Answering Complex Questions by Joining Multi-Document Evidence with Quasi Knowledge Graphs

Xiaolu Lu*
RMIT University, Australia
xiaolu.lu@rmit.edu.au

Soumajit Pramanik
MPI for Informatics, Germany
pramanik@mpi-inf.mpg.de

Rishiraj Saha Roy
MPI for Informatics, Germany
rishiraj@mpi-inf.mpg.de

Abdalghani Abujabal
Amazon Alexa, Germany
abujabaa@amazon.de

Yafang Wang
Ant Financial Services Group, China
yafang.wyf@antfin.com

Gerhard Weikum
MPI for Informatics, Germany
weikum@mpi-inf.mpg.de

ABSTRACT

Direct answering of questions that involve multiple entities and relations is a challenge for text-based QA. This problem is most pronounced when answers can be found only by joining evidence from multiple documents. Curated knowledge graphs (KGs) may yield good answers, but are limited by their inherent incompleteness and potential staleness. This paper presents QUEST, a method that can answer complex questions directly from textual sources on-the-fly, by computing similarity joins over partial results from different documents. Our method is completely unsupervised, avoiding training-data bottlenecks and being able to cope with rapidly evolving ad hoc topics and formulation style in user questions. QUEST builds a noisy quasi KG with node and edge weights, consisting of dynamically retrieved entity names and relational phrases. It augments this graph with types and semantic alignments, and computes the best answers by an algorithm for Group Steiner Trees. We evaluate QUEST on benchmarks of complex questions, and show that it substantially outperforms state-of-the-art baselines.

1 INTRODUCTION

1.1 Motivation

Question answering (QA) over the Web and derived sources of knowledge, has been well-researched [12, 24, 40]. Studies show that many Web search queries have the form of questions [64]; and this fraction is increasing as voice search becomes ubiquitous [29]. The focus of this paper is on providing direct answers to fact-centric ad hoc questions, posed in natural language, or in telegraphic form [35, 51]. Earlier approaches for such QA, up to when IBM Watson [25] won the Jeopardy! quiz show, have mostly tapped into textual sources (including Wikipedia articles) using passage retrieval and other techniques [49, 62]. In the last few years, the paradigm of translating questions into formal queries over structured knowledge graphs (KGs), also known as knowledge bases (KBs), and databases (DBs), including Linked Open Data, has become prevalent [18, 59].

QA over structured data translates the terms in a question into the vocabulary of the underlying KG or DB: entity names, semantic types, and predicate names for attributes and relations. State-of-the-art systems [1, 6, 7, 24, 69] perform well for simple questions that involve a few predicates around a single target entity (or a qualifying entity list). However, for complex questions that refer to multiple entities and multiple relationships between them, the question-to-query translation is very challenging and becomes the make-or-break point. As an example, consider the question:

"footballers of African descent who played in the FIFA 2018 final and the Euro 2016 final?"

A high-quality, up-to-date KG would have answers like *'Samuel Umtiti'*, *'Paul Pogba'* or *'Blaise Matuidi'*. However, this works only with a perfect mapping of question terms onto KG-predicates like `bornIn`, `playedFor`, `inFinal`, etc. This strong assumption is rarely satisfied for such complex questions. Moreover, if the KG misses some of the relevant pieces, for example, that a football team played in a final (without winning it), then the entire query will fail.

1.2 State of the Art and its Limitations

QA over KGs. State-of-the-art work on QA over KGs has several critical limitations: (i) The question-to-query translation is brittle and tends to be infeasible for complex questions. (ii) Computing good answers depends on the completeness and freshness of the underlying KG, but no KG covers everything and keeps up with the fast pace of real-world changes. (iii) The generation of good queries is tied to a specific language (usually English) and style (typically full interrogative sentences), and does not generalize to arbitrary languages and language registers. (iv) Likewise, the translation procedure is sensitive to the choice of the underlying KG/DB source, and cannot handle ad hoc choices of several sources that are seen only at QA-execution time.

QA over text. State-of-the-art methods in this realm face major obstacles: (i) To retrieve relevant passages for answer extraction, all significant question terms must be matched in the *same document*, ideally within short proximity. For example, if a QA system finds players in the 2018 FIFA World Cup Final and the UEFA Euro Cup 2016 Final in *different* news articles, and information on their birthplaces from Wikipedia, there is no single text passage for proper answering. (ii) The alternative of fragmenting complex questions

*Part of this work was done during the author's internship at the MPI for Informatics.

SIGIR'19, July 2019, Paris, France
© 2019 Association for Computing Machinery.
ACM ISBN 123-4567-24-567/08/06…$15.00
https://doi.org/10.475/123_4

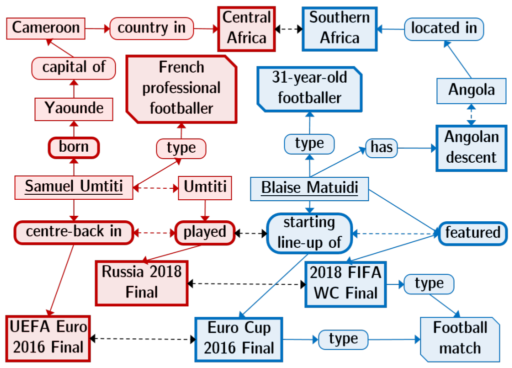


Figure 1: A quasi KG for our running example question.

into simpler sub-questions requires syntactic decomposition patterns that break with ungrammatical constructs, and also a way of stitching sub-results together. Other than for special cases like temporal modifiers in questions, this is beyond the scope of today's systems. (iii) Modern approaches that leverage deep learning critically rely on training data, which is not easily available for complex questions, and have concerns on interpretability. (iv) Recent works on text-QA considered scenarios where a question is given with a specific set of documents that contain the answer(s).

This paper overcomes these limitations by providing a novel unsupervised method for combining answer evidence from multiple documents retrieved dynamically, joined together via KG-style relationships automatically gathered from text sources.

1.3 Approach and Contribution

We present a method and system, called QUEST (for "QUEstion answering with Steiner Trees"), that taps into text sources for answers, but also integrates considerations from the KG-QA (also referred to as KB-QA) paradigm. QUEST first constructs an ad hoc, noisy knowledge graph by dynamically retrieving question-relevant text documents and running Open Information Extraction (Open IE) [44] on them to produce subject-predicate-object (SPO) triples. In contrast to a curated KG (like YAGO or Wikidata), these triples contain names and phrases rather than canonicalized entities and predicates, and hence exhibit a high degree of noise. Thus, we additionally compute edges that connect potentially synonymous names and phrases. We refer to the resulting graph as a *quasi KG*, which captures combined cues from many documents and is then treated as the knowledge source for the QA algorithm (example in Fig. 1 for the footballer question, with bidirectional dashed edges denoting potential synonymy among the nodes).

Good answers among the nodes of the quasi-KG should be well-connected with all nodes that (approximately) match the phrases from the input question. We refer to these matching nodes as *cornerstones* (nodes with thick borders in Fig. 1). This criterion can be cast into computing Group Steiner Trees (GST) with the cornerstones as terminals [26]. All non-terminal nodes of the trees are candidate answers. This computation is carried out over a weighted graph, with weights based on matching scores and extraction confidences. Finally, answers are filtered and ranked by whether they are compatible with the question's lexical answer type and other criteria.

In Fig. 1, all red nodes and edges, and all blue nodes and edges, constitute two GSTs, respectively yielding the answers *'Samuel Umtiti'* and *'Blaise Matuidi'* (underlined). Unlike most QA systems where correct entity and relation linking are major bottlenecks for success, QUEST *does not need any explicit disambiguation* of the question concepts, and instead harnesses the effect that GSTs themselves establish a common context for ambiguous names. Thus, finding a GST serves as a *joint disambiguation* step for relevant entities, relations, and types, as different senses of polysemous concepts are unlikely to share several inter-connections. Notably, the Steiner tree provides *explainable* insights into how an answer is derived.

