Open-Domain Question Answering Using a Knowledge Graph and Web Corpus

Uma Sawant, Soumen Chakrabarti and Ganesh Ramakrishnan
IIT Bombay

In Web search, entity-seeking queries often trigger a special Question Answering (QA) system. It uses syntactic patterns to extract structure from the query, identify a semantic interpretation and then return direct entity responses from a knowledge graph (KG). Such QA systems tend to be brittle. Minor query variations may fail to trigger the QA system. Moreover, KG coverage is patchy at best. Rather than fall off the "structure cliff" in such cases, we propose a more robust approach that degrades gracefully on a "structure ramp". Our system, called AQQUCN, accepts a broad spectrum of queries, between well-formed questions to short "telegraphic" keyword sequences. In the face of inherent query ambiguities, AQQUCN aggregates signals from KGs and large corpora to directly rank KG entities, rather than commit to one semantic interpretation of the query. AQQUCN models the ideal interpretation as an unobservable or latent variable. Pairs of interpretations and candidate entity responses are scored as pairs, by combining signals from multiple convolutional networks that operate collectively on the query, KG and corpus. On four public query workloads amounting to over 8,000 queries in different query formats, we see 16–18% absolute improvement in mean average precision (MAP), compared to recent systems. Our system is also competitive when compared to recent KBQA systems.

DOI: 10.1145/3183639.3183643 http://doi.acm.org/10.1145/3183639.3183643

1. INTRODUCTION

A large fraction of Web queries involve and seek entities [Lin et al. 2012]: queries seeking details of celebrities or movies (e.g., kingsman release date), historical events (e.g., Who killed Gandhi?), travel (e.g., nearest airport to baikal lake), to name a few. Queries that match certain patterns are handed off to specialized QA systems that directly return entity responses from a KG. Often, a semantic parse of the textual query is attempted [Berant et al. 2013; Yih et al. 2015] to translate it to a structured query over the KG, which is then executed to fetch a *set* of response entities [1]. Although providing precise answers when everything goes well, this approach to entity-driven QA is fraught with several difficulties.

- The input textual query may range from grammatically well-formed questions (e.g., In which band did Jimmy Page perform before Led Zeppelin?) to free-form "telegraphic" keyword queries (e.g., band jimmy page was in before led zeppelin). QA systems are often brittle with regard to input syntax, backing off if the input does not

[1] These are known as KBQA or "Knowledge Base Question Answering" systems.

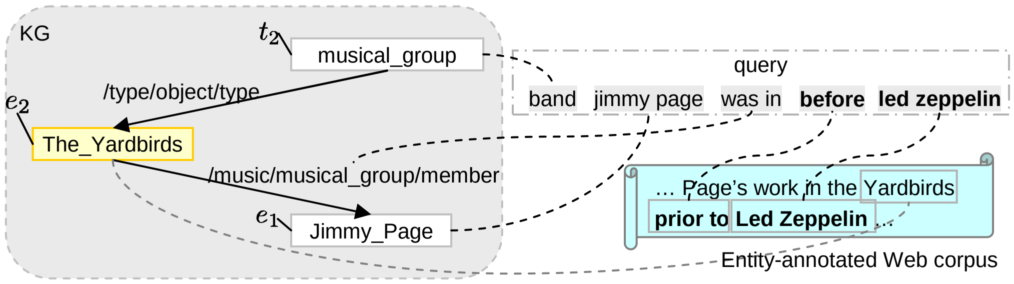
KG
t_2
musical_group
/type/object/type
e_2
The_Yardbirds
/music/musical_group/member
e_1
Jimmy_Page
query
band jimmy page was in **before** **led zeppelin**
... Page's work in the Yardbirds
prior to **Led Zeppelin** ...
Entity-annotated Web corpus

Fig. 1. QA example using KG and corpus. Parts of the query have different *roles* and match diverse artifacts in the KG and corpus, requiring a complex flow of evidence toward the correct response.

match specific syntactic patterns.

