XQA: Relationship Explanation in Knowledge Graphs via Graph-Augmented Neural Networks

Anonymous Author(s)

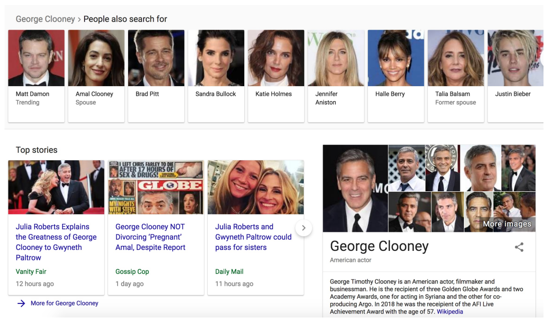
ABSTRACT

Knowledge graphs capturing heterogeneous relationships between different type of entities are used to support search capabilities for a wide variety of domains such as general web search, social media, healthcare and business intelligence. Explaining the relationship between entities that appear as anomalies or are found to be statistically correlated in the data is an important task. For instance, a doctor will want to understand the relationship between a symptom and a disease, and a financial analyst will seek to explain the association of a company and supply-chain materials. The problem of finding interesting paths that explain the relationship between two entities has been studied in the literature. However, finding "interesting" paths as an explanation of relationship is highly subjective and the challenge is amplified as the number of entity and relation types grow. We propose a novel algorithm named XQA (eXplanatory QA) that integrates structured knowledge from graphs to augment the memory of a LSTM network and performs neural network-guided graph search. We perform user studies on Amazon Turk for capturing explanation preferences for entity pairs in multiple prominent KBs, and formulate metrics that capture properties of what humans consider as good answers. XQA outperforms a reinforcement learning based state-of-the-art system by finding 21% more answers on 1-hop relationship queries for NELL, and 90X more for longer-hop queries on NELL and Wikispeedia dataset. Empirical evaluation demonstrates the capability of XQA to find intuitive answers that outperform human subjects in many cases.

ACM Reference Format:
Anonymous Author(s). 2018. XQA: Relationship Explanation in Knowledge Graphs via Graph-Augmented Neural Networks. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Knowledge graphs have gained wide acknowledgement as a repository for structured knowledge derived from multi-source data. Starting with generic knowledge bases such as Freebase, YAGO etc., there has been a proliferation of domain specific knowledge graphs (LinkedIn Economic Graph, SNOMED). Knowledge graphs have enabled applications such as web search and digital assistants (such as Alexa, Cortana, Google Now, Siri) for diverse tasks such as question-answering, recommendation and summarization.



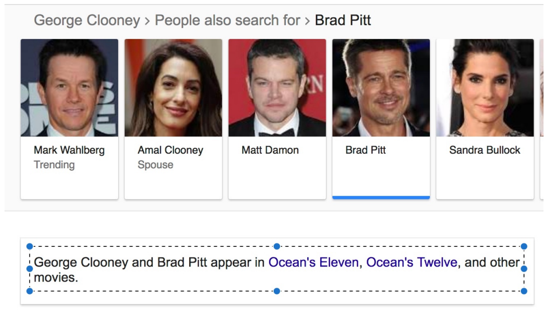
Julia Roberts Explains the Greatness of George Clooney to Gwyneth Paltrow

George Clooney NOT Divorcing 'Pregnant' Amal, Despite Report

Julia Roberts and Gwyneth Paltrow could pass for sisters

George Timothy Clooney is an American actor, filmmaker and businessman. He is the recipient of three Golden Globe Awards and two Academy Awards, one for acting in Syriana and the other for co-producing Argo. In 2018 he was the receipient of the AFI Live Achievement Award with the age of 57. Wikipedia

(a) A search for George Clooney on Google lists related entities shown on top line.



George Clooney and Brad Pitt appear in Ocean's Eleven, Ocean's Twelve, and other movies.

(b) Clicking on some of the related entities such as Brad Pitt shows an explanation of their relationship.

Figure 1: Motivation for web search: example of limited relationship explanation to improve contextual relevance of shown results

Question-answering (QA) on knowledge graphs (KBs) has progressed from answering factoid queries [8, 17] to more complex tasks, such as reasoning over long paths [4, 5]. We focus on the **Relationship Explanation task** [10, 21]: given a KB [1] G_{kb} and a pair of entities (v_s, v_t) that are nodes in the graph, return a set of k-paths as an explanation of the relationship between v_s and v_t. Figure 1 shows how simple relationship explanation algorithms are used by the Google search engine to provide contextual relevance for results.

1.1 Use cases for Domain Specific Relationship Explanation

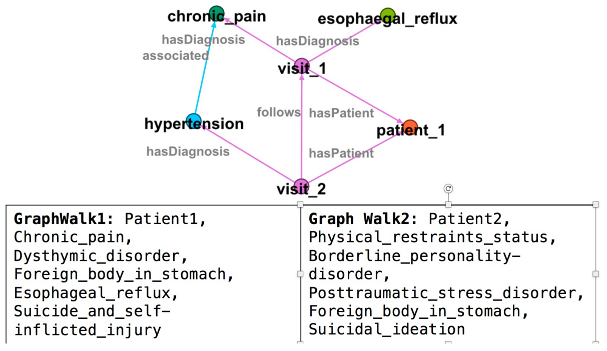
Application of relationship explanation into web search encourages us to explore its applications further into novel settings. We present two use cases illustrated in Figure 2. For both examples, we select queries as two entities in the graph, and present candidate answers as paths connecting the entities through the graph.

Use case 1: Business Intelligence: Financial analysts constantly search the news stream for upcoming product releases or emerging

[1] We use the terms knowledge base (KB) and knowledge graph (KG) interchangeably in this paper.

1) **GraphWalk1**: *Ford isA CarManufacturer, CarManufacturer make Cars, Cars contain CatalyticConvertors, CatalyticConvertors uses Palladium.*
2) **GraphWalk2**: *Ford isA Manufacturer, Manufacturer make Products, Products use RareEarthMineral, Palladium isA RareEarthMineral.*
3) **GraphWalk3**: *Ford isHeadquarteredIn USA, USA hasDiplomaticRelation with Russia, Russia supplies Palladium.*

(a) Can a personalized search assistant reason about how two entities trending in the news are related and notify the user?



```
GraphWalk1: Patient1,
Chronic_pain,
Dysthymic_disorder,
Foreign_body_in_stomach,
Esophageal_reflux,
Suicide_and_self-
inflicted_injury
```

```
Graph Walk2: Patient2,
Physical_restraints_status,
Borderline_personality-
disorder,
Posttraumatic_stress_disorder,
Foreign_body_in_stomach,
Suicidal_ideation
```

(b) Can we explain a patient's probable trajectory by finding evolution of similar patients in the past?

Figure 2: Use cases for complex relationship explanation.

trends. However, the analysts prefer to understand the underlying context between the entities for the alert to be actionable. Given a report of Ford importing high volume of Palladium, Figure 2a shows different explanation candidates retrieved from a KB integrated from YAGO3 and triples extracted from Wall Street Journal articles.

Use case 2: Explaining recommendation to patients: This use case requires caregivers to spot the transition of a patient from current state to a critical state in future, such as suicide ideation. Finding diagnosis code trajectories of past patients that show how a patient suffering from extreme pain may evolve towards suicide ideation is seen as a promising way to communicate recommendations to patients or their families. Figure 2b shows a knowledge graph extracted from electronic health records from the MIMIC-III dataset [18] with nodes of type patient, visit-event, and multiple diagnosis groups. The bottom row shows two candidate paths in the graph showing the temporal evolution of a patient's diagnosis.