It is the nature of ad hoc Web questions that dynamically retrieved documents contain a substantial amount of uninformative and misleading content. Instead of attempting to perfectly clean this input upfront, our rationale is to cope with this noise in the answer computation rather than through tedious efforts on entity disambiguation and relation canonicalization in each step.

GST algorithms have been used for keyword search over relational graphs [8, 13, 71], but they work for a simple class of keyword queries with the sole condition of nodes being related. For QA, the problem is much more difficult as the input is a full question that contains multiple conditions with different entities and predicates.

Contribution. The salient points of this work are:

- QUEST is a novel method that computes direct answers to complex questions by dynamically tapping arbitrary text sources and joining sub-results from multiple documents. In contrast to neural QA methods that rely on substantial amounts of training data, QUEST is unsupervised, avoiding training bottlenecks and the potential bias towards specific benchmarks.
- QUEST combines the versatility of text-based QA with *graph-structure awareness* of KG-QA, overcoming the problems of incompleteness, staleness, and brittleness of QA over KGs alone.
- We devise advanced graph algorithms for computing answers from noisy text-based entity-relationship graphs.
- Experiments show the viability of our method and its superiority over state-of-the-art baselines. To facilitate comparison and reproducibility, all **data, code**, and **results** from this work are available at https://github.com/pramanik/quest and http://people.mpi-inf.mpg.de/~rsaharo/quest-data-code.zip. An online **demo** is accessible at https://gate.d5.mpi-inf.mpg.de/quest.

2 SYSTEM OVERVIEW

Complex questions. Our emphasis is on complex questions that refer to multiple entities and relationships. There are other notions of complex questions, for example, those requiring grouping, comparison, and aggregation, or when the question involves negations. These are not considered in this paper.

Answering pipeline. QUEST processes questions in two phases: (1) on-the-fly construction of the quasi KG for the question, and, (2) the graph algorithm for computing ranked answers. Together, these phases comprise the following five steps:

(1a) Retrieving question-relevant documents from an open corpus (e.g., the entire Web),

(1b) Extracting proximity-based SPO triples from these documents using Open IE techniques,

(1c) Building a noisy quasi KG from these triples,
(2a) Computing GSTs on the quasi KG to derive candidate answers,
(2b) Filtering and scoring candidates to produce ranked answers.

Step (1a), document retrieval, is performed using any Web search engine or an IR system. Working with a reasonably-sized pool of pseudo-relevant documents ensures that the subsequent graph algorithm is computationally tractable. Preprocessing of documents includes part-of-speech (POS) tagging, named entity recognition (NER), and lightweight coreference resolution by linking personal and possessive pronouns like *he, she, him, her, his*, and *hers*, to the nearest named entity in the preceding text. Sec. 3 and 4 give details on steps (1b) and (1c) for graph construction and steps (2a) and (2b) for the graph algorithm, respectively.

3 GRAPH CONSTRUCTION

3.1 Extracting SPO Triples

Answering complex questions often requires evidence from multiple documents. For our running example, the evidence needs to comprise *born in Africa, being a footballer, playing in the UEFA Euro 2016 final*, and *playing in the FIFA World Cup 2018 final.* It is unlikely that all these cues can be found in a single document. Therefore, we first retrieve a pool of relevant documents from a search engine (Google or Bing) using the entire question as a keyword query. To identify cues in the matching documents and to join them across multiple documents, we apply Open IE [37, 44] to extract SPO triples.

Popular tools for Open IE include Stanford OpenIE [3], OpenIE 5.0 (previously ReVerb) [44], and ClausIE [14], but each comes with limitations. Stanford OpenIE focuses on precision, and produces correct but relatively few triples, thus losing cues from long sentences. For example, consider the opening sentence from Wikipedia's article on Umtiti: *"Samuel Yves Umtiti is a French professional footballer who plays as a centre-back for Spanish club Barcelona and the French National Team."* Stanford's extractor misses the noun-mediated relation *'centre-back for'*, and the information about playing for the French National Team. While OpenIE 5.0 and ClausIE incorporate better support for such dependent clauses, they often produce very long objects from complex sentences, making it difficult to align them across different triples and rendering subsequent graph construction infeasible (e.g. *'a French professional footballer who plays as a centre-back for Spanish club Barcelona and the French National Team'* and *'as a centre-back for the French National Team'* are objects from OpenIE and ClausIE from the sentence above). Therefore, we devised our own (recall-oriented) IE method with judicious consideration to *phrase ordering* and *term proximities.* In contrast to the other tools, our method produces more noisy triples, but these are taken care of in the subsequent steps (see Sec. 4).

Pattern-based extraction. We perform POS tagging and NER on each sentence of the retrieved documents, and subsequently treat the constituent words of named entities as single tokens. Based on these annotations, we extract the following SPO triples respecting phrase *ordering*:

- **Verb-phrase-mediated triples:** (X, V, Y) for named entities (NE) or noun phrases (NP) X and Y such that the sentence has the form "$\ldots X \ldots V \ldots Y \ldots$", where V is of the POS pattern `verb` (e.g., "plays") or `verb+preposition` (e.g., "plays for"), and no other verb appears in the text span from X to Y.
- **Noun-phrase-mediated triples:** (X, N, Y) for NEs or NPs X and Y such that the sentence has the form "$\ldots X \ldots N \ldots Y \ldots$", where N is of the POS pattern `noun+preposition` (e.g., "centre-back for"), and no other phrase of this POS pattern appears in the text span from X to Y (rules from Yahya et al. [68]).

Auxiliary verbs (*is, can, may, ...*) are not considered. Sample triples from QUEST's noisy extractor for the previous example: *(Samuel Umtiti, centre-back for, Spanish club Barcelona), (Samuel Umtiti, centre-back for, French National Team)* (both correct), and *(French professional footballer, plays as, Spanish club Barcelona)* (noisy).

Proximity-based scoring. To associate an SPO triple with a confidence score, we use pairwise distances between the triple's parts (S, P, or O) in the document where it stems from. We define the distance d between two items as the number of intruding words plus one (to avoid zero distances), and the score is set to $1/d$. This captures the intuition that closer two parts are in text, higher is their score. As a result, unlike conventional tools [3, 44], S-P and P-O pairs are allowed to take different scores. When two items co-occur in more than one sentence $\{\mathcal{S}_i\}$, we use the *sum* of inverses of their distances in these sentences $\{d(\mathcal{S}_i)\}$ as the score ($= \sum_{\mathcal{S}_i} 1/d(\mathcal{S}_i)$), thereby *leveraging redundancy of evidence.*

3.2 Building the Quasi KG

Quasi KG. The quasi KG consists of:

- nodes corresponding to the S, P, and O arguments of the extracted triples,
- type nodes corresponding to the S and O arguments,
- directed edges connecting S to P and P to O from the extracted triples above,
- directed (bidirectional) edges connecting nodes with equal or highly similar names or phrases, and,
- directed edges connecting S or O nodes to their types.

The quasi KG is the key asset to aggregate evidence from multiple documents. Typically, an O node from one triple would have an edge to an S node from another triple if they refer to the same entity, or they would be the same node in case they match exactly. Note, however, that the graph contains only surface forms; there is no canonicalization of entity names or phrases for S/O nodes – hence our terminology of *quasi* KGs. For example, 'Pogba', 'Paul Pogba' and 'Paul Labile Pogba' co-exist as different nodes in the graph, but could later be identified as synonyms for the same entity. This notion of similarity-based equivalence also holds for nodes that represent P arguments by phrases such as *'born in'* and *'comes from'*, and will be harnessed in subsequent steps. Fig. 1 shows an example of a noisy quasi KG that contains evidence for answers of our running example. Bidirectional dashed edges denote potential synonymy between nodes.