- A curated, structured collection of facts in a KG reduces the QA task to "compiling" the textual query into a structured form which is directly executed on the KG. But KG coverage is always patchy — with nodes and/or edges missing — particularly when less popular entities are concerned. Over 70% of people in Freebase do not have a place of birth in Freebase [West et al. 2014]. If the types and relations expressed textually in the query cannot be mapped confidently to the KG, most QA systems back off.
- Alternatively, one can extend IR-style text search by using an entity-annotated corpus of Web pages. Any text snippet s in the corpus which contains surface form of an entity e and also matches the query q; is considered supporting evidence for that entity e to be the answer for q. However, such evidence from the Web corpus can be noisy due to incorrect entity linking of e in s and imperfect text matching between q and s.

Thus, both the KG and the entity-annotated corpus present challenges when used for QA. Yet, they can be used to reinforce or complement information when used together, as we illustrate next.

2. EXAMPLE QUERY AND RESPONSE

Figure 1 demonstrates the advantages and complexities of effective entity-level QA involving both KG and corpus. Tokens in the query have diverse, possibly overlapping *roles*. Specifically, a query span may hint at an entity, type or relation, or it can be used to match passages in the corpus. Understanding the roles and disambiguating the hint to respective semantic nodes in the KG (wherever applicable) helps interpret that query. For example, the query band jimmy page was in before led zeppelin has a reference to entities e_1 = `Jimmy_Page` and `Led_Zeppelin`. Mentions in both the query and corpus documents are linked to entity nodes in the KG (e.g. jimmy page). Band hints at target type t_2 = `musical_group` of the expected answer entity e_2 = `The_Yardbirds`. The (rather weak) hint was in hints at the relation `/music/musical_group/member` connecting e_1, e_2. Thus, identifying e_1, r and t_2 can lead us to many candidate e_2s, `The_Yardbirds` being one of them. Yet, the query interpretation is not complete because an important token 'before' is not considered. If the KG does not have timestamps on membership, or the QA engine cannot do arithmetic with timestamps, passages in the corpus can still offer supplementary evidence by matching before with prior to, along with

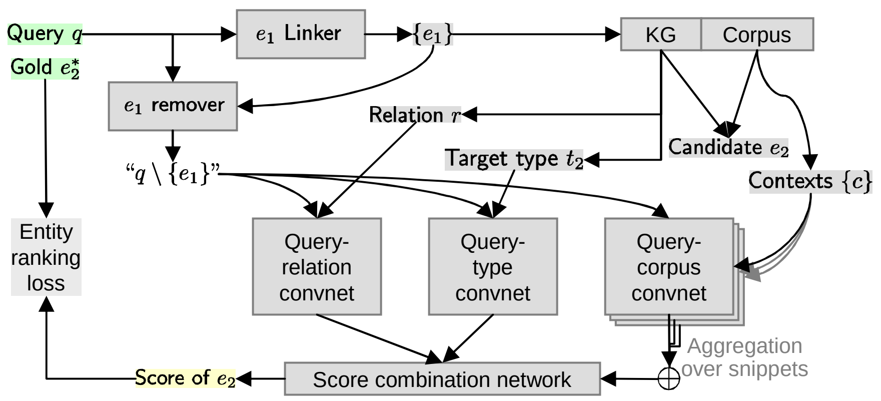

Fig. 2. AQQUCN block diagram.

mentions of e_1 and e_2. Thus, using both KG and corpus allows combining structured and unstructured evidence to answer a query.

This example also serves to highlight our challenges. The dotted lines in Figure 1 map the query hints to the KG or the corpus evidence, either created during pre-processing stage (e.g. entity linking in the corpus) or at run-time (e.g. matching the target type $t_2 = \texttt{musical\_group}$ to the query text Band through a type model). The machine-learnt models which map the hints to KG entities, types or relations need to handle a lot of ambiguity, as the same hint may match many correct/incorrect KG artifacts. Thus, there may be multiple KG subgraphs and corpus text snippets, each appearing to support different correct or incorrect entity candidates. There is thus a clear need for robust and seamless aggregation of supporting evidence across corpus and KG.

3. AQQUCN

AQQUCN, a system we have built based on AQQU [Bast and Haußmann 2015], implements all the inference pathways shown in Figure 1. A system sketch is shown in Figure 2. Unlike AQQU and other systems, the end goal of AQQCN is not to "compile" the input into a structured query to execute on the KG, because broken/missing input syntax can make this attempt fail in brittle ways. Instead, the end goal of AQQUCN is to directly create a *ranking* over entities using KG and corpus.