1.2 Challenges for State-of-the-Art

State-of-the-art algorithms answer relationship explanation queries (or simply, relationship queries) [2] by learning diverse patterns of answer paths. The performance of such symbolic representation-based approaches are strongly tied to the quality of rules [12] or frequent subgraph patterns [38] mined from the knowledge graph. As the graphs grow big and heterogeneous with large number of entity and relation types, it introduces a number of challenges discussed below. Also, evaluation of answer quality becomes highly subjective with such information explosion, and demands the query algorithms incorporate human preferences for answers depending on the context of queried entities.

[2]we use the terms "relationship query", "relationship explanation" and "explanatory query" interchangeably

Addressing heterogeneity in KBs Many KBs have nodes labeled with coarse grained types. As an example, consider KBs built from off-the-shelf information extraction methods [7] where most entities are grouped into coarse types such as person, organization etc. Such issues are acknowledged in recent literature [20, 29] and lead to overgeneralization of patterns or other structural properties derived using node labels. Another issue is of dealing with synonymous relations. Populating a KB with a large number of unique relations represented by semantically close verb phrases leads to learning a larger set of closely related rules with lower support for each rule.

Accounting for Human Preferences Relationship queries do not always need a deductive explanation, derived from a pattern. As Figure 2b suggests, some applications expect a trail of entities leading from one entity (or context) to another. The implicit assumption is that user posses the domain knowledge to interpret the "chain of entities" to execute a "chain of reasoning". The transitions from one entity to the next one is strongly tied to local context. Thus, in addition to the the labels of the involved entities, their characteristics as revealed by individual connectivity or topological properties alter the user's preference. Such explanations are characterized by collectively sharing strong logical consistency. We refer to such paths as "coherent paths", or more generally, "coherent subgraphs".

1.3 Approach and Contributions

The aforementioned issues are not new. In fact, they are extensively discussed and addressed for related topics such as knowledge base completion (see related work, section 6). Our input and output representation is symbolic. Therefore, even with embracing vector space methods, we need to find a way to come back to the symbolic representation. This motivates us to develop a search algorithm using a "graph-augmented LSTM" (section 4).

The graph-augmented LSTM guides the graph walk for relationship query execution. Unlike seq2seq-based approaches that would only use embeddings of nodes encountered during a graph walk, we dynamically select a subset of neighboring edges from every walked node in the graph and use their composite vector representation as input to the model. Effectively, the model not only remembers the traversed nodes, but also explicitly accounts for the graph structure around them. We refer to this process as "Graph-based Memory Augmentation", and present extensive results with multiple selection strategies.

The body of the paper begins with our other major contribution: extensive user studies on relationship queries across multiple datasets. REX [10], an early inspiration for our work, proposed a number of interestingness metrics and presented with user studies for 5 questions answered by 10 users. In contrast, we present extensive user studies conducted in Amazon Turk over 250 questions for two prominent KBs, with each question being rated by 10 unique users (section 3).

We define a set of metrics to quantitatively model the highly subjective notion of coherent explanatory paths. We introduce a "path divergence" metric that is computed from vector-space embeddings. Coherence and path divergence are inversely related. The path divergence measure accounts for both individual entity characteristics as well as their semantic relationship. Our user studies indicate that path length, path average degree and path divergence capture salient characteristics of preferred explanatory path across the datasets. The first two metrics are in agreement with past studies, formulation of path divergence is a contribution from this work.

Our experiments (section 5) demonstrate robust performance from the model-driven graph walk. It consistently maintains answer quality even where human answer quality degrades, and shows remarkable generalization ability to generate higher quality answers than human answers on number of occasions. We conduct experiments on real-world KBs such as Wikispeedia and NELL, and demonstrate that XQA outperforms MINERVA [4], a reinforcement learning based state of the art method by 21% and 90x for two different query classes as measured in terms of successful answers.

2 FUNDAMENTALS

2.1 Heterogeneous Graph

Notation	Description
G	Knowledge graph described as a ordered tuple $(V, E, \Sigma_V, \Sigma_E, \lambda_v, \lambda_e)$
$V(G)$	Node set of graph G
$E(G)$	Edge set of graph G
$\Sigma_V(G)$	Set of node labels in G
$\Sigma_E(G)$	Set of edge labels G
λ_v	Labeling function that maps each node $v \in V$ to set of label $l_v \in \Sigma_V$
λ_e	Labeling function that maps each edge $e \in E$ to a label $l_e \in \Sigma_E$
G_{fp}	Set of frequent subgraph patterns of G
$\Gamma(v)$	Embedding function mapping v to $\mathbb{R}^d$
$N(v)$	Neighborhood of node v
$\Gamma_N(v)$	Embedding function mapping neighborhood of v to $\mathbb{R}^d$
$Q = (v_s, v_t)$	Query expressed as a source (v_s) and target node (v_t)
P_{st}	Path connecting nodes v_s and v_t
$\phi_{div}(P)$	Divergence of path P
$\phi_l(P)$	Length of path P
$\phi_{sp}(P)$	Specificity of path P
L_{max}	Maximum length of answer paths
b	Branching factor used by search algorithm

DEFINITION: KNOWLEDGE GRAPH We represent a graph-based KB as a 6-tuple $G_{kb} = (V, E, \Sigma_V, \Sigma_E, \lambda_v, \lambda_e)$ where V is the set of nodes and E denotes the set of relationships between the nodes. Σ_V and Σ_E are functions mapping each node (or edge) into a specific set of node (or edge) classes or types.

DEFINITION: SUBGRAPH A graph g is a subgraph of G if $V(g) \subseteq V(G)$ and $E(g) \subseteq E(G)$

DEFINITION: PATH PATTERN $P = (v_0^L, e_0^L, v_1^L)$, (v_1^L, e_1^L, v_2^L), .., $(v_{k-1}^L, e_{k-1}^L, v_k^L)$, expressed as a sequence of triples is a path pattern from G if all node labels $v_i^L \in \Sigma_V(G)$ and all edge labels $e_i^L \in \Sigma_E(G)$.

DEFINITION: PATH PATTERN INSTANCE $P_{inst} = (v_0, e_0, v_1)$, (v_1, e_1, v_2), .., (v_{k-1}, e_{k-1}, v_k), expressed as a sequence of triples is a path pattern instance from G if for all nodes in P_{inst}, $\lambda_v(v_i) \in \Sigma_V(G)$ and all edges $\lambda_e(e_i) \in \Sigma_E(G)$.

2.2 Problem Definition

We introduce a variant of the original relationship explanation problem: Given a heterogeneous graph G, and a query expressed as a pair of nodes (v_s, v_t), report k-paths with highest ranking by a user defined function ϕ.

2.3 Long Short-Term Memory Network

Long short-term memory networks (LSTM) are a specialization of Recurrent Neural Networks (RNN) [14]. A recurrent neural network is defined by repeating a simple structure, where the input x_t is transformed to the output y_t through a non-linear activation function such as $y_t = tanh(Wx_t + b)$. W is the weight matrix learnt from the data and the b represents a bias vector. A LSTM controls the influence of the prior iteration through the introduction of four gates as described by the equations below.

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{1}$$

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{2}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{3}$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot \tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{4}$$

$$h_t = o_t \cdot \tanh(c_t) \tag{5}$$

2.4 Vector Representation of Nodes

We map each node in a graph G to a low-dimensional space $\mathbb{R}^d$ such that geometric relationships in the metric space reflects the structural properties of the graph. Generally, given a graph $G = (V, E)$ and optionally, matrices capturing node (and edge attributes) $A_v \in \mathbb{R}^{|V| \times N_{va}}$ $(A_e \in \mathbb{R}^{|E| \times N_{ea}})$, such methods seek to map each node $u \in V(G)$ to real-valued vector in a low-dimensional space d such that $\Gamma(u) \in \mathbb{R}^d$ where $d << |V(G)|$.

