicting structured outputs. In *Neurips*, 2018.

[19] K. Hassani and A. H. Khasahmadi. Contrastive multi-view representation learning on graphs. In *ICML*, 2020.

[20] G. He, Y. Lan, J. Jiang, W. X. Zhao, and J.-R. Wen. Improving multi-hop knowledge base question answering by learning intermediate supervision signals. In *WSDM*, 2021.

[21] X. Huang, J. Zhang, D. Li, and P. Li. Knowledge graph embedding based question answering. In *WSDM*, 2019.

[22] K. Jiang, D. Wu, and H. Jiang. Freebaseqa: a new factoid qa data set matching trivia-style question-answer pairs with freebase. In *NAACL*, 2019.

[23] V. Karpukhin, B. Oğuz, S. Min, L. Wu, S. Edunov, D. Chen, and W.-t. Yih. Dense passage retrieval for open-domain question answering. In *EMNLP*, 2020.

[24] U. Khandelwal, O. Levy, D. Jurafsky, L. Zettlemoyer, and M. Lewis. Generalization through memorization: Nearest neighbor language models. In *ICLR*, 2020.

[25] U. Khandelwal, A. Fan, D. Jurafsky, L. Zettlemoyer, and M. Lewis. Nearest neighbor machine translation. In *ICLR*, 2021.

[26] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[27] T. Kwiatkowski, E. Choi, Y. Artzi, and L. Zettlemoyer. Scaling semantic parsers with on-the-fly ontology matching. In *EMNLP*, 2013.

[28] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[29] A. Miller, A. Fisch, J. Dodge, A.-H. Karimi, A. Bordes, and J. Weston. Key-value memory networks for directly reading documents. In *EMNLP*, 2016.

[30] S. Mohammed, P. Shi, and J. Lin. Strong baselines for simple question answering over knowledge graphs with and without neural networks. In *NAACL*, 2018.

[31] A. Neelakantan, B. Roth, and A. McCallum. Compositional vector space models for knowledge base completion. In *ACL*, 2015.

[32] J. Qiu, Q. Chen, Y. Dong, J. Zhang, H. Yang, M. Ding, K. Wang, and J. Tang. Gcc: Graph contrastive coding for graph neural network pre-training. In *KDD*, 2020.

[33] Y. Qiu, Y. Wang, X. Jin, and K. Zhang. Stepwise reasoning for multi-relation question answering over knowledge graph with weak supervision. In *WSDM*, 2020.

[34] H. Ren, W. Hu, and J. Leskovec. Query2box: Reasoning over knowledge graphs in vector space using box embeddings. In *ICLR*, 2020.

[35] A. Saxena, A. Tripathi, and P. Talukdar. Improving multi-hop question answering over knowledge graphs using knowledge base embeddings. In *ACL*, 2020.

[36] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 2008.

[37] R. C. Schank. *Dynamic memory: A theory of reminding and learning in computers and people*. cambridge university press, 1982.

[38] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling. Modeling relational data with graph convolutional networks. In *ESWC*, 2018.

[39] L. B. Soares, N. FitzGerald, J. Ling, and T. Kwiatkowski. Matching the blanks: Distributional similarity for relation learning. In *ACL*, 2019.

[40] F.-Y. Sun, J. Hoffmann, V. Verma, and J. Tang. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *ICLR*, 2020.

[41] 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*, 2018.

[42] H. Sun, T. Bedrax-Weiss, and W. W. Cohen. Pullnet: Open domain question answering with iterative retrieval on knowledge bases and text. In *EMNLP*, 2019.

[43] Z. Sun, Z.-H. Deng, J.-Y. Nie, and J. Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. In *ICLR*, 2019.

[44] K. Teru, E. Denis, and W. Hamilton. Inductive relation prediction by subgraph reasoning. In *ICML*, 2020.

[45] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks. In *ICLR*, 2018.

[46] P. Verga, H. Sun, L. B. Soares, and W. W. Cohen. Facts as experts: Adaptable and interpretable neural memory over symbolic knowledge. *arXiv preprint arXiv:2007.00849*, 2020.

[47] P. Wu, S. Huang, R. Weng, Z. Zheng, J. Zhang, X. Yan, and J. Chen. Learning representation mapping for relation detection in knowledge base question answering. In *ACL*, 2019.