Types of nodes. S, P, and O arguments of extracted triples become individual nodes. Nodes are typed based on their semantic roles:

- **Entity nodes** are all nodes corresponding to S or O arguments (regular rectangular nodes in Fig. 1, e.g. *'Samuel Umtiti'*).
- **Relation nodes** are all nodes corresponding to P arguments (rectangular nodes with rounded corners, e.g. *'centre-back in'*).

- **Type nodes**, explained below, are added by inferring semantic types for entity nodes (rectangular nodes with snipped corners, e.g. *'French professional footballer'*).

The distinction into entity and relation nodes is important for answer extraction later, as relations rarely ever serve as final answers. Usually, KGs represent relations as edge labels; our choice of having explicit relation nodes is better suited for the graph algorithms that QUEST runs on the quasi KG.

QUEST creates *type nodes* connected to entity nodes via *Hearst patterns* [30] (e.g., NP_1 *such as* $\{NP_2,\}^*$ with POS tag *NP* for noun phrases) applied to sentences of the retrieved documents (e.g., "*footballers* such as Umtiti, Matuidi and Pogba"). As an alternative or complement, QUEST can also use type information from curated KGs by mapping the names in the entity nodes to KG entries by named entity disambiguation (NED) methods. Like entities and relations, types are not canonicalized to any taxonomy. We found that even such free-form text extractions for entity typing is a valuable asset for QA, as substantiated by our graph ablation experiments.

Types of edges. The quasi KG contains four types of edges:

- **Triple edges** connect two adjacent arguments of the extracted triples, i.e., S and P arguments, and P and O arguments from the same triple (entity-relation solid edges, e.g., between *'Blaise Matuidi'* and *'featured'* and, *'featured'* and *'2018 FIFA WC Final'*).
- **Type edges** connect entity nodes and their corresponding type nodes (e.g., between *'Blaise Matuidi'* and *'type'*, and *'type'* and *'31-year-old footballer'*).
- **Entity alignment edges** connect potentially equivalent entity nodes, that is, with sufficiently high similarity (dashed edges, e.g., between *'Samuel Umtiti'* and *'Umtiti'*).
- **Relation alignment edges** connect potentially synonymous relation nodes (dashed edges, e.g., between *'played'* and *'starting line-up of'*).

To identify node pairs for alignment edges, we can harness resources like *entity-mention dictionaries* [31, 52], *paraphrase databases* [28, 46], and *word/phrase embeddings* [45, 47].

Node weights. Node weights are derived from similarity scores with regard to tokens in the input question. For entity nodes, we use thresholded similarities from entity-mention dictionaries as explained in Sec. 5.3. For type nodes and relation nodes, the similarity of the node label is with regard to the highest-scoring question token (after stopword removal). In QUEST, this similarity is computed using word2vec [45], GloVe [47], or BERT [17] embeddings.

Edge weights. For *triple edges*, confidence scores (see Sec. 3.1) are used as weights. Having different confidence scores for S-P and P-O fits in well in this model as weights for the corresponding edges. For *alignment edges*, the weight is the similarity between the respective pair of entity nodes or relation nodes. See Sec. 5.3 for specifics of these computations. For *type edges* we set edge weights to 1.0, as these are the most reliable (relative to the noisier categories).

4 GRAPH ALGORITHM

4.1 Computing Group Steiner Trees

Cornerstones. To find answers in the quasi KG, we first identify pivotal nodes that we call *cornerstones*: every node that matches a word or phrase in the question, with similarity above a threshold, becomes a cornerstone. For example, *'FIFA 2018 final'* from the example question is matched by *'2018 FIFA WC Final'* and *'Russia 2018 Final'* (high similarity via lexicons) in the graph. Also, relation nodes (e.g., *'born'* and *'Angolan descent'*, with high word2vec embedding similarity to *'descent'*) and type nodes (e.g., *'French professional footballer'* and *'31-year-old footballer'* matching question term *'footballers'*) become cornerstones. All cornerstone nodes for our running example question have thick borders in Fig. 1. These nodes are weighted based on the matching or similarity scores.

Group Steiner Trees. The key idea for identifying answer candidates is that these nodes should be tightly connected to many cornerstones. To formalize this intuition, we consider three factors: (i) answers lie on paths connecting cornerstones, (ii) short paths are preferred, and (iii) paths with higher weights are better. These criteria are captured by the notion of a **Steiner Tree**:

- Given an undirected and weighted graph (V, E) with nodes V, edges E, and weights $w_{ij} \geq 0$ (for the edge between nodes i and j), and given a subset $T \subseteq V$ of nodes called *terminals*, compute a tree (V^*, E^*) where $V^* \subseteq V, E^* \subseteq E$ that connects all terminals T and has minimum cost in terms of total edge weights: $\min \sum_{ij \in E^*} w_{ij}$ with $T \subseteq V^*$.

For two terminals, the solution is the shortest path, but our application comes with many *terminals, namely the cornerstones*. Moreover, our terminals are grouped into sets (the cornerstones per token of the question), and it suffices to include at least one terminal from each set in the Steiner Tree. This generalized problem is known as computing a **Group Steiner Tree (GST)** [20, 26, 41]:

- Given an undirected and weighted graph (V, E) and given groups of terminal nodes $\{T_1, \ldots, T_l\}$ with each $T_\nu \subseteq V$, compute the minimum-cost tree (V^*, E^*) that connects at least one node from each of $\{T_1, \ldots, T_l\}$: $min \sum_{ij \in E^*} w_{ij}$ s.t. $T_\nu \cap V^* \neq \phi, \forall T_\nu$.

Answer candidates for a question are *inner nodes* of a GST (non-terminals). For example, the quasi KG of Fig. 1 shows two GSTs, with nodes in red and blue, respectively, that contain correct answers *'Samuel Umtiti'* and *'Blaise Matuidi'* (underlined). Algorithms for computing GSTs typically operate on undirected graphs with non-negative edge weights reflecting costs. Hence, we cast the quasi KG into an undirected graph by ignoring the orientation of edges, and we convert the $[0, 1]$-normalized similarity-score weights into cost weights by setting $cost = 1 - score$. Node weights were used for cornerstone selection and are disregarded for the GST computation.

Algorithm. Steiner trees are among the classical NP-complete problems, and this holds for the GST problem too. However, the problem has tractable fixed-parameter complexity when the number of terminals is treated as a constant [22], and there are also good polynomial-time approximation algorithms extensively applied in the area of keyword search over databases [20, 36, 41]. In QUEST, we build on the exact-solution method by [20], which uses dynamic programming and has exponential run-time in the length of the question but has $O(n \log n)$ complexity in the graph size. The algorithm works as follows. Starting from each terminal, trees are iteratively *grown* by adding least-cost edges from their neighborhoods. Trees are periodically *merged* when common vertices are encountered. A priority queue (implemented by a Fibonacci heap) holds all trees in increasing order of tree costs, and maintains the set of terminals a specific tree *covers*. The process stops when a tree is found that *covers* all terminals (contains at least one terminal per group) for the GST problem. This bottom-up dynamic programming approach using a priority queue ensures optimality of the result.

Relaxation to GST-*k*. In our setting, we are actually interested in the top-k trees in ascending order of cost [20, 41]. This is for robustness, as the best tree may be a graph with cornerstones only; then we cannot read off any answers from the GST. Moreover, computing several trees provides us with a good way of ranking multiple candidate answers (see 'Answer scoring' below). Therefore, we use the extended GST-k algorithm of [20], where k is typically small (≤ 50). The priority-queue-based algorithm naturally supports this top-k computation (fetch top-k trees instead of the top-1 only).

4.2 Filtering and Ranking Answers

The GSTs may contain several candidate answers, and so it is crucial to filter and rank the candidates.

Answer filtering. We first remove all candidates that are not entities (i.e., relation and type nodes are not considered). The main pruning is based on lexical *type checking*. For a given question, we infer its expected answer type using lexico-syntactic patterns from [72]. This expected answer type is then compared to the types of a candidate answer (type node labels attached to entity candidate), using cosine similarity between word2vec embeddings [45]. Similarity between multi-word phrases is performed by first averaging individual word vectors in the phrase [65], followed by computing the cosine similarity between the phrase embeddings. Candidates that do not have types with similarity above a threshold are dropped.