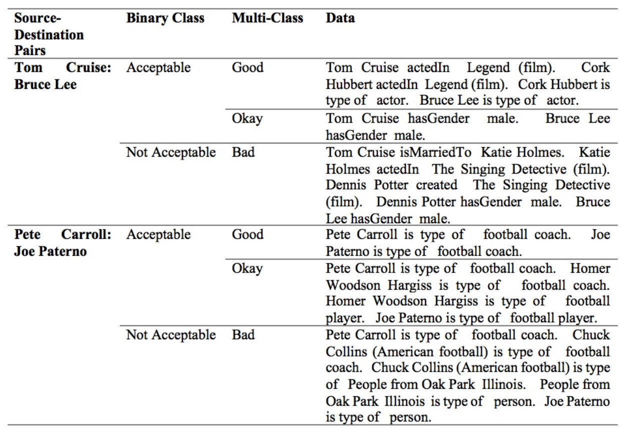
Representation learning algorithms for graphs can be broadly separated into two groups based on their reliance on matrix factorization versus random walks. Given a graph G, matrix factorization based methods such as GraRep [3] and [1, 27] seek to learn a representation Γ that minimizes a loss function of the form $||\Gamma^T \Gamma - P_V||^2$, where P_V is a matrix containing pairwise proximity measures for $V(G)$. Random walk based methods such as DeepWalk [28] and node2vec [15] try to learn representations that roughly minimize a cross-entry loss function of the form $\sum_{v_i, v_j \in V(G)} -log(p_L(v_j|v_i))$, where $p_T(v_j|v_i)$ is the probability of visiting a node v_j on a random walk of length L starting from node v_i. Node2vec like algorithms are further extended by metapath2vec [6] to incorporate multi-relational properties by constraining random walks.

3 CHARACTERISTICS OF USER PREFERRED EXPLANATIONS OVER LONG PATHS

The use cases in section 1 motivates us to experimentally study characteristics of human preferred explanations. We perform user studies on two datasets well accepted for their relevance in knowledge graph research as well as studies for human pathfinding on the web: Freebase and Wikispeedia.

3.1 User study design

For each dataset, we generated question entity pairs and their candidate answer paths using the following steps. First, we divided entities into node categories and randomly sampled up to 100 nodes (if available) from each category. For all possible entity pairs, we selected those whose shortest path ranges from 2-4 hops. Next, we generated random walks between entity pairs and calculate path feature metrics (discussed later in this section). Finally, we sampled 200 entity pairs that provide an uniform coverage across feature space. We present each QA pair to users in Amazon Mechanical Turk and ask the users to rate the answers as “good", “'ok" or “bad" based on their relevance and intuitive appeal. Figure 3) provides an example of questions and answer formats.



Source-Destination Pairs	**Binary Class**	**Multi-Class**	**Data**
Tom Cruise: Bruce Lee	Acceptable	Good	Tom Cruise actedIn Legend (film). Cork Hubbert actedIn Legend (film). Cork Hubbert is type of actor. Bruce Lee is type of actor.
		Okay	Tom Cruise hasGender male. Bruce Lee hasGender male.
	Not Acceptable	Bad	Tom Cruise isMarriedTo Katie Holmes. Katie Holmes actedIn The Singing Detective (film). Dennis Potter created The Singing Detective (film). Dennis Potter hasGender male. Bruce Lee hasGender male.
Pete Carroll: Joe Paterno	Acceptable	Good	Pete Carroll is type of football coach. Joe Paterno is type of football coach.
		Okay	Pete Carroll is type of football coach. Homer Woodson Hargiss is type of football coach. Homer Woodson Hargiss is type of football player. Joe Paterno is type of football player.
	Not Acceptable	Bad	Pete Carroll is type of football coach. Chuck Collins (American football) is type of football coach. Chuck Collins (American football) is type of People from Oak Park Illinois. People from Oak Park Illinois is type of person. Joe Paterno is type of person.

Figure 3: Example of question-answer pairs presented to Amazon Turk users for rating answers.

3.2 Qualitative Analysis of User Study Results

We performed extensive analysis to understand the features that distinguish between the set of “Good/Ok” vs “Bad” answers. One of the primary challenge in the studies turned to be the lack of “Good” answers available from the randomly generated paths between entities. In hindsight, this reinforces the notion that generating random paths between entity pairs is not sufficient to generate meaningful explanations. Following are some observations about defining characteristics of good answers. Table 1 shows 18 freebase paths rated as “Good” by users.

Most good paths go through list of minimal entities that seem to be logically moving from “source topic” to “destination topic”. We use the term “topic” loosely to describe entity context. When source and destination are connected through a short 1-hop common entity (example 1, 7, 8, 9, 12) the answer is generally rated as *good* explanation. For longer paths the transitions are gradual from source to destination topic, example (2,4,10,14,16).

The *bad* paths also stand out in few ways. They generally have very generic nodes in the path. For example , in Freebase 'United_States' was the most frequently occurring node in all paths (481 answer paths contained it) and 96% (466/481) of those paths were marked as *bad* explanations. In wikispeedia we observed similar trend, 'United States' was the most frequently occurring node and 95% (22/23) explanations containing it were rejected as 'Bad'.

1) edgar_allan_poe, writer, william_woodruff
2) alan_turing, mathematician, edmond_halley, astronomer, nicolaus_copernicus
3) benjamin_franklin, statesman, winston_churchill
4) marvin_gaye, record_producer, frank_zappa;musician, jim_morrison
5) john_f_kennedy, assassination_by_firearm, mahatma_gandhi, assassination, julius_caesar
6) robert_oppenheimer, harvard_university, ted_kennedy
7) benjamin_franklin, politician, andrew_jackson
8) galileo_galilei, italian_people, leonardo_da_vinci
9) alan_turing, mathematician, nicolaus_copernicus
10) mathematician, edmond_halley, physicist, linus_pauling
11) university_of_california_berkeley, leonard_j_fick, ohio_state_university, edward_g_breen, politician, woodrow_wilson
12) thomas_jefferson, lawyer, woodrow_wilson
13) michael_jackson, songwriter, clifton_chenier, musician
14) budapest, gyorgy_faludy, writer, william_woodruff
15) vladimir_i_lenin, politician, woodrow_wilson
16) hollywood, guy_oliver, actor, marilyn_monroe
17) mahatma_gandhi, peace_activist, martin_luther_king_jr
18) alfred_hitchcock, united_states, steve_gaines, guitarist, johnny_cash

Table 1: “Good” explanations in Freebase (rated by Amazon Turk users >= 50%)

In closing, we summarize the above discussions with the following aspects:

1) Need for specificity In accordance with prior findings, Users prefer answer paths that go through prominent entities. Entities that are prominent and well-connected in the graph to a large space of entities through diverse relations are discouraged as it dilutes the answer specificity. The degree of a node serves as a simple and useful indicator for this purpose.

2) Conceptual overlap Each edge transition must be guided by a strong logical connection across involved edge types and participating entities. However, the entire subgraph should be coherent. Simply ensuring pairwise affinity through the chain is insufficient.

3) Coverage of topics Coverage of topics binding both entities is critical. Later in this section, we propose a new metric called “Subgraph Divergence" that captures both “conceptual overlap" as well as “coverage" aspects discussed above. A lower divergence will indicate higher coherence for the answer path.