[48] W. Xiong, T. Hoang, and W. Y. Wang. Deeppath: A reinforcement learning method for knowledge graph reasoning. In *EMNLP*, 2017.

[49] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? In *ICLR*, 2019.

[50] B. Yang, W.-t. Yih, X. He, J. Gao, and L. Deng. Embedding entities and relations for learning and inference in knowledge bases. In *ICLR*, 2015.

[51] W.-t. Yih, M. Richardson, C. Meek, M.-W. Chang, and J. Suh. The value of semantic parse labeling for knowledge base question answering. In *ACL*, 2016.

[52] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen. Graph contrastive learning with augmentations. *Neurips*, 2020.

[53] M. Yu, W. Yin, K. S. Hasan, C. d. Santos, B. Xiang, and B. Zhou. Improved neural relation detection for knowledge base question answering. In *ACL*, 2017.

[54] J. M. Zelle and R. J. Mooney. Learning to parse database queries using inductive logic programming. In *NCAI*, 1996.

[55] L. Zettlemoyer and M. Collins. Online learning of relaxed ccg grammars for parsing to logical form. In *EMNLP*, 2007.

[56] L. S. Zettlemoyer and M. Collins. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI*, 2005.

[57] M. Zhang and Y. Chen. Link prediction based on graph neural networks. In *Neurips*, 2018.

[58] Y. Zhang, H. Dai, Z. Kozareva, A. J. Smola, and L. Song. Variational reasoning for question answering with knowledge graph. In *AAAI*, 2018.

[59] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, and L. Wang. Deep graph contrastive representation learning. *arXiv preprint arXiv:2006.04131*, 2020.

A Hyperparameters

For MetaQA, we use 3 GCN layers with GCN layer dimension of 32. For training we have used 5 nearest neighbors and 10 are used for evaluation for the 1-hop, 2-hop and 3-hop queries.We optimize the loss using Adam Optimizer with beta1 of 0.9, beta2 of 0.999 and epsilon of 1e-8. As well as the learning rate is set to be 0.00099 with temperature value of 0.0382 (1-hop), 0.0628 (2-hop) ,0.0779 (3-hop). All the models are trained for 5 epochs.

Similarly for WebQSP, we use 3 GCN layers with GCN layer dimension of 32. But for training we used 10 nearest neighbors and 5 are used for evaluation. We optimize the loss using Adam Optimizer with beta1 of 0.9, beta2 of 0.999 and epsilon of 1e-8. As well as a learning rate of 0.0024 and temperature of 0.0645 is used. Though the model is trained for about 30 epochs.

B Generating synthetic data for control experiments

We generate the dataset for running control experiments extending the Erdős-Rényi model [12] for sampling random graphs to heterogeneous graphs (graphs with types edges and/or nodes).

(i) In the first stage, a type system for the KB is created by sampling a fixed set of 16 entity types and edges are added between types with a chance of 0.3. Our sampled KB type system has 74 relation types. This is the exact Erdős-Rényi model.

(ii) Next, given a pattern shape we generate a 'grounded pattern'. The first query entity is selected at random. From there every entity type/relation in the pattern type is sampled from the types allowed by the KB system. For example, given a $2i$ pattern shape, we sample an entity type $t1$ for the first query entity $e1$. Then we assign a type $r1$ to outgoing edge from the allowed outgoing edge types for $t0$. This assigns a type t_ans to the answer node $?ans$. Then we sample a type $r2$ to incoming edge from the allowed incoming edge types for t_ans. This assigns a type $t2$ to the second query entity $e2$. The final 'grounded' $2i$ pattern is then (($e1$, $r1$, $?ans$), ($e2$, $r2$, $?ans$)).

(iii) Next, to sample a query graph, we create an empty graph with 120 entities each randomly assigned a type from the 16 types. The query entities have pre-assigned entity types based on the pattern type ($e1$ and $e2$ from the previous example are fixed to be type $t1$ and $t2$ respectively) . Starting from the query entities, we sample edges allowed by the KB type system. We add an edge between two entities with a chance of 0.4. We ensure that the entities in the subgraph are at most a distance of 3-hops from the query entities.

(iv) Finally, we execute the pattern on the graph to assign labels to answer nodes.