Answer aggregation. Since answers are surface forms extracted from text, we need to reduce redundancy (e.g. to avoid returning both *'Umtiti'* and *'Samuel Umtiti'* to the user). QUEST aggregates answers based on (i) token sequences, and (ii) alignments. For (i), two answers are merged if one is a *subsequence* (not necessarily substring) of another (e.g., *'Paul Labile Pogba'* and *'Paul Pogba'*). For (ii), two answers are merged if there is an alignment edge between them. This indicates that they are possibly aliases of the same entity (e.g., *'Christiano Ronaldo'* and *'CR7'*).

Answer scoring. After aggregation, answers are scored and ranked by exploiting their presence in multiple GSTs. However, instead of simple counts, we consider a weighted sum by considering the inverses of the tree cost as the weight for a GST. We examine effects of alternatives like total node weights in these GSTs, and distances of answers to cornerstones, in our empirical analysis later (Sec. 7).

5 EVALUATION SETUP

5.1 Rationale for Experiments

The key hypotheses that we test in experiments is that QUEST can handle *complex questions* that involve multiple entities and relations, and can cope with the *noise in the quasi KG*. Popular QA benchmarks like WebQuestions [7], SimpleQuestions [10], TREC [2, 19], QALD [60], or SQuAD [48], are not suitable, as they mostly focus on answering simple questions or understanding natural language passages. In contrast, we are interested in computing direct answers for ad hoc information needs by advanced users, tapping into all kinds of contents including informal text and Web tables. Therefore, we adopted the benchmark from [1] for structurally complex questions, and we compiled a new benchmark of complex questions from trending topics with questions that stress the dynamic and ad hoc nature of evolving user interests. Further, questions in both of these benchmarks indeed require stitching information across *multiple documents* for faithful answering, another desideratum for evaluating the capability of QUEST.

As for baselines against which we compare QUEST, we focus on unsupervised and distantly supervised methods. The latter include neural QA models which are pre-trained on large question-answer collections, with additional input from word embeddings. These methods are well-trained for QA in general, but not biased towards specific benchmark collections. We are interested in robust behavior for ad hoc questions, to reflect the rapid evolution and unpredictability of topics in questions on the open Web. Hence this focus on unsupervised and distantly supervised methods.

5.2 Question Benchmarks

Complex questions from WikiAnswers (CQ-W). This is a set of 150 complex fact-centric questions [1] paired with answers that are extracted from a curated WikiAnswers corpus [23]. Questions in this dataset were specifically sampled to have multiple entities and/or relations. Unfortunately, baselines in [1] cannot be adopted here, as they only run over KGs and cannot operate over text.

Complex questions from Trends (CQ-T). To study situations where KG incompleteness is a major concern, we created a new benchmark of 150 *complex* questions using *emerging entities* from Google Trends, including entities not having Wikipedia pages at all. Five students visited https://trends.google.com/trends/, where *'trending searches'* lists topics of current interest (with USA as location). For every trending topic, the students looked at *'related queries'* by Google users on this topic. Wherever possible, the students then selected a fact-centric information need from these queries (e.g., *"caitlin mchugh engaged"*) and augmented it into a more complex question (e.g., Q: *"Which TV actress was engaged to John Stamos and had earlier played in the Vampire Diaries?"*; A: *'Caitlin McHugh'*). The added nugget of complexity is a mix of *conjunctive, compositional, temporal*, and other *constraints*. Finally, the students provided answers by searching the Web. Each student contributed 30 (question, answer) pairs.

Wikidata is one of the most popular KGs today: we found that 29% of questions in CQ-T *do not have their answer entities in Wikidata* and 50% *do not have Wikidata facts* connecting question and answer entities (as of January 2019). This is often due to relations like (*interviewed, co-appeared in event, married at venue*, etc.) which are of popular interest yet beyond most KGs. This illustrates **KG-incompleteness** and motivates our focus on answering from dynamically retrieved Web text.

Answers to all 300 questions are manually augmented with **aliases** (2.5 aliases per original gold answer on average) to be *fair to competing systems* for extracting correct alternative surface forms (e.g., the football player *'Cristiano Ronaldo dos Santos Aveiro'* is expanded with *subsequences 'Cristiano Ronaldo'*, *'Ronaldo'*, etc.). The mean number of unique gold answers is about 1.42 (243/300 questions have exactly one correct answer). We verified that almost

Dataset	#Nodes			#Edges		
	Entity	Relation	Type	Triple	Alignment	Type
CQ-W	440	853	181	1.7k	99.1k	234
CQ-T	348	597	191	1.2k	44.8k	237

Table 1: Basic properties of quasi KGs, averaged over all questions.

all questions (93% in CQ-W and 98% in CQ-T) require aggregating **multi-document evidence**: there are *no single pages* in our corpora that contain *all* the information needed to accurately answer these questions.

5.3 Text Corpora and Quasi KGs

To decouple system performance from the choice of text corpora, we experimented with a number of scenarios for sampling pseudo-relevant documents using Web search results [55]:

- Top-10 documents from Google Web search, where the whole question was issued as a keyword query;
- To weaken the effect of Google's (usually high-quality) ranking from the QA performance, we constructed different settings for *stratified sampling* [61] of 10 documents: we take the top-x_1% of documents from the original top-10 ranking, then sample another x_2% of our pool randomly from ranks $(0.1 * x_1 + 1)$ to 25, then take the remaining x_3% from ranks 26 to 50 (avoiding duplicates wherever applicable), such that $x_1 \geq x_2 \geq x_3$. We use the following five configurations of $x1 - x2 - x3$, with gradually "degrading" ranking quality: $60 - 30 - 10$ (*Strata 1*), $50 - 40 - 10$ (*Strata 2*), $50 - 30 - 20$ (*Strata 3*), $40 - 40 - 20$ (*Strata 4*), and $40 - 30 - 30$ (*Strata 5*).

These varied choices of strata reduce influence of the underlying search engine. For reproducibility w.r.t changing search results, we also make the retrieved corpora available with our data. Stanford CoreNLP [43] was used for POS tagging and NER on all documents.

Similarities and thresholds. Entity similarity scores for alignment edges were computed using the AIDA dictionary [31]. It consists of a large lexicon (an updated version was obtained from the authors of [31]) of (entity, mention) pairs, where a *mention* refers to the surface form in the text (like our *node labels*), and a canonicalized entity is specified by its Wikipedia identifier. The similarity between two mentions is computed as the Jaccard index of the sets of entities they refer to. All other similarities, for relations and types, require *soft matching* and are computed using cosine similarities between 300-dimensional word/phrase embeddings based on word2vec [45]. All three thresholds are set to 0.5: (i) cornerstone selection, (ii) alignment edge insertion, and (iii) answer merging; no tuning is involved. The number of top-k GSTs to use was set to 50 (effect of this choice is examined later in Sec. 7). Note that 50 is a very small fraction of all possible trees in the graph containing the cornerstones (mean = 983, max $\simeq$ 13k). Summary statistics of our noisy quasi KGs are in Table 1.

5.4 Baselines and Metrics

Neural QA. As a strong neural baseline, we select DrQA [12], a very recent open-source QA system. DrQA has large-scale training on SQuAD [48], and is based on recurrent neural networks and multitask learning. It was designed for reading comprehension, but it can select relevant documents from a corpus and extract the best answer span from these documents. DrQA and other baselines run on the same set of input documents that QUEST is exposed to. Specifically, DrQA has two components DocumentRetriever and DocumentReader, which are both run on the top-10 and stratified corpora from Google Web search.