AQQUCN first extracts entities e_1 directly mentioned in the query q [Cornolti et al. 2014]. Each e_1 is located in the KG and the entity-annotated corpus. Entities within a small graph neighborhood in KG become candidate answers, i.e., e_2s. Any e_2 occurring in textual context snippets c from the corpus become additional candidates. Each e_2 is associated with one or more tuples of target types t_2, relation r connecting e_1, e_2, and textual contexts c from the corpus, creating a candidate query interpretation. Candidate e_2s paired with corresponding interpretations are sent to three kinds of convolutional networks (convnets), along with q (with e_1 span/s marked or removed). The overall function of the convnets is to score the match between the query and the interpretation.

Two of the convnets are similar in form: the query-relation network (QRN) and query-type network (QTN). QRN tries to find a hint of r in $q \setminus \{e_1\}$. QTN tries to find a hint of t_2

in $q \setminus \{e_1\}$. Unlike earlier work [Joshi et al. 2014], we do not seek hard segmentations of q. The same query segment can hint towards r and t_2 (e.g., band is primarily a type hint, but can also act as a useful relation hint of `/music/musical_group/member`). Figure 3 shows the architecture of this multi-label prediction network; which consists of an embedding layer, convolutional layer, pooling layer, a fully connected layer, followed by a sigmoid activation function layer that outputs a score for each type. Multiple types can be given large scores. Apart from the automatically computed features by the convolutional and pooling layers, we also compute simple word match features between the query text and the type/relation description text; a strategy quite useful in case of words not seen while training QTN and QRN.

The third query-corpus network (QCN) [Severyn and Moschitti 2015] is slightly different. It measures the similarity between $q \setminus \{e_1\}$ and snippets and has one copy instantiated for each corpus snippet mentioning e_2. Scores from all snippets are pooled together (sum works well).

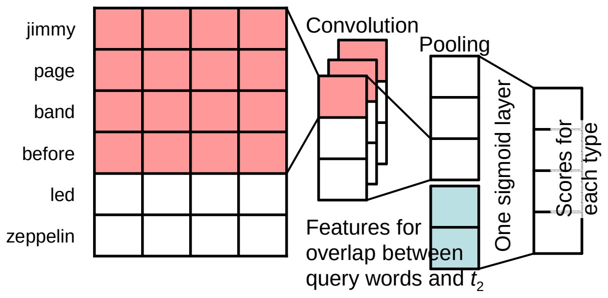

Fig. 3. Query-Type Network (QTN) architecture and inputs. Query-Relation Network (QRN) architecture is identical except that types are replaced by relations in the training data.

QRN, QTN, and multiple QCNs send their scores as features to a final combination network that represents each candidate e_2 as a feature vector and scores it in conjunction with e_1, r and t_2. We can marginalize over e_1, r, t_2 in various ways. Max works well, giving the score of e_2 and thereby a ranking among candidate e_2s. The set of gold e_2^+s is then used to define a loss and train the combination network. We use a pairwise loss, comparing, for a fixed query q, a relevant entity e_2^+ with an irrelevant entity e_2^-:

$$\max_{e_1^+, t_2^+, r^+} w \cdot \phi(q, e_1^+, t_2^+, r^+; e_2^+) + \xi_{q, e_2^+, e_2^-} \geq 1 + \max_{e_1^-, t_2^-, r^-} w \cdot \phi(q, e_1^-, t_2^-, r^-; e_2^-) \quad (1)$$

Note that the best supporting e_1^+, t_2^+, r^+ for e_2^+ may be different from the best supporting e_1^-, t_2^-, r^- for e_2^-.

- Q is the set of queries, and q is one query.
- e_2^+ is a relevant entity, e_2^- is an irrelevant entity, for query q.
- ϕ is the feature vector representing an interpretation, composed of e_1 (one or more entities mentioned in query q), r (relation mentioned or hinted at in q), t_2 (type mentioned or hinted at in q) and e_2 (candidate answer entity). ϕ incorporates inputs from the three convnets.
- ξ is a vector of non-negative slack variables.

- C a balancing regularization parameter.
- w is the weight vector to be learnt.