Our control dataset samples 200 pattern types and 15 graphs per pattern type distributing them equally (i.e. 5 each) between train, validation and test. For each of the 15 graphs that share a common pattern type, we assign the 5 graphs that were put in the train set as the kNN queries.

C Dataset details and statistics

Table 6 summarizes the basic statistics of the datasets used in our experiments.

Dataset	Train	Dev	Test
MetaQA 1-hop	96,106	9,992	9,947
MetaQA 2-hop	118,980	14,872	14,872
MetaQA 3-hop	114,196	14,274	14,274
WebQSP	2,848	250	1,639
FreebaseQA	20,358	2308	3996

Table 6: Dataset Statistics

D Retrieving cases by masking query entities

	KNN for Unmasked Query	KNN for Masked Query
Query	what did **james k polk** do before he was president	what did [MASK] do before he was president
Retrieved kNN	1. what did **james k polk** believe in 2. what did barack obama do before he took office	1. what did abraham lincoln do before he was president 2. what did barack obama do before he took office
Query	what are the songs that **justin bieber** wrote	what are the songs that [MASK] wrote
Retrieved kNN	1. what is the name of **justin bieber** brother 2. what are all the inventions benjamin franklin made 3. what are all the movies channing tatum has been in	1. what are all the songs nicki minaj is in 2. what songs did mozart write 3. what songs did richard marx write
Query	where did edgar allan poe died	where did [MASK] died
Retrieved kNN	1. what college did **edgar allan poe** go to 2. what magazine did **edgar allan poe** work for 3. what year did **edgar allan poe** go to college	1. where did mendeleev died 2. where did benjamin franklin died 3. where did thomas jefferson died

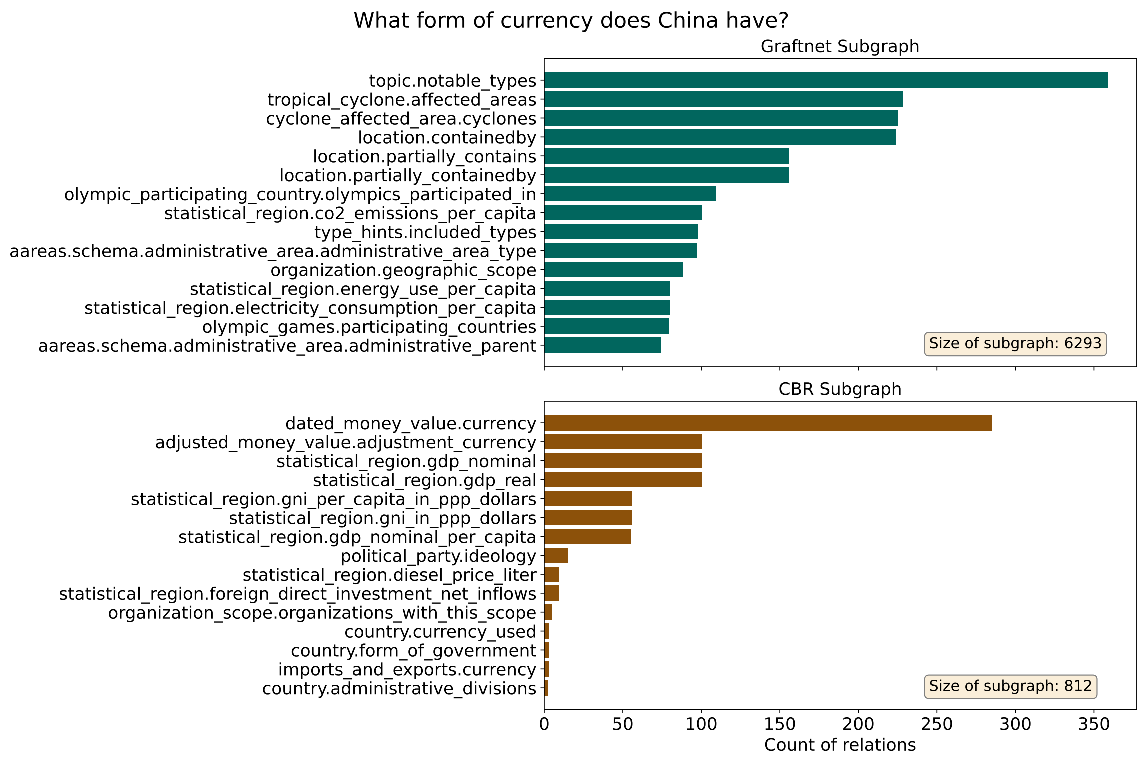
Table 7: Retrieval by masking the question entity prevents the returned kNN queries from focusing on the entity and instead rely on the question structure and relation involved.