Graph-based algorithms. As a competitor to the GST algorithm of QUEST, we adapted the *breadth-first search* (BFS) phase of the STAR algorithm [38] for entity relatedness in curated KGs. The full STAR method can only work in the presence of a taxonomic backbone in the KG, which is inapplicable in our case. The BFS baseline runs graph-traversal iterators from each terminal node, invoked in a round-robin manner. As soon as the iterators meet, a result is constructed. To respect sets of cornerstones, we require only one iterator from each group of terminals to meet for a candidate answer. Results are ranked in descending order of their weighted distances from the cornerstones in the BFS tree.

To examine the importance of the optimal subgraph identified by the GST, we also compare results using *shortest paths* as follows. We compute shortest paths in the graph between every pair of terminals, where each node in a pair is from a different cornerstone group. Every non-terminal that lies on any shortest path is a candidate answer. An answer is scored by the numbers of different shortest paths that it lies on, and ranked in descending order of these scores. For both BFS and ShortestPaths, *for fairness, answers are post-processed by the same type-based filtering and aggregation* as in QUEST, before applying respective answer ranking strategies.

Metrics. We use the Mean Reciprocal Rank (MRR) as the main metric. We also report other key metrics for QA: Precision@1 (P@1), which measures the fraction of times a correct answer was obtained at rank 1, and Hit@5, which is 1 when one of the top-5 results is a gold answer and 0 otherwise.

6 MAIN RESULTS AND INSIGHTS

We present our main results in Table 2, and discuss key insights from these comparisons. Our main experiments test the postulates:

- QUEST outperforms its neural and graph baselines;
- Performance of QUEST is robust to corpus perturbations;
- Multi-document evidence is vital for retrieving correct answers;
- Group Steiner Trees are key to locating answers in quasi KGs.

Systematic improvement over state-of-the-art. Looking at the "Top" and "Strata" columns (Sec. 5.3) for both benchmarks, we find that QUEST significantly and consistently outperforms the neural baseline DrQA, and other graph-based methods, at almost all settings. This performance of QUEST is clearly robust to variations in the underlying corpus. We attribute the success of the proposed method to its unique ability to stitch facts from more than one source, and the powerful GST algorithm that discovers answers in the large and very noisy quasi KGs. The task of fetching crisp text answers to complex questions directly over Web corpora is generally a very difficult one; this is reflected by relatively low values of MRR (best numbers of 0.328 for CQ-W and 0.355 for CQ-T in top-10 corpora). This also shows ample room for improvement on these challenging benchmarks.

Method	**Metric**	**ComplexQuestions from WikiAnswers (CQ-W)**						**ComplexQuestions from Trends (CQ-T)**					
		Top	**Strata-1**	**Strata-2**	**Strata-3**	**Strata-4**	**Strata-5**	**Top**	**Strata-1**	**Strata-2**	**Strata-3**	**Strata-4**	**Strata-5**
QUEST	MRR	**0.328***	**0.329***	**0.303***	**0.303***	**0.331***	**0.315***	**0.355***	**0.336***	**0.305***	**0.322***	**0.302***	**0.284***
DrQA [12]	MRR	0.253	0.247	0.263	0.235	0.229	0.246	0.157	0.178	0.241	0.227	0.209	0.232
BFS [38]	MRR	0.249	0.256	0.266	0.212	0.219	0.254	0.287	0.256	0.265	0.259	0.219	0.201
ShortestPaths	MRR	0.240	0.261	0.249	0.237	0.259	0.270	0.266	0.224	0.248	0.219	0.232	0.222
QUEST	P@1	**0.267***	**0.253***	**0.227**	**0.220***	**0.240***	**0.213**	**0.300***	**0.280***	**0.250***	**0.260***	**0.270***	**0.230***
DrQA [12]	P@1	0.193	0.200	0.207	0.167	0.160	0.193	0.090	0.110	0.140	0.160	0.130	0.160
BFS [38]	P@1	0.160	0.167	0.193	0.113	0.100	0.147	0.210	0.170	0.180	0.180	0.140	0.130
ShortestPaths	P@1	0.147	0.173	0.193	0.140	0.147	0.187	0.190	0.140	0.160	0.160	0.150	0.130
QUEST	Hit@5	**0.393***	**0.420***	**0.400***	**0.413***	**0.427***	**0.453***	**0.450***	**0.410***	**0.380**	**0.420***	0.340	0.340
DrQA [12]	Hit@5	0.347	0.340	0.353	0.313	0.327	0.340	0.250	0.290	0.360	0.320	**0.350**	**0.350**
BFS [38]	Hit@5	0.360	0.353	0.347	0.327	0.327	0.360	0.380	0.360	0.370	0.360	0.310	0.320
ShortestPaths	Hit@5	0.347	0.367	0.387	0.327	0.393	0.340	0.350	0.320	0.340	0.310	0.330	0.290

Table 2: Performance comparison of methods on top-10 and stratified search results from the Web. For every metric, the best value per column is in **bold**. "$*$" denotes statistical significance of QUEST over DrQA, with p-value ≤ 0.05 for a two-tailed paired t-test.

Effect of corpus variations. Exact reliance on Google's top-10 Web search results is not a prerequisite: we show this by weakening the search ranking with stratified sampling (as discussed in Sec. 5.3) by intentionally introducing controlled amounts of noisy pages in the pseudorelevant corpora. Results are in columns labeled "Strata-1" through "Strata-5" in Table 2. The key observation is the across-the-board superiority of QUEST, and that it was able to cope well with this injected noise. Note that Google's ranking may not always be perfect, as the stratified configuration $60-30-10$ (Strata 1) resulted in slightly better performance than the top-10 (Hit@5 of 0.420 vs. 0.393 on CQ-W for QUEST). We also experimented with the setting where search was restricted to Google News, and observed similar trends (0.261 MRR for QUEST vs. 0.227 for DrQA on top-10, aggregated over CQ-W and CQ-T). Google search over Wikipedia only turned out in favor of DrQA (0.244 MRR vs. 0.189 for QUEST, top-10). This is due to the low redundancy of facts in Wikipedia, that hurts QUEST (explained shortly), and the Wikipedia-specific training of DrQA.

Usage of multi-document evidence. QUEST improved over DrQA on both benchmarks (MRR of 0.328 vs. 0.253 on CQ-W; 0.355 vs. 0.157 on CQ-T), even though the latter is a supervised deep learning method trained on the large SQuAD dataset. This is because reading comprehension (RC) QA systems search for the best answer span within a passage, and will not work well unless the passage matches the question tokens and contains the answer. While DrQA can additionally select a good set of documents from a collection, it still relies on the best document to extract the answer from. QUEST, by joining fragments of evidence across documents via GSTs, thus improves over DrQA without any training or tuning. QUEST benefits from multi-document evidence in two ways:

- Confidence in an answer increases when all conditions for correctness are indeed satisfiable (and found) only when looking at multiple documents. This increases the answer's likelihood of appearing in *some GST*.
- Confidence in an answer increases when it is spotted in multiple documents. This increases its likelihood of appearing in the *top-k* GSTs, as presence in multiple documents increases weights and lowers costs of the corresponding edges.

A detailed investigation of the use of multi-document information is presented in Table 3. We make the following observations: (i) Looking at the "Avg. #Docs in GST" columns in the upper half, we see that considering the top-50 GSTs is worthwhile as all the bins combine evidence from multiple (2+ on average) documents. This is measured by labeling edges in GSTs with documents (identifiers) that contribute the corresponding edges. (ii) Moreover, they also contain the correct answer uniformly often (corresponding "#Q's with A in GST" columns; $48-57$ for CQ-W, $51-55$ for CQ-T). (iii) The bottom half of the table inspects the inverse phenomenon, and finds that GSTs that contain information from fewer documents are likely to be ranked relatively higher, again justifying the consideration of top-50 GSTs. (iv) Finally, there is a sweet spot for GSTs aggregating nuggets from multiple documents to contain correct answers, and this turns out to be around three documents (see corresponding "#Q's with A in GST" columns). This, however, is a likely result of the questions in our benchmarks, that are not complex enough to require stitching evidence across more than three documents.