3.3 Path Divergence Metric

Guided by observations from user study, we define the path divergence metric to capture the semantic variability in entity 'topics' for a given explanation. Let $\Gamma(v)$ be the function mapping v to its semantic representation in vector space $\mathbb{R}^d$. Equating vector space dimensions to dimensions in a topic space, we focus on seeking tightness around a few selected dimensions denoted by n_T. Typically, n_T would be much smaller than the dimensionality of the embedding space $\mathbb{R}^d$. Hence, we identify salient topics n_T for a path P given by $v_1, v_2, \cdots, v_k$, by the maximally weighed dimensions of $\bar{\Gamma}_v$ where $\bar{\Gamma}_v = \sum_{i=1}^{i=k} \frac{\Gamma(v_i)}{k}$ represents the centroid of all embeddings of nodes

Metrics	Freebase	Wikispeedia
Number of (Questions, Answers)	204,1020	46, 230
Number of Answers rated good (more than 50%)	18	37
Number of Answers rated ok(more than 50%)	102	18
Number of Answers rated bad (more than 50%)	816	125
Average Path Length (Good+Ok, Bad)	4.80, 5.417	4.814, 5.362
Average Path Divergence (Good+Ok, Bad)	0.047, 0.06	0.012, 0.004
Average Path Degree (Good/Ok, Bad)	5285.05, 13580.15	353.92, 653.95
SVM Accuracy Path Length (Precision, Recall, F1-Score)	0.76, 0.87, 0.82	0.43, 0.61, 0.51
SVM Accuracy Path Average Divergence (Precision, Recall, F1-Score)	0.76, 0.87, 0.82	0.44, 0.67, 0.53
SVM Accuracy Path Degree (Precision, Recall, F1-Score)	0.72, 0.84 , 0.77	0.84, 0.83, 0.82
SVM Accuracy for Path Length , Divergence and Degree (Precision, Recall, F1-Score)	0.86, 0.87, 0.84	0.69, 0.72 , 0.70

Table 2: Summary of User Study Results on Amazon Turk : Each (question, answer) pair is rated Good,Ok, Bad or 'Other' by 10 unique users. The average feature set for 'Good/Ok' explanations tends to be 'short path length, low topic divergence, low average degree compared to 'Bad' explanations. The proposed metrics have classification accuracy of more than 80% across the datasets

in the path P. We further calculate the divergence of path δ_P as the maximum divergence in the most coherent topic among n_T topics for a path P given by $v_1, v_2, \cdots, v_k$:

$$\delta_P = \min_{j \in n_T} \{ \max_{i \in k-1} \{ ||\Gamma(v_i)[j] - \Gamma(v_{i+1})[j]|| \} \} \tag{6}$$

3.4 Feature Analysis

Hence we propose a combination of Path Length, Topic Divergence and Path Average Degree to be the set of features deterministic of user preferred answers (Table 2). We present average values of these features for good and bad paths across datasets for informational purposes, and present Support Vector Classifier (SVC) analysis using "linear" and "rbf" kernels to evaluate the efficacy of the proposed features.

The metrics are reported on best performing kernel for each dataset. The SVC results indicates that our proposed metrics for answer quality capture the distinctive features of "good" and "bad" answers well, and better than length or divergence or degree alone. We achieve best precision ,recall, f1-score in freebase using all features (0.86, 0.87 , 0.84). In wikispeedia the average degree was the best deterministic feature (0.84, 0.83, 0.82) performing better than the combined set. In general, the SVC study is highly skewed towards predicting 'bad' answers and recall for 'good' answers was low due to limited samples. Hence we refer to the reader on average feature values to determine best performing features as a combination of (low path length, low divergence, low average degree).

4 ALGORITHMS FOR MODEL DRIVEN RELATIONSHIP QUERY

This section describes algorithms and related data structures for relationship queries. We begin with describing a simple beam search based approach for finding top-k ranked paths, and then extend to a heuristic based approach utilizing both graph structure and a neural network model to search the graph. We describe the key parameters and loss function for training the model, followed by integration of the model into the beam search algorithm.

4.1 Notes on complexity

A naive way to solve the problem is to find all paths connecting v_s and v_t, and then rank them by ϕ_{div}, ϕ_l or ϕ_{sp} (or more generally, referred to as $\phi(P)$). However, this is impractical as the number of paths with length L_{max} grows exponentially. Given G, and a pair of query vertices v_s and v_t, the related problem of finding top-k lightweight paths corresponding to a query pattern P_q has been shown to be NP-complete [21]. For the current problem, while paths connecting v_s and v_t in the graph can be enumerated in polynomial time, the complexity arises from the nature of the ranking function.

OBSERVATION 1. Ranking function $\phi(P)$ is neither additive nor monotonic.

Consider two paths as node sequences. $P_1 = (v_1, v_2, v_3)$, and $P_2 = P_1 \cup (v_4)$. $\phi(P_2)$ can be either greater or lower than $\phi(P_1)$. If v_4 is contextually relevant to v_1, v_2, v_3, then P_2 may be more coherent than P_1 alone, and $\phi(P_2) < \phi(P_1)$. On the other hand, introduction of a completely random v_4 can cause $\phi(P_2) \geq \phi(P_1)$.

Another practical issue is that k is typically one or multiple orders of magnitude smaller than the number of paths connecting a query pair. If two query nodes are 3-4 hops apart in the graph, and the number of candidate paths is large, application level constraints typically restrict us to showing 3-5 answers to the end user. This motivates us to consider a beam search based algorithm that explores the search space in a breadth-first-search (BFS) fashion. With each hop, we maintain a search frontier consisting of nodes to visit, as well as track the b most promising paths to expand in the next hop. This is discussed in detail in section 4.5.

4.2 Integrating Graph and Neural Model

This section describes two major tasks: 1) learning a model M that given a pair of source and target nodes (v_s, v_t), and a path prefix starting with the source node, predicts the next graph node to visit, and 2) given the query node pair and M, generating the explanatory answer path using G_{kb} and M.

4.3 Model Learning

Given an example path from node v_s to v_t, via v_1, v_2, v_3, we seek to learn a model about the patterns involving the path prefix and the

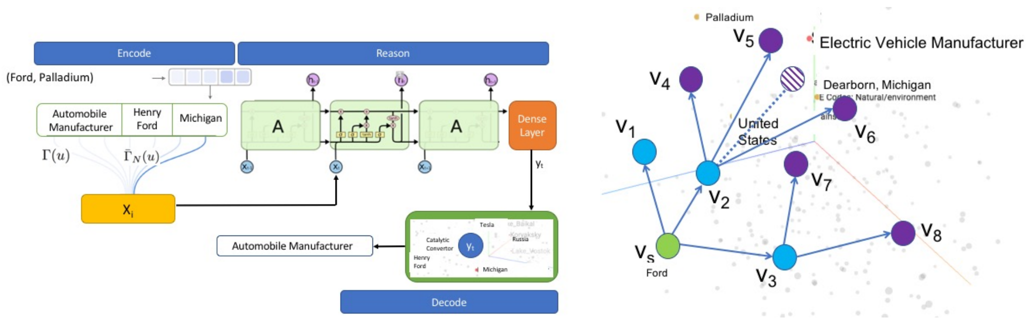


Figure 4: (a) Given an intermediate node v_i, $(v_i, v_t, N(v_i))$ serve as input to query encoder Q. The decoder maps y_i to the nearest neighbor of v_i. (b) The graph search process is illustrated by overlaying an example graph on the embedding space. Nodes that are identical hops away from v_s are marked in same color. The node with dashed lines indicate a point in vector space that may be predicted by the LSTM, however the graph walk will resume by selecting the nearest node (v_5).

destination. In other words, we seek to pass the information that a path explaining v_s's relationship with v_t begins with v_1. Formally, we adapt a sliding window approach and extract a training sequence of v_s, v_1, v_t. Next, we slide the window and generate another training sequence v_1, v_2, v_t. However, we recognize that just using the node-level embedding is not sufficient to learn an accurate model. The embedding information learnt through node2vec-like embedding learning approaches [15] can provide stronger guarantees about a node's membership in a group (and placing similar nodes closer), but it may not succeed in preserving all node-level features, such as structural patterns it may participate in.