E Adaptive subgraph collection tailors subgraphs to the query

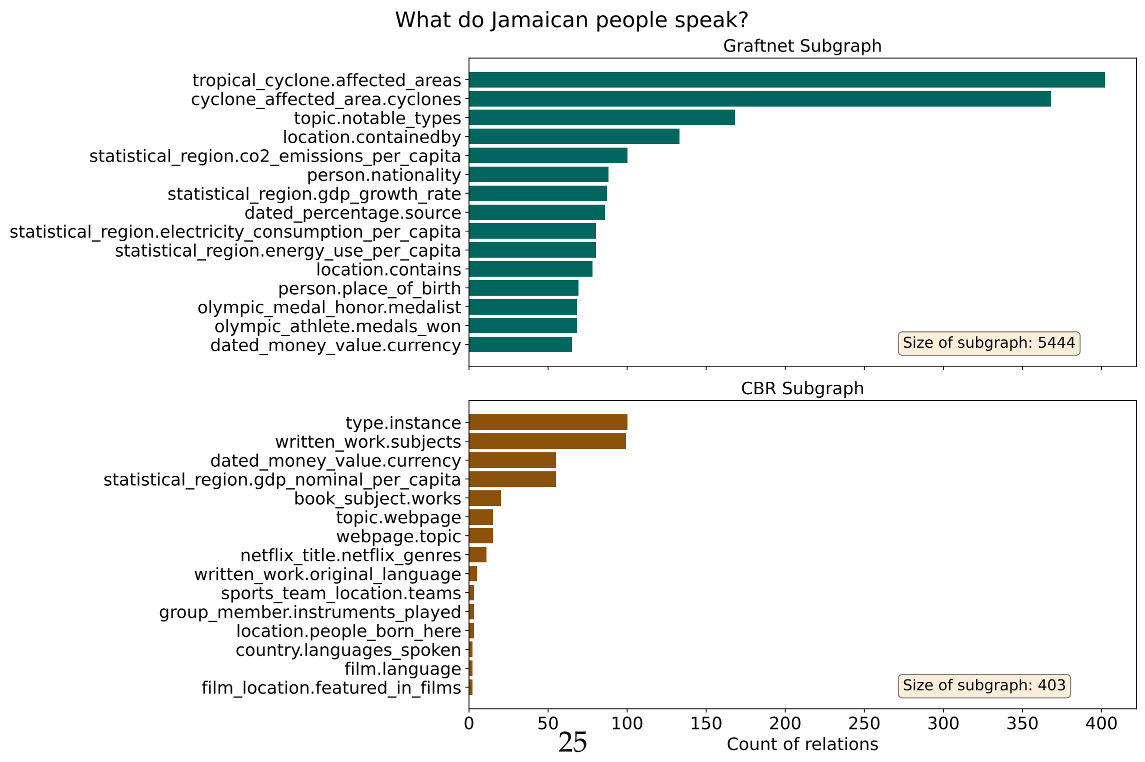
Figure 7, 8 and 9 shows example of few query-subgraphs collected by GraftNet and our adaptive subgraph collection strategy (§3.2). Each figure plots the most frequent (top 15) relations gathered by each subgraph collection procedure. It should be clear that our adaptive subgraph collection strategy gathers relations that are more specific to the query and also leads to more compact subgraph.

F Further Related Work

CBR-SUBG shares similarities with the RETRIEVE-AND-EDIT framework [18] which utilizes retrieved nearest neighbor for structured prediction. However, unlike our method they only retrieve a single nearest neighbor and will unlikely be able to generate programs for questions requiring relations from multiple nearest neighbors. There has also been a lot of recent work in general NLP which uses KNN-based approaches. For example, Khandelwal et al. [24] demonstrate improvements in language modeling by utilizing explicit examples from training data. There has been work in machine translation [15, 25] that uses nearest neighbor translation pair to guide the decoding process.