Deep-learning-based QA methods over text can handle syntactic complexity very well, but are typically restricted to identifying answer spans from a single text passage. DrQA could not properly tap the answer evidence that comes from combining cues spread across multiple documents.

Impact of Steiner Trees. Observing the graph-based methods, QUEST is clearly better than both BFS and ShortestPaths, obtaining correct answers at better ranks. This demonstrates that computing cost-optimal GSTs is indeed crucial, and cannot be easily approximated by simpler methods. It is not simply the connectivity alone that qualifies a node as a good answer, which is why the simpler ShortestPaths method substantially loses against QUEST. Computing the GST can be viewed as a *joint disambiguation* for all the semantic items in the question, like entities and predicates in a KG.

Anecdotal results. To highlight the complexity of questions that are within our reach, Table 4 shows representatives where QUEST had the correct answer at the very first rank, but all the baselines failed. Note that some of the questions refer to long-tail entities not covered yet by KGs like Wikidata, and also have predicates like *met, split*, and *dated*, which are beyond the scope of KGs.

Benchmark	**CQ-W**		**CQ-T**	
GST Ranks	**Avg. #Docs in GST**	**#Q's with A in GST**	**Avg. #Docs in GST**	**#Q's with A in GST**
01 − 10	2.637	48	3.139	52
11 − 20	2.789	56	3.156	53
21 − 30	2.778	54	3.245	55
31 − 40	2.833	54	3.267	54
41 − 50	2.882	57	3.278	51
#Docs in GST	**Avg. Rank of GST**	**#Q's with A in GST**	**Avg. Rank of GST**	**#Q's with A in GST**
1	24.525	12	22.124	7
2	27.003	37	24.216	23
3	25.777	53	27.165	50
4	27.064	36	27.069	49
5	29.291	25	26.554	29

Table 3: Effect of multi-document evidence shown via edge contributions by distinct documents to GSTs (on top-10 corpora).

Questions from CQ-W ($P@1 = 1$)
Q: *"which actor is married to kaaren verne and played in casablanca?"* **A**: *'Peter Lorre'*
Q: *"what river flows through washington and oregon?"* **A:** *'Columbia River'*
Q: *"what movie did russell crowe and denzel washington work on together?"* **A**: *'American Gangster'*
Questions from CQ-T ($P@1 = 1$)
Q: *"where did sylvie vartan meet her future husband johnny hallyday?"* **A**: *'Paris Olympia Hall'*
Q: *"which aspiring model split with chloe moretz and is dating lexi wood?"* **A:** *'Brooklyn Beckham'*
Q: *"which japanese baseball player was contracted for los angeles angels who also played for hokkaido nippon-ham fighters?"* **A:** *'Shohei Ohtani'*

Table 4: Examples of correctly answered questions by QUEST but not by any of the baselines (on top-10 corpora).

7 ANALYSIS AND DISCUSSION

Graph ablation experiments. The noisy graph is the backbone of QUEST and has several components working in tandem. We systematically analyzed this interplay of components by deactivating each separately (Table 5). The key insight is that type nodes and edges are essential to the success of QUEST; removing them results in degraded performance. Next, using informative edge weights driven by document proximity and alignment levels, are another vital element (MRR drops with degenerate edge weights). Removing alignment edges also adversely affects performance.

Answer ranking variants. Our answer ranking strategy is motivated by considering a weighted (with reciprocal of tree cost) sum for exploiting answer presence in multiple GSTs. Nevertheless, we explored various alternatives to this choice, and observed (Table 5): (i) just counting GSTs in which an answer is present is not enough; (ii) reusing node weights for scoring trees with answers does not really help; (iii) There is no additional benefit in zooming in to consider the position of an answer *within a GST*; nodes that are the closest to cornerstones are not necessarily the best answers. However, differences between the first four choices are not statistically significant: hence, QUEST is robust to slight ranking variants as long as answer evidence across multiple GSTs is considered.

Error analysis. Failure cases for QUEST, and corresponding occurrences are shown in Table 5. We treat a case as an error when QUEST cannot locate any correct answer in the top-5 (Hit@5= 0). The first case suggests use of a better retrieval model, considering semantic matches. The second is a key reason for failure, and demands an Open IE extractor with better fact recall. The third scenario is mostly caused by matching wrong groups of cornerstones. For example, matching relation *'born'* in question with *'lived'* in the quasi KG; or *'play'* (drama) intended as a noun, matching a relation node *'played'* (role). This can be improved with better NER, lexicons, and similarity functions. Case (iv) happens due to pruning by type mismatch. This calls for more informed prediction and matching of expected answer types, and improving type extraction from documents. Situation (v) indicates that improving the ranking function is the most worthwhile effort for improving performance.

Effect of Open IE. QUEST's noisy triple extractor results in quasi KGs that contain the correct answer 51.3% of the times for CQ-W (42.7% for CQ-T). If we use triples from Stanford OpenIE [3] to build the quasi KG instead, the answer is found only 46.7% and 35.3% times for CQ-W and CQ-T, respectively. Thus, in the context of QA, losing information with precision-oriented, fewer triples, definitely hurts more than the adding potentially many noisy ones.

Answering simple questions. On the popular WebQuestions benchmark [7], QUEST was able to find a correct answer 54% of the time, which is respectable in comparison to the best QA systems specifically trained for this setting of simple questions. GSTs are augmented with 1-hop neighbors of terminals to handle two-cornerstone (single-entity single-relation) questions.

Effect of threshold variations. There are three parameters in QUEST: Number of GSTs k, the alignment edge insertion threshold on node-node similarity, and cornerstone selection thresholds on node weight. Variation of these parameters are shown with MRR at cut-off ranks $r = 1, 3, 5$ in Fig. 2. We observe that: (i) going beyond the chosen value of $k = 50$ gives only diminishing returns; (ii) Fig. 2b shows that having several alignment edges in the graph (corresponding to a very low threshold), actually helps improve performance though apparently inviting noise; (iii) QUEST is not really sensitive to cornerstone selection thresholds – the dark zone, indicating good performance, is towards the interiors of the grid, but broadly spread out: so most choices of non-extreme thresholds will work out fine. The white zone in the top right corner corresponds to setting both thresholds to very high values (no cornerstones chosen, resulting in zero performance).

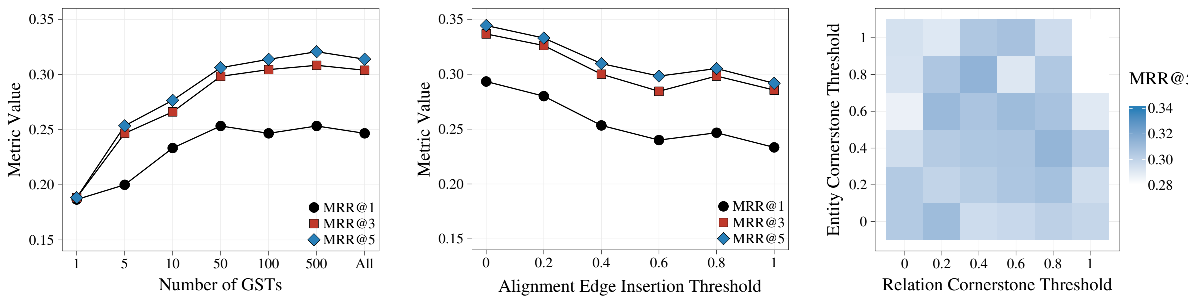
Run-time. QUEST computes answers with interactive response times: the median run-time for computing GSTs was 1.5 seconds (with a mean of about five seconds). All experiments were performed with Java 8 on a Linux machine (RHEL-v6.3) using an Intel Xeon E5 CPU with 256 GB RAM.