We compensate, or guard against such shortcoming by encoding the neighborhood information of walked nodes into the learning process itself. Following the Memory Network architecture [33], we design the network underlying XQA with three components: 1) **Q**: a query encoder, 2) **R**: a graph-based reasoning component and 3) **D**: answer decoder that returns graph nodes.

Q is responsible for translating discrete inputs such as v_s and v_t to their continuous representations. Every graph node u is encoded as a concatenation of two vectors: $\Gamma(u)$ and $\bar{\Gamma}_N(u)$. $\Gamma(u)$ is u's embeddings looked up from a pre-learned table. To obtain $\bar{\Gamma}_N(u)$, we first build the set $\Gamma_N(u) = (v \in \pi(Neighbors(u)) \mid \Gamma(u))$, where $\Gamma_N(u)$ contains the vector embeddings of a subset of u's neighbors. π is a sampling operation (details follow herein). $\bar{\Gamma}_N(u)$ is obtained by performing a dimension-by-dimension average of entries in $\Gamma_N(u)$.

Assume given a node u, if $\Gamma(u)$ returns a $N_d \times 1$ dimensional continuous representation of u. Therefore, each node, such as v_s and v_t, is mapped to a $2N_d \times 1$ dimensional matrix.

We introduce 3 different sampling strategies for π,

1. *Random*: Given a set of neighbors, $Neighbors(u)$, return a randomly sampled set of n_{out} elements.

2. *Degree*: Given a set of neighbors, $Neighbors(u)$, return the top n_{out} nodes with the highest degree.

3. *Coherence*: Given a set of neighbors of u and a destination node v, return the top n_{out} nodes that are close to v in an embedding space.

The core component of **R** is a Bi-Directional LSTM, or BiLSTM. Given a BiLSTM model of length L_w (where L_w is the window parameter mentioned earlier), our input to R is a matrix of dimension $2N_d \times L_w$, where each element of a sequence of L_w nodes are substituted by their corresponding $2N_d \times 1$ dimensional embedding.

D is a dense layer followed by a graph component. Output from the dense layer is represented as $\bar{Y} = f_{activ}(W_d * X + b_d)$, where X is the $N_d \times 1$ dimensional input obtained from R. W_d is a weights matrix learnt from the data and b_d is a bias vector. We set f_{activ} to the hyperbolic tangent function.

4.4 Loss function

We seek to structure the dense layer output as a projection into the N_d-dimensional space and minimize the cosine distance to the vector encoding of the target node. The cosine proximity loss function between actual vector y and predicted vector $\bar{y}$ is defined as,

$$\mathcal{L} = \frac{\sum_{i=1}^{N_d} y[i] \cdot \bar{y}[i]}{\sqrt{\sum_{i=1}^{N_d}(y[i])^2} \cdot \sqrt{\sum_{i=1}^{N_d}(\bar{y}[i])^2}} \tag{7}$$

4.5 k-Coherent Path Generation

Given the KB G_{kb}, a model M and a query (v_s, v_t), we use a beam search algorithm to generate first k-paths. The algorithm begins with a search queue $\mathbb{P}_{cand}$. In every iteration, it removes the top candidate from the queue and uses the model M to generate the next set of nodes to visit. If a new candidate reaches the goal v_t, it is added to the result list. The algorithm terminates when the first-k answers are obtained or it reaches maximum hops h_{max}.

Input to M is generated using the query encoder Q, with input path prefix of length L_w and target node v_t, which returns a N_d-dimensional vector O_v. At this point, we find b-nodes in the $Neighbor(v)$ that are closest to O_v in vector space. b-serves a branching factor in the search, and is an internal parameter of the search algorithm (Algorithm 1). Note that with branching factor b, model generates top k = $b^{h_{max}-1}$ paths between v_s and v_t.

5 EXPERIMENTS

We evaluate our approach for generating explanatory paths for three unique knowledge bases. Our algorithms are implemented using Keras version 2.0 and all evaluations were performed using NVIDIA

Algorithm 1 Search algorithm to find k-explanatory paths between nodes v_s and v_t in graph G.

```
1:  procedure SEARCH($G_{kb}, M, v_s, v_t, b, h_{max}$)
2:      INIT($results, \emptyset$)
3:      INIT($\mathbb{P}_{cand}$, PREDICT($M, path(v_s)$)))
4:      while $size(\mathbb{P}_{cand}) > 0$ do
5:          $path$ = POP($\mathbb{P}_{cand}$)
6:          $\Delta$ = PREDICT($M, path$)
7:          $N_{cand}$ = NEAREST($G_{kb}, \Delta, b$)
8:          for all $v \in N_{cand}$ do
9:              $P_{new} = path \cup \{v\}$
10:             if $v = v_t$ then
11:                 $results = results \cup \{P_{new}\}$
12:                 if $size(results) = k$ then
13:                     break
14:             else
15:                 if $size(P_{new}) < L_{max}$ then
16:                     PUSH($\mathbb{P}_{cand}, P_{new}$)
17:     RETURN $results$
18: end procedure
```

Table 3: Summary of test datasets

Dataset	WIKI	NELL	FB
#Entities	4604	74431	75043
#Facts	119882	128870	345873
#Types	130	268	Unavailable
#Entity without type	10	5317	NA
#Relations	1	200	13
#Avg.Clus.Coef.	0.207	0.047	0.190
#Training Paths	44219	57973	359367
#Test Questions	5000	2819	5000

Tesla P100 GPUs, using Tensorflow-GPU version 1.5. The model learning was performed over different parameter configurations of learning rate : {0.01, 0.001}, optimizer : {'rmsprop, 'adam'}, and with or without l1, l2 regularization. The results are reported on best performing parameters for each dataset.

5.1 Datasets

We evaluate our model on Wikispeedia (a subset of Wikipeedia previously used in previous human wayfinding games) [36, 37], NELL and Freebase (FB) knowledge graphs (Table 3) [3] .

The datasets vary widely in characteristics, the Wikispeedia graph has average degree of 27 and significantly higher transitivity than others. Transitivity is computed as the fraction of possible triangles present in a graph G. Higher transitivity(indicates a denser graph, and in turn suggests higher query complexity for any search algorithm. NELL is extremely sparse with average degree of less than 2 and transitivity of 0.007. The Freebase version is used from [16], an almost a bipartite network containing edges of the form (p,r,t) for some person p, relation r and attribute t.

Wikispeedia database in addition to the KB also contains results of the human way finding game. The game presented users with a query entity pair selected from wikipedia and asked them to click through links on wikipedia pages to go from source to target entity. We use the paths generated by human users during this study to compare against the paths generated by the XQA model later in evaluation.

[3]using distribution from [4]

5.2 Embedding learning for KB entities

We use node2vec [15] to generate graph embedding for all the KBs. Based on random walks, node2vec controls the probability of selecting the next node in the graph walk by two parameters p and q ($p > 1$ => do not backtrack, $q > 1$ =>stay in local neighborhood). We used the default configuration, which ensures that nodes exhibiting high connectivity are closer in vector space. The generated sequences are used as input to an LSTM model to generate feature representation of every node in the graph. Each node embedding is initialized with a set of random numbers close to zero. We evaluate the embeddings over the node 'type' multi class prediction for different embedding learning parameters(random walk length, number of walks per node etc) and embedding dimensions. The embeddings performance stabilized for embedding size of 64, random walk length = 20 and number of walk per node =10 across datasets and were used in subsequent analysis.