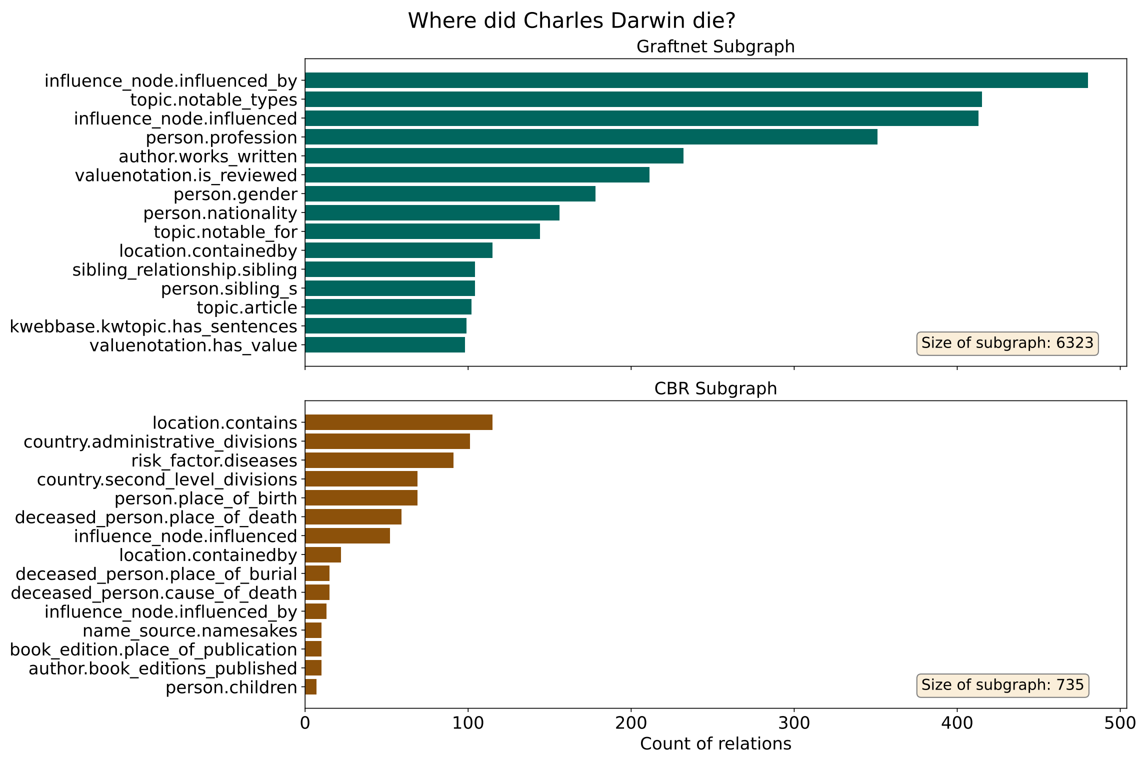


(a)