8 RELATED WORK

QA over text. Classical approaches [49] extracted answers from passages that matched most cue words from the question followed

Graph configuration	**CQ-W**	**CQ-T**	**Answer ranking criterion**	**CQ-W**	**CQ-T**	**Error scenario**	**CQ-W**	**CQ-T**
Full configuration	**0.355**	**0.467**	Wted. sum of GSTs (inv. tree cost sum)	**0.355**	**0.467**	Ans. not in corpus	1%	7%
No types	0.321	0.384*	Wted. sum of GSTs (node wt. sum)	0.318	0.426	Ans. in corpus but not in quasi KG	23%	30%
Degenerate edge weights	0.282*	0.377*	Count of GSTs	0.334	0.432	Ans. in quasi KG but not in top-50 GSTs	10%	6%
No entity alignment	0.329	0.413	Wted. dist. to cornerstones	0.321	0.417	Ans. in top-50 GSTs but not in candidates	1%	7%
No predicate alignment	0.337	0.403	Unwted. dist. to cornerstones	0.257*	0.372*	Ans. in candidates but not in top-5	66%	49%

Table 5: Understanding QUEST's mechanism with top-10 corpora. **Left:** Graph ablation (MRR); **Middle:** Answer ranking (MRR); **Right:** Error analysis (Hit@5 = 0). Highest column values in the first two sections in **bold**. Significant drops from these values are shown with *.



(a) No. of GSTs (k) **(b)** Alignment edge insertion threshold **(c)** Cornerstone selection thresholds

Figure 2: Robustness of QUEST to various system configuration parameters, on CQ-W with top-10 corpora (similar trends on CQ-T).

by statistical scoring. TREC ran a QA benchmarking series from 1999 to 2007, recently revived it as the LiveQA track. IBM Watson [25] extended this paradigm by combining it with learned models for special question types.

QA over KGs. The advent of large knowledge graphs like Freebase [9], YAGO [53], DBpedia [4] and Wikidata [63] has given rise to QA over KGs (overviews in [18, 59]). The goal is to translate a natural language question into a structured query, typically in the Semantic Web language SPARQL, that directly operates on the entities and predicates of the underlying KG [18, 59]. Early work on KG-QA built on paraphrase-based mappings and query templates that cover simple forms of questions that involve a single entity predicate [7, 11, 23, 58, 67]. This line was further advanced by [1, 5, 6, 24, 32], including the learning of templates from graph patterns in the KG. However, reliance on templates prevents such approaches from robustly coping with arbitrary syntactic formulations. This has motivated deep learning methods with CNNs and LSTMs [21, 33, 57, 66, 69]. These have been most successful on benchmarks like WebQuestions [7] and QALD [60]. However, all these methods critically build on sufficient amounts of training data in the form of question-answer pairs. In contrast, QUEST is fully unsupervised and neither needs templates nor training data.

QA over hybrid sources. Limitations of QA over KGs has led to a revival of considering textual sources, in combination with KGs [50, 54, 66]. Some methods like PARALEX [23] and OQA [24] supported noisy KGs in the form of triple spaces compiled via Open IE [27, 44] on Wikipedia articles or Web corpora. TupleInf [39] extended and generalized PARALEX to complex questions, but is limited to *multiple-choice answer options* and is thus inapplicable for our task. TAQA [70] is another generalization of Open-IE-based QA, by constructing a KG of n-tuples from Wikipedia full-text and question-specific search results. Unfortunately this method is restricted to questions with prepositional and adverbial constraints only. [56] addressed complex questions by decomposing them into a sequence of simple questions, but relies on training data obtained via Amazon Mechanical Turk. Some methods start with KGs as a source for candidate answers and use text corpora like Wikipedia or ClueWeb as additional evidence [15, 54, 66], or start with answer sentences from text corpora and combine these with KGs for entity answers [50, 55]. Most of these are based on neural networks, and are only designed for *simple* questions like those in the WebQuestions, SimpleQuestions, or WikiMovies benchmarks. In contrast, QUEST can handle arbitrary kinds of complex questions and can construct *explanatory evidence* for its answers – an unsolved concern for neural methods.

Reading comprehension. This is a QA variation where a question needs to be answered from a given text paragraph [34, 48]. This is different from the fact-centric answer-finding task considered here, with input from dynamically retrieved documents. Nevertheless, we compared with, and outperformed, the state-of-the-art system DrQA [12], which can both select relevant documents and extract answers from them. Traditional fact-centric QA over text, and multi-document reading comprehension are recently emerging as a joint topic referred to as open-domain question answering [16, 42].

9 CONCLUSION

We presented QUEST, an unsupervised method for QA over dynamically retrieved text corpora based on Group Steiner Trees. QUEST substantially outperforms DrQA, a strong deep learning baseline, on challenging benchmarks. As noisy content is unavoidable with Web content and ad hoc questions for which extensive training is infeasible, QUEST deliberately allows noise in its computational pipeline, and copes with it using cross-document evidence, smart answer detection, and graph-based ranking strategies. Adapting QUEST to work over combinations of text-based quasi KGs and curated KGs will be the focus of future studies.