5.3 Training Data Generation

Given a graph G, we generate training entity pairs (v_s, v_t) such that (v_s, v_t) is not in $E(G)$ i.e. the source and target are not directly connected in G. To generate training paths, we first perform random walks of max hop h_{max}. Guided by the feature analysis of *good* paths from user studies, we filter the randomly generated paths to include only paths with short length (2-hop) or paths with least divergence and high specificity for each entity pair, restricting training to most meaningful explanations for each question. While not using absolute thresholds for divergence and specificity introduces the possibility of selecting training paths that may have high divergence for some question pairs, we do not discard any entity pair or employ a divergence threshold to maintain coverage for possibly complex questions.

5.4 Evaluation Approach

5.4.1 Question Complexity. Given a question entity pair (v_s, v_t), we identify following features to characterize query complexity. 1) Shortest path distance as measured in hops. 2) Question Divergence measured as cosine($\Gamma_N(v_s)$,$\Gamma_N(v_t)$) , which provides a measure of their conceptual distance and strength of connectivity.

5.4.2 Evaluating Model Explanations. We evaluate the model generated answers based on following metrics. 1) Successful answer count: Given a set of query entity-pairs, we use the number of paths returned by Algorithm 1 as the primary evaluation metric to evaluate different model configurations 2) For queries for which an answer was found, we further analyze the answers from XQA in terms of their a) path divergence, b) specificity, and c) length. We benchmark our results against the answers generated by the human user for same questions in wikispeedia, and by the answers generated by MINERVA , a recently developed algorithms for reasoning over paths. For the evaluation, explanation generation used h_{max}=5 and branching factor b=2, to limit the exploration space

Table 4: Percentage of successful explanations as a function of shortest path

Shortest Path Len	2	3	4
Wikispeedia	93%	82%	0/1
Freebase	75%	71%	1/6
NELL	97%	92%	93%

5.5 Performance Evaluation

5.5.1 Number of Successful Explanations. We evaluate the most effective model configurations in generating explanations for each dataset across varying degree of question complexity. We evaluate the model for different neighbor sampling strategies (Random, Degree, Coherence) and number of neighbors included during query encoding for source and target nodes. Figure 5 and shows the impact of neighbor selection strategies and number of node neighbors incorporated during model learning step for wikispeedia and NELL test questions.We observe that adding neighborhood information during query encoding helps model answer significantly more questions for wikispeedia (7.57%) and NELL (8%). compared to model without any graph memory augmentation. However, the graph augmentation did not impact the freebase predictions significantly and remained consistent at around 73% success. This leads us to believe that graph augmentation is impactful for dense graphs with high transitivity , however may not be needed for very sparse graph like freebase.

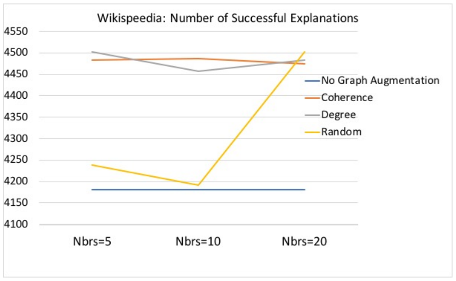
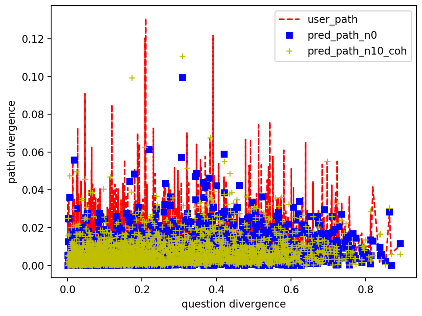


Figure 5: Wikispeedia: Impact of Graph Augmentation on Number of Successful Explanations

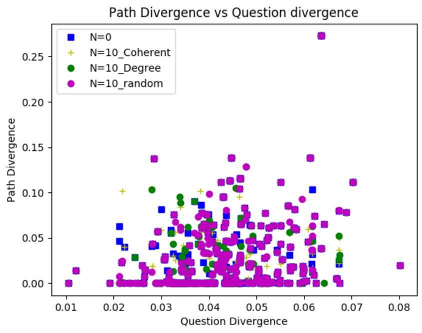
The percentage of successful explanations for each dataset are shown in Table 4 as a function of question complexity (measured by shortest path distance) for our most successful model configuration per dataset. XQA models perform very well on simpler as well as more complex questions, finding paths for more than 90% test entity pairs on wikispeedia and NELL for shortest distance=2. Further the performance degradation is very low as question complexity increases on each of the datasets, achieving 93% on NELL for queries with shortest distance=4. Wikispeedia and Freebase do not have sufficient test entities (< 10) for such and we provide the number of successful paths for given number of test cases in that category.

5.5.2 Evaluating Answer Quality. Table 5 illustrates the metrics for explanatory paths generated by various modeling strategies as well as by the human user for Wikispeedia and freebase. The model for Wikispeedia finds consistently paths with lower average length, lower divergence and and higher specificity(lower average degree) than human user in the wikispeedia way finding game. However, as model finds more paths with graph memory augmentation, the average path length increases slightly for *neighbors* >= 10 across different sampling strategies. For Freebase, the model learns to find short paths and with high coherence. However we observe that model also generalizes the paths through very generic (high degree) nodes in the freebase graph, given its bipartite and sparse characteristics.

Figure 6a plots the divergence of answer paths as a function of question divergence for the Wikispeedia dataset. First, it shows that the answer path divergences are robust to the variation of question divergence. Further the model with graph augmentation(n=10, coherence) finds paths with lower divergence i.e. high coherence compared to user and the model without memory augmentation.



(a) Impact of memory augmentation for Wikispeedia



(b) Impact of memory augmentation for Freebase

Figure 6: Validation of the impact of memory augmentation in producing answers with higher coherence.

5.6 Comparison with Human Answers

Figure 7 presents a comparison of model generated answers w.r.t. answers provided by the human users, and shows that performance is more promising for Freebase.

Table 5 and Figure 6a also summarizes the performance of XQA with respect to the answers generated by a human user. It can be seen that the coherence of the human answer suffers as the query entity pairs drift further in a vector space. However, any of the XQA approaches consistently manages to produce coherent answers, with shorter path length and less divergence.

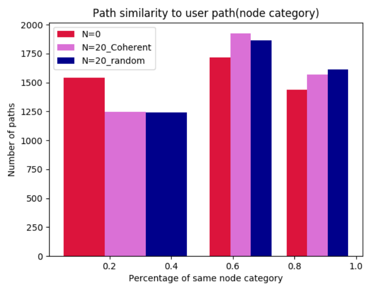
Table 6 contains two prime examples that demonstrates XQA's ability to match or outperform the human user. In the first example, we ask the human user to find a path between (Milan, Algebra). The

	Wikispeedia			Freebase		
Path Generator	**Average Path Length**	**Average Divergence**	**Average Degree**	**Average Path Length**	**Average Divergence**	**Average Degree**
Human User	4.49645	0.0105496	799.412	4.66197	0.0504211	6070.71
Model Nbrs_0	4.00122	0.00637905	479.697	2.72113	0.0319011	28568.4
Model 10_Random	4.09909	0.00529085	491.65	2.68169	0.0313085	27169.3
Model 10_Coherence	4.22558	0.00728912	616.647	2.68451	0.0306014	27659.7
Model 10_Degree	4.20772	0.00742075	614.13	2.67887	0.031391	27650.3
Model 20_Coherence	4.28995	0.00617365	577.46	NA	NA	NA
Model 20_Random	4.24589	0.00638251	589.175	2.5662	0.0303545	26860.6