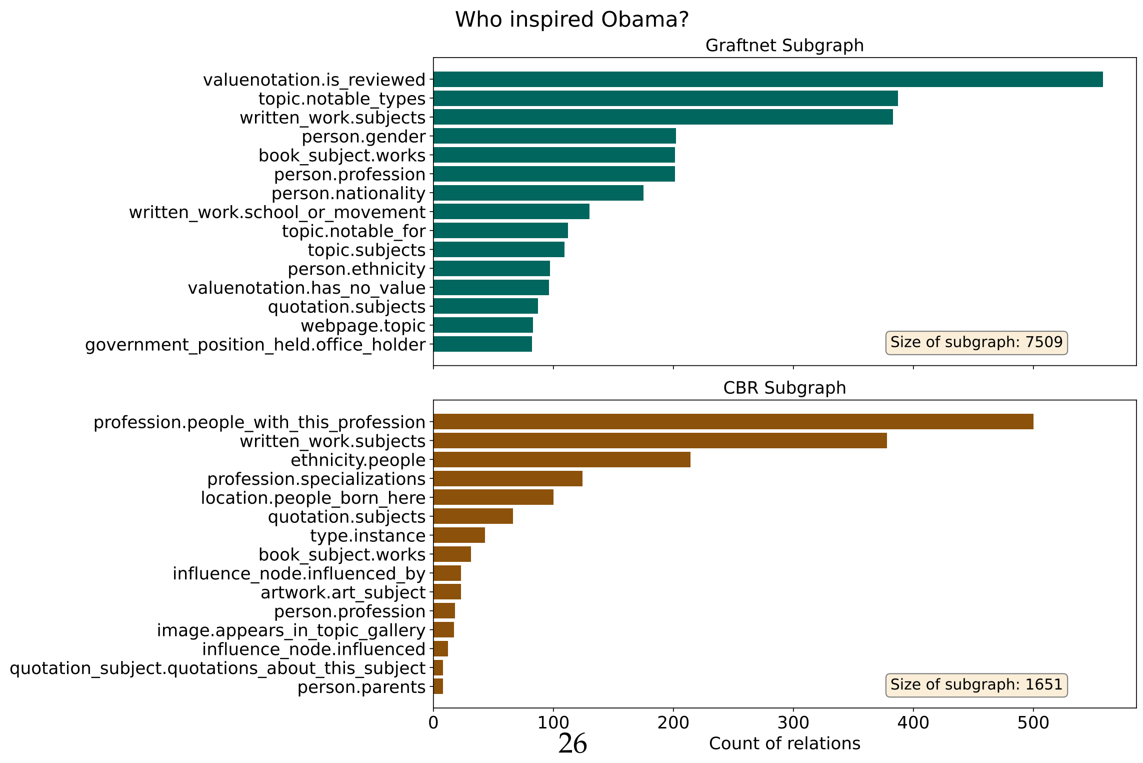


(b)

Figure 7: Figure shows the count of most frequent relations present in the subgraph collected by Graftnet and by our adaptive subgraph collection strategy for the query shown in the title of each figure. The CBR-subgraph has relations which are more relevant for the given query whereas the Graftnet subgraph has more generic relations. The total number of edges in the CBR-subgraph is also less than the Graftnet subgraph

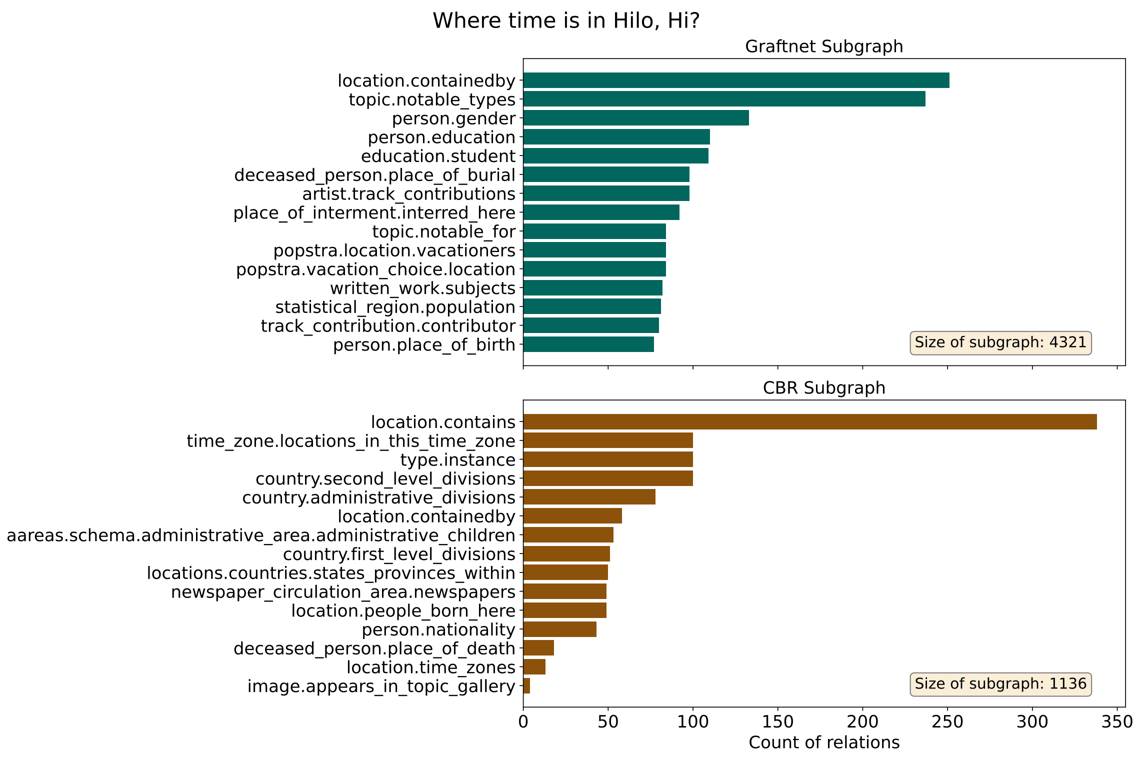


(a)