REFERENCES

[1] A. Abujabal, M. Yahya, M. Riedewald, and G. Weikum. 2017. Automated template generation for question answering over knowledge graphs. In *WWW*.
[2] E. Agichtein, D. Carmel, D. Pelleg, Y. Pinter, and D. Harman. 2015. Overview of the TREC 2015 LiveQA Track. In *TREC*.
[3] G. Angeli, M. J. J. Premkumar, and C. D. Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *ACL*.
[4] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. 2007. DBpedia: A nucleus for a Web of open data. In *The Semantic Web*. Springer.
[5] J. Bao, N. Duan, Z. Yan, M. Zhou, and T. Zhao. 2016. Constraint-Based Question Answering with Knowledge Graph. In *COLING*.
[6] H. Bast and E. Haussmann. 2015. More accurate question answering on Freebase. In *CIKM*.
[7] J. Berant, A. Chou, R. Frostig, and P. Liang. 2013. Semantic Parsing on Freebase from Question-Answer Pairs. In *ACL*.
[8] G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan. 2002. Keyword searching and browsing in databases using BANKS. In *ICDE*.
[9] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *SIGMOD*.
[10] A. Bordes, N. Usunier, S. Chopra, and J. Weston. 2015. Large-scale simple question answering with memory networks. In *arXiv*.
[11] Q. Cai and A. Yates. 2013. Large-scale Semantic Parsing via Schema Matching and Lexicon Extension. In *ACL*.
[12] D. Chen, A. Fisch, J. Weston, and A. Bordes. 2017. Reading Wikipedia to Answer Open-Domain Questions. In *ACL*.
[13] J. Coffman and A. C. Weaver. 2014. An empirical performance evaluation of relational keyword search techniques. In *TKDE*.
[14] L. Del Corro and R. Gemulla. 2013. ClausIE: Clause-based open information extraction. In *WWW*.
[15] R. Das, M. Zaheer, S. Reddy, and A. McCallum. 2017. Question Answering on Knowledge Bases and Text using Universal Schema and Memory Networks. In *ACL*.
[16] M. Dehghani, H. Azarbonyad, J. Kamps, and M. de Rijke. 2019. Learning to Transform, Combine, and Reason in Open-Domain Question Answering. In *WSDM*.
[17] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv* (2018).
[18] D. Diefenbach, V. López, K. Deep Singh, and P. Maret. 2018. Core techniques of question answering systems over knowledge bases: A survey. *Knowl. Inf. Syst.* 55, 3 (2018).
[19] L. Dietz and B. Gamari. 2017. TREC CAR: A Data Set for Complex Answer Retrieval. In *TREC*.
[20] B. Ding, J. X. Yu, S. Wang, L. Qin, X. Zhang, and X. Lin. 2007. Finding top-k min-cost connected trees in databases. In *ICDE*.
[21] L. Dong, F. Wei, M. Zhou, and K. Xu. 2015. Question Answering over Freebase with Multi-Column Convolutional Neural Networks. In *ACL*.
[22] R. G. Downey and M. R. Fellows. 2013. *Fundamentals of Parameterized Complexity*. Springer.
[23] A. Fader, L. Zettlemoyer, and O. Etzioni. 2013. Paraphrase-driven learning for open question answering. In *ACL*.
[24] A. Fader, L. Zettlemoyer, and O. Etzioni. 2014. Open question answering over curated and extracted knowledge bases. In *KDD*.
[25] D. Ferrucci et al. 2012. This is Watson. *IBM Journal Special Issue on IBM Watson* 56, 3 (2012).
[26] N. Garg, G. Konjevod, and R. Ravi. 2000. A Polylogarithmic Approximation Algorithm for the Group Steiner Tree Problem. *J. Algorithms* 37, 1 (2000).
[27] K. Gashteovski, R. Gemulla, and L. Del Corro. 2017. MinIE: Minimizing Facts in Open Information Extraction. In *EMNLP*.
[28] A. Grycner and G. Weikum. 2016. POLY: Mining Relational Paraphrases from Multilingual Sentences. In *EMNLP*.
[29] Ido Guy. 2018. The Characteristics of Voice Search: Comparing Spoken with Typed-in Mobile Web Search Queries. *ACM Trans. Inf. Syst.* (2018).
[30] M. A. Hearst. 1992. Automatic Acquisition of Hyponyms from Large Text Corpora. In *COLING*.
[31] J. Hoffart, M. A. Yosef, I. Bordino, H. Fürstenau, M. Pinkal, M. Spaniol, B. Taneva, S. Thater, and G. Weikum. 2011. Robust disambiguation of named entities in text. In *EMNLP*.
[32] S. Hu, L. Zou, J. X. Yu, H. Wang, and D. Zhao. 2018. Answering Natural Language Questions by Subgraph Matching over Knowledge Graphs. *Trans. on Know. and Data Eng.* 30, 5 (2018).
[33] M. Iyyer, J. L. Boyd-Graber, L. M. B. Claudino, R. Socher, and H. Daumé III. 2014. A Neural Network for Factoid Question Answering over Paragraphs. In *EMNLP*.
[34] M. Joshi, E. Choi, D. S. Weld, and L. Zettlemoyer. 2017. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. In *ACL*.
[35] M. Joshi, U. Sawant, and S. Chakrabarti. 2014. Knowledge graph and corpus driven segmentation and answer inference for telegraphic entity-seeking queries. In *EMNLP*.
[36] V. Kacholia, S. Pandit, S. Chakrabarti, S. Sudarshan, R. Desai, and H. Karambelkar. 2005. Bidirectional expansion for keyword search on graph databases. In *VLDB*.
[37] A. Kadry and L. Dietz. 2017. Open Relation Extraction for Support Passage Retrieval: Merit and Open Issues. In *SIGIR*.
[38] G. Kasneci, M. Ramanath, M. Sozio, F. M. Suchanek, and G. Weikum. 2009. STAR: Steiner-tree approximation in relationship graphs. In *ICDE*.
[39] T. Khot, A. Sabharwal, and P. Clark. 2017. Answering complex questions using open information extraction. In *ACL*.
[40] C. Kwok, O. Etzioni, and D. S. Weld. 2001. Scaling Question Answering to the Web. *ACM Trans. Inf. Syst.* (2001).
[41] R. Li, L. Qin, J. X. Yu, and R. Mao. 2016. Efficient and progressive group steiner tree search. In *SIGMOD*.
[42] Y. Lin, H. Ji, Z. Liu, and M. Sun. 2018. Denoising distantly supervised open-domain question answering. In *ACL*.
[43] C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, and D. McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *ACL*.
[44] Mausam. 2016. Open information extraction systems and downstream applications. In *IJCAI*.
[45] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *NIPS*.
[46] E. Pavlick, P. Rastogi, J. Ganitkevitch, B. Van Durme, and C. Callison-Burch. 2015. PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *ACL*.
[47] J. Pennington, R. Socher, and C. D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *EMNLP*.
[48] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *EMNLP*.
[49] D. Ravichandran and E. Hovy. 2002. Learning Surface Text Patterns for a Question Answering System. In *ACL*.
[50] D. Savenkov and E. Agichtein. 2016. When a knowledge base is not enough: Question answering over knowledge bases with external text data. In *SIGIR*.
[51] U. Sawant and S. Chakrabarti. 2013. Learning joint query interpretation and response ranking. In *WWW*.
[52] V. I. Spitkovsky and A. X. Chang. 2012. A Cross-Lingual Dictionary for English Wikipedia Concepts. In *LREC*. 3168–3175.
[53] F. M. Suchanek, G. Kasneci, and G. Weikum. 2007. YAGO: A core of semantic knowledge. In *WWW*.
[54] H. Sun, B. Dhingra, M. Zaheer, K. Mazaitis, R. Salakhutdinov, and W. W. Cohen. 2018. Open domain question answering using early fusion of knowledge bases and text. In *EMNLP*.
[55] H. Sun, H. Ma, W. Yih, C. Tsai, J. Liu, and M. Chang. 2015. Open domain question answering via semantic enrichment. In *WWW*.
[56] A. Talmor and J. Berant. 2018. The Web as a Knowledge-base for Answering Complex Questions. In *NAACL-HLT*.
[57] C. Tan, F. Wei, Q. Zhou, N. Yang, B. Du, W. Lv, and M. Zhou. 2018. Context-Aware Answer Sentence Selection With Hierarchical Gated Recurrent Neural Networks. *IEEE/ACM Trans. Audio, Speech & Language Processing* 26, 3 (2018).
[58] C. Unger, L. Bühmann, J. Lehmann, A. N. Ngomo, D. Gerber, and P. Cimiano. 2012. Template-based question answering over RDF data. In *WWW*.
[59] C. Unger, A. Freitas, and P. Cimiano. 2014. An Introduction to Question Answering over Linked Data. In *Reasoning Web*.
[60] R. Usbeck, A. N. Ngomo, B. Haarmann, A. Krithara, M. Röder, and G. Napolitano. 2017. 7th Open Challenge on Question Answering over Linked Data (QALD-7). In *Proc. SemWebEval*.
[61] E. M. Voorhees. 2014. The Effect of Sampling Strategy on Inferred Measures. In *SIGIR*.
[62] E. M. Voorhees and D. K. Harman. 2005. *TREC: Experiment and evaluation in information retrieval*. MIT press Cambridge.
[63] D. Vrandečić and M. Krötzsch. 2014. Wikidata: A free collaborative knowledge base. *Commun. ACM* 57, 10 (2014).
[64] R. W. White, M. Richardson, and W. T. Yih. 2015. Questions vs. queries in informational search tasks. In *WWW*. 135–136.
[65] J. Wieting, M. Bansal, K. Gimpel, and K. Livescu. 2016. Towards universal paraphrastic sentence embeddings. In *ICLR*.
[66] K. Xu, S. Reddy, Y. Feng, S. Huang, and D. Zhao. 2016. Question answering on freebase via relation extraction and textual evidence. In *ACL*.
[67] M. Yahya, K. Berberich, S. Elbassuoni, M. Ramanath, V. Tresp, and G. Weikum. 2012. Natural Language Questions for the Web of Data. In *EMNLP*.
[68] M. Yahya, S. Whang, R. Gupta, and A. Halevy. 2014. ReNoun: Fact extraction for nominal attributes. In *EMNLP*.
[69] W. Yih, M. Chang, X. He, and J. Gao. 2015. Semantic Parsing via Staged Query Graph Generation: Question Answering with Knowledge Base. In *ACL*.
[70] P. Yin, N. Duan, B. Kao, J. Bao, and M. Zhou. 2015. Answering Questions with Complex Semantic Constraints on Open Knowledge Bases. In *CIKM*.
[71] J. X. Yu, L. Qin, and L. Chang. 2009. *Keyword Search in Databases*. Morgan & Claypool.
[72] D. Ziegler, A. Abujabal, R. Saha Roy, and G. Weikum. 2017. Efficiency-aware Answering of Compositional Questions using Answer Type Prediction. In *IJCNLP*.