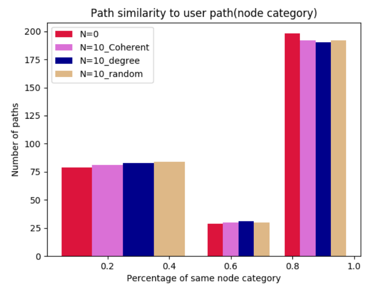
Table 5: Metrics for Explanatory Paths generated by XQA and human users(where available) for Wikispeedia, Freebase and Nell

Question : $Milan, Algebra$, shortest_path_length=3, embedding_distance=0.244				
Path Generator	**Path**	**Path Length**	**Divergence**	**Specificity**
Human User	Milan, Leonardo_da_Vinci, Mathematics, Algebra	4	0.0116	214
Model 20_Coherence	Milan, Leonardo_da_Vinci, Geometry, Algebra	4	0.0017	107
Question : $Oliver\_Cromwell, Max\_Planck$, shortest_path_length=2, embedding_distance=0.4192				
Human User	Oliver_Cromwell, Cambridge, Isaac_Newton, Physics, Max_Planck	5	0.0144	235
Model 20_Coherence	Oliver_Cromwell, David_Hume, Albert_Einstein, Max_Planck	4	0.0095	176

Table 6: Example Explanatory paths generated by XQA for different classes of entity pairs in Wikispeedia.



(a) Wikispeedia



(b) Freebase

Figure 7: Pattern Similarity of Model Generated Paths to User Paths (models with most successful explanations shown). X-axis for both plots indicate a fractional similarity between two paths measured by Jaccard overlap.

user finds a path through “Leonardo da Vinci” transitioning gradually to “Mathematics” and then Algebra. Noticeably the path human user finds connects (Milan, Algebra) through a prominent Mathematical personality from Milan. XQA traverses a path similar to the human user; however, it picks a more specific node and produces a more coherent and specific answer, choosing 'Geometry' over 'Mathematics' to transition from 'Vinci' to 'Algebra'. The second example is more intriguing where query entities (Oliver Cromwell, Max Planck) are much farther apart in vector space (and hence not densely as connected as first example). Oliver Cromwell was a English political leader and the human user found a path connecting to Max Planck through 'Cambridge' and 'Physics'. When compared to the human answer, our model found a shorter path with a different structure, selecting 'David Hume' who was also an English historian whom Einstein mentioned as inspiration (facts stated from Wikipedia) and produced an answer with higher coherence. This example demonstrates XQA's ability to generalize, capture the query intent and decompose it into appropriate sub-structures. Specifically, starting from “Oliver Cromwell”, it successfully captured the intent to reach an “Astronomer-Physicist" (which Isaac Newton, Albert Einstein and Max Planck belong to in Wikispeedia), and found a shorter path through a “Philosopher".

5.7 Comparison with Minerva

We evaluate XQA's performance against MINERVA [4] on Wikispeedia and NELL. While close, XQA and Minerva's objectives have subtle and important differences. Specification of relation as part of the query is key for Minerva. Given a training triple, it's learning process is oriented to associate frequent patterns in the graph as an explanation for the relation. As it traverses the graph, subgraphs that provide reachability between queries are treated as a reward. Instead of rewarding reachability through frequent subgraph patterns, XQA assigns greater importance to learning patterns of coherent subgraph structures that provide connectivity.

We compare Minerva with two question classes. First involve entity pairs connected through direct relationships in the graph. Second, we test on queries with entity pairs with 2-4 hops. Since, Minerva explanations are generated based on target relation, we use

Model/System	Successful paths
Minerva	1206
No memory augmentation	1456
Memory aug. + random neighbor selection	1460
Memory aug. + coh. based selection	1468

Table 7: NELL : Comparison with Minerva for question pairs with shortest path distance = 1 (input relation known). Number of entity pairs=2819

node category labels to create relations between training and test entity pairs.

Table 7 shows results for Minerva on NELL where the target relation for all test entity pairs were precisely available. XQA model outperforms minerva by finding paths for 21% more such entity pairs using graph augmentation. Further we observe that for entities where target relation is not specifically known shortest_path_length $(v_s, v_t) \geq 2$ hops, minerva is unable to find any paths, while XQA consistently finds paths for more than 70% test pairs 4. We observe similar limitation for Wikispeedia dataset, where specific relation type between nodes is not available (all edges are $linksTo$ relation) and minerva finds 15 paths out of 5000 entity pairs, whereas XQA models consistently find 4400+ paths.

Minerva generates explanations based on target relations, and works well when the target relation manifest in graph through frequent patterns. However, 1) for noisy KBs where lot more of the graph is manifested through generic relationships like $linksTo$ (specific relation types are not known a-priori), or semantically similar relationships may be modeled using different relation names, such approach do not perform as well. XQA in contrast, uses entity embeddings which can be learned for any arbitrary noisy KBs, using graph structure alone. 2) Further our path explanation metrics enable us to learn more coherent and meaningful answers for queries which require long chain of explanations.

6 RELATED WORK

Algorithms for relationship explanation build upon rich literature in the graph mining community such as mining connection subgraphs [9]. Different variants of the problem has gained attention over past years. These range from reporting the relationship between two entities as a function of frequent subgraph patterns [10], efficient enumeration of paths [21, 34], and exploring web-scale implementations using graph databases [11, 22]. There are also variants of the task itself, that expand the query from a pair of entities to a set of entities, or find connecting communities as an explanation for relationship [30]. Similar problem is also studied in the natural language processing community for explaining the relationship between two entities by finding a sequence of sentences [35].

The path ranking algorithm (PRA) [19] and REX [10] are two early inspirations for this work. Path ranking algorithm (PRA) [19] encodes a KB as a graph and performs random walks to collect paths between source and target nodes. Subsequently, this is trained by a classifier (such as logistic regression) with paths as features. Thus, we can feed in a path from a graph and predict a relation between the source and end nodes. [13] improved the PRA algorithm by performing walks in vector space and traversing semantically similar edges. This allows use of distributional closeness with symbolic inference, while circumventing the need to precisely map every distinct edge type to an ontology. REX [10] uses the knowledge of frequent subgraph patterns to rank and explain the relationship between two entities. Subgraphs connecting two entities are decomposed in terms of minimal, discriminative structures and ranked accordingly.

Algorithms for relationship explanation share many algorithmic challenges and solution approaches to few other topics, link prediction [2] and knowledge base completion [32]. Our approach is inspired by the recent body of work on neural network guided graph walk in vector space [5, 16, 26]. These papers primarily focus on the *knowledge base population* (KBP) problem, where given a starting node v_s, a sequence of relations $\mathcal{R} = \{r_1, r_2, ..., r_k\}$, the goal is to return a set of nodes $\{v_t\}$ that are reachable from v_s via $\mathcal{R}$, or given a pair of query nodes v_s, v_t, and a sequence of relations $\mathcal{R} = \{r_1, r_2, ..., r_k\}$ connecting v_s and v_t, the goal is to classify a relation between the query pair.

Semantic proximity search [23–25, 31] is another body of related work that is closer to KBP in spirit. Given a node in the network as the query, it ranks other nodes according to some semantic relation. For example, given a user (e.g., Alice) as a query, and the goal of semantic proximity search is to ask "who are her schoolmates?" The major difference between ours and this body of work is that we focus on the inverse problem. Given a pair of nodes v_s and v_t, our objective is to return a path in the knowledge graph as an evidence.