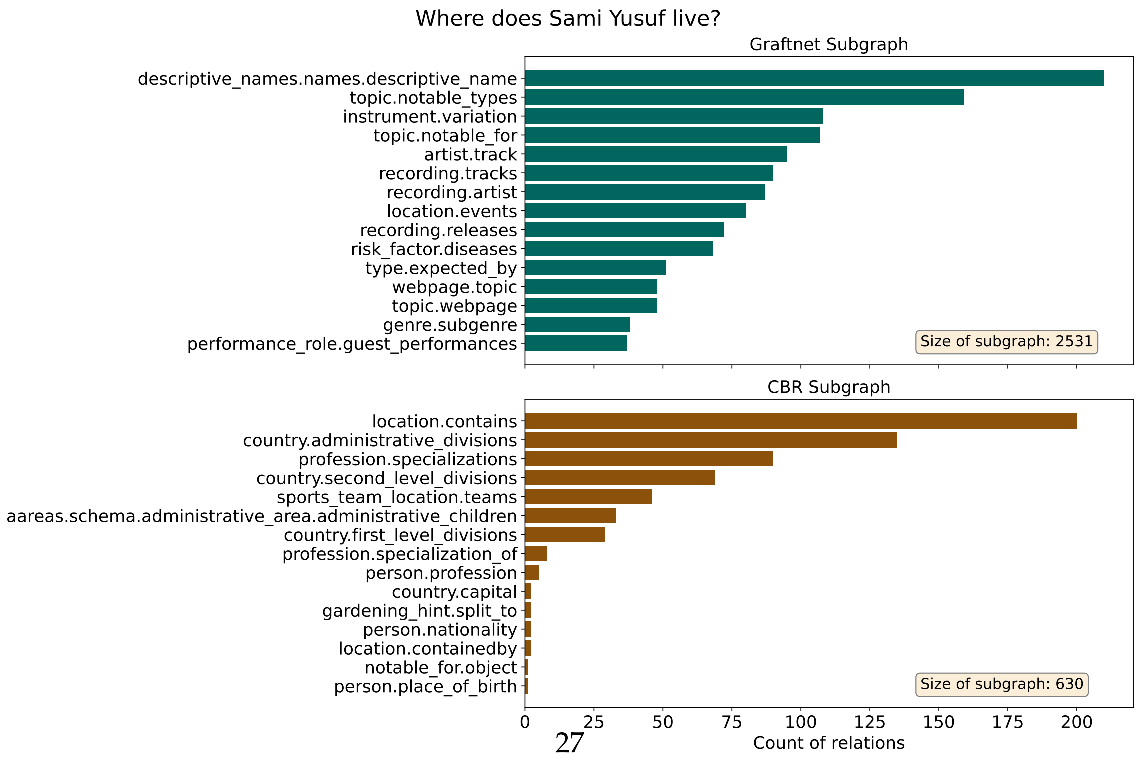


(b)

Figure 8: Figure shows the count of most frequent relations present in the subgraph collected by Graftnet and by our adaptive subgraph collection strategy for the query shown in the title of each figure. The CBR-subgraph has relations which are more relevant for the given query whereas the Graftnet subgraph has more generic relations. The total number of edges in the CBR-subgraph is also less than the Graftnet subgraph



(a)



(b)

Figure 9: Figure shows the count of most frequent relations present in the subgraph collected by Graftnet and by our adaptive subgraph collection strategy for the query shown in the title of each figure. The CBR-subgraph has relations which are more relevant for the given query whereas the Graftnet subgraph has more generic relations. The total number of edges in the CBR-subgraph is also less than the Graftnet subgraph