To our knowledge, Minerva [4] is the first work among the neural network driven approaches that study the explanatory query problem. However, its focus is restricted to the KBP or link prediction context. Given a KB, and a query of the form v_s, r_k, v_t, Minerva returns a path explaining the relationship r_k between v_s and v_t. We focus on the general relationship problem introduced by REX [10], where given v_s and v_t, our goal is to return top-k paths explaining their relationship.

7 CONCLUSION AND FUTURE WORK

We focus on *answering* of *relationship explanation queries*, that aims to explain-by-example the relationship between two entities in a KB. Our main contributions are through defining a set of metrics for quantitative characterization of the highly subjective notion of "interesting" and "logically consistent" subgraph explanations through user studies. Additionally, we present an algorithm XQA, that uses a graph-augmented neural-network to steer a graph walk that finds high-quality answer paths. This algorithm also outperforms a state of the art system on Wikispeedia and NELL datasets. When compared to human provided answers, XQA produces more coherent and specific explanations on the Freebase dataset. Our experiments on the highly noisy Wikispeedia dataset clearly shows the impact of "graph-based memory augmentation" for LSTM. Improving performance w.r.t. human users across a broad swath of datasets and developing algorithms with better performance guarantees remain topics for future work. Last but not the least, we foresee how our work can complement some of the recent developments in this field. The divergence measure can be extended to return subgraphs instead of a path explaining the relationship. Finding divergent explanation of entity relationships beyond top-k querying remains another topic for future work.

REFERENCES

[1] Amr Ahmed, Nino Shervashidze, Shravan Narayanamurthy, Vanja Josifovski, and Alexander J Smola. 2013. Distributed large-scale natural graph factorization. In *Proceedings of the 22nd international conference on World Wide Web*. ACM, 37–48.

[2] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*. 2787–2795.

[3] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 891–900.

[4] Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. 2018. Go for a Walk and Arrive at the Answer: Reasoning Over Paths in Knowledge Bases using Reinforcement Learning. *ICLR* (2018).

[5] Rajarshi Das, Arvind Neelakantan, David Belanger, and Andrew McCallum. 2017. Chains of reasoning over entities, relations, and text using recurrent neural networks. *EACL* (2017).

[6] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 135–144.

[7] Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S Weld. 2008. Open information extraction from the web. *Commun. ACM* 51, 12 (2008), 68–74.

[8] Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open question answering over curated and extracted knowledge bases. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1156–1165.

[9] Christos Faloutsos, Kevin S McCurley, and Andrew Tomkins. 2004. Fast discovery of connection subgraphs. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 118–127.

[10] Lujun Fang, Anish Das Sarma, Cong Yu, and Philip Bohannon. 2011. Rex: Explaining relationships between entity pairs. *Proceedings of the VLDB Endowment* 5, 3 (2011), 241–252.

[11] Valeria Fionda and Giuseppe Pirro. 2017. Explaining and querying knowledge graphs by relatedness. *Proceedings of the VLDB Endowment* 10, 12 (2017), 1913–1916.

[12] Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian Suchanek. 2013. AMIE: association rule mining under incomplete evidence in ontological knowledge bases. In *Proceedings of the 22nd international conference on World Wide Web*. ACM, 413–422.

[13] Matt Gardner, Partha Pratim Talukdar, Jayant Krishnamurthy, and Tom Mitchell. 2014. Incorporating vector space similarity in random walk inference over knowledge bases. (2014).

[14] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. 2016. *Deep learning*. Vol. 1. MIT press Cambridge.

[15] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 855–864.

[16] Kelvin Guu, John Miller, and Percy Liang. 2015. Traversing knowledge graphs in vector space. *EMNLP* (2015).

[17] Mohit Iyyer, Jordan Boyd-Graber, Leonardo Claudino, Richard Socher, and Hal Daumé III. 2014. A neural network for factoid question answering over paragraphs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 633–644.

[18] Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. MIMIC-III, a freely accessible critical care database. *Scientific data* 3 (2016), 160035.

[19] Ni Lao, Tom Mitchell, and William W Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 529–539.

[20] Omer Levy, Luke Zettlemoyer, Yejin Choi, and Eunsol Choi. 2018. Ultra-Fine Entity Typing.. In *ACL (1)*. 87–96.

[21] Jiongqian Liang, Deepak Ajwani, Patrick K Nicholson, Alessandra Sala, and Srinivasan Parthasarathy. 2016. What links alice and bob?: Matching and ranking semantic patterns in heterogeneous networks. In *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 879–889.

[22] Matteo Lissandrini, Davide Mottin, Yannis Velegrakis, and Themis Palpanas. 2018. X 2 Q: your personal example-based graph explorer. *Proceedings of the VLDB Endowment* 11, 12 (2018), 2026–2029.

[23] Zemin Liu, Vincent W Zheng, Zhou Zhao, Hongxia Yang, Kevin Chen-Chuan Chang, Minghui Wu, and Jing Ying. 2018. Subgraph-augmented Path Embedding for Semantic User Search on Heterogeneous Social Network. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 1613–1622.

[24] Zemin Liu, Vincent W Zheng, Zhou Zhao, Fanwei Zhu, Kevin Chen-Chuan Chang, Minghui Wu, and Jing Ying. 2017. Semantic Proximity Search on Heterogeneous Graph by Proximity Embedding.. In *AAAI*. 154–160.

[25] Zemin Liu, Vincent W Zheng, Zhou Zhao, Fanwei Zhu, Kevin Chen-Chuan Chang, Minghui Wu, and Jing Ying. 2018. Distance-aware dag embedding for proximity search on heterogeneous graphs. AAAI.

[26] Arvind Neelakantan, Benjamin Roth, and Andrew Mc-Callum. 2015. Compositional vector space models for knowledge base inference. In *2015 aaai spring symposium series*.

[27] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. 2016. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1105–1114.

[28] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 701–710.

[29] Arun V Sathanur, Sutanay Choudhury, Cliff Joslyn, and Sumit Purohit. 2017. When Labels Fall Short: Property Graph Simulation via Blending of Network Structure and Vertex Attributes. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 2287–2290.

[30] Stephan Seufert, Klaus Berberich, Srikanta J Bedathur, Sarath Kumar Kondreddi, Patrick Ernst, and Gerhard Weikum. 2016. ESPRESSO: Explaining Relationships between Entity Sets. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, 1311–1320.

[31] Yu Shi, Po-Wei Chan, Honglei Zhuang, Huan Gui, and Jiawei Han. 2017. Prep: Path-based relevance from a probabilistic perspective in heterogeneous information networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 425–434.

[32] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in neural information processing systems*. 926–934.

[33] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*. 2440–2448.

[34] Kristina Toutanova, Victoria Lin, Wen-tau Yih, Hoifung Poon, and Chris Quirk. 2016. Compositional Learning of Embeddings for Relation Paths in Knowledge Base and Text. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, 1434–1444. http://www.aclweb.org/anthology/P16-1136

[35] Nikos Voskarides, Edgar Meij, Manos Tsagkias, Maarten De Rijke, and Wouter Weerkamp. 2015. Learning to explain entity relationships in knowledge graphs. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Vol. 1. 564–574.

[36] Robert West and Jure Leskovec. 2012. Human wayfinding in information networks. In *Proceedings of the 21st international conference on World Wide Web*. ACM, 619–628.

[37] Robert West, Joelle Pineau, and Doina Precup. 2009. Wikispeedia: An Online Game for Inferring Semantic Distances Between Concepts. In *Proceedings of the 21st International Jont Conference on Artifical Intelligence (IJCAI'09)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1598–1603. http://dl.acm.org/citation.cfm?id=1661445.1661702

[38] Xifeng Yan and Jiawei Han. 2002. gspan: Graph-based substructure pattern mining. In *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*. IEEE, 721–724.