The max in the LHS of constraint (1) leads to nonconvexity, which we address by introducing auxiliary variables $u(q, e_1^+, t_2^+, r^+; e_2^+)$ for each relevant candidate entity in the following optimization.

$$\begin{aligned}
&\min_{\xi \geq \vec{0}, w} \quad \tfrac{1}{2}\|w\|_2^2 + \frac{C}{|\mathcal{Q}|}\xi \cdot \vec{1} \quad \text{such that} \\
\forall q, e_2^+, e_2^-; e_1^-, t_2^-, r^- :&\ \sum_{e_1^+, t_2^+, r^+} u(q, e_1^+, t_2^+, r^+; e_2^+)\, w \cdot \phi(q, e_1^+, t_2^+, r^+; e_2^+) \\
&\geq 1 - \xi_{q, e_2^+, e_2^-} + w \cdot \phi(q, e_1^-, t_2^-, r^-; e_2^-) \qquad (2) \\
\forall q, e_1^+, t_2^+, r^+; e_2^+ :&\ u(q, e_1^+, t_2^+, r^+; e_2^+) \in \{0, 1\} \\
\forall q, e_2^+ :&\ \sum_{e_1^+, t_2^+, r^+} u(q, e_1^+, t_2^+, r^+; e_2^+) = 1 \\
\forall q, e_2^+, e_2^- :&\ \xi_{q, e_2^+, e_2^-} \geq 0
\end{aligned}$$

For tractability, we relax the 0/1 constraint over u variables to the continuous range $[0, 1]$:

$$\forall q, e_1^+, t_2^+, r^+; e_2^+ : \quad u(q, e_1^+, t_2^+, r^+; e_2^+) \in [0, 1] \qquad (3)$$

We obtain local optima for (2) by alternately updating w and u. Each of these is a convex optimization problem. Through optimization (2), AQQUCN integrates query interpretation and entity response ranking into a unified framework, rather than a two-stage compile-and-execute strategy common in other QA systems, which effectively gambles on one best structured interpretation. That being said, QRN, QTN and QCN are trained separately for their individual goals. This is actually better than end-to-end training [Roth 2017] with limited QA data with potential idiosyncrasies (e.g., largely well-formed questions, or all answers verifiable from KG alone).

4. EXPERIMENTS AND RESULTS

We use both keyword and original natural language forms of the queries from [Joshi et al. 2014], leading to four query sets[2] (Table I). By design, all WebQuestions queries can be answered using the Freebase KG. In contrast, only 57% of TREC-INEX queries can be answered from KG alone under the restriction that e_1 and e_2 lie within two hops. Thus corpus evidence is important for TREC-INEX.

Source	Name	#train	#test	Query type
TREC-INEX	TI-KW	493	211	Syntax-poor
	TI-NLQ	493	211	Syntax-rich
WebQuestions	WQ-KW	563	240	Syntax-poor
	WQ-NLQ	3778	2032	Syntax-rich

Table I. Summary of different querysets. A portion of the train set is internally set aside for dev.

[2]WQ-NLQ is exactly the same as the WebQuestions query set.

4.1 Entity ranking comparison

We present entity *ranking* comparison against [Joshi et al. 2014] in Table II. Both AQQUCN and the system of [Joshi et al. 2014] output a *ranking* over entities, and we compare using standard ranking performance measures: Mean Average Precision (MAP), Mean Reciprocal Rank (MRR) and Normalized Discounted Cumulative Gain (NDCG).

Data	System	MAP	MRR	NDCG
TI-KW	Joshi et.al. 2014	40.9	41.9	50.2
	AQQUCN	**56.5**	**59.1**	**61.7**
WQ-KW	Joshi et.al. 2014	37.7	40.1	47.4
	AQQUCN	**55.6**	**55.9**	**59.3**
TI-NLQ	Joshi et.al. 2014	35.8	36.2	42.6
	AQQUCN	**51.4**	**53.6**	**56.4**
WQ-NLQ	AQQUCN	**60.0**	**61.2**	**62.8**

Table II. Entity ranking performance comparison with Joshi et. al. 2014. WQ-NLQ scores are not reported due to scaling issue.

In Table II, we see 16–18% absolute improvement in mean average precision (MAP) over various querysets. This improvement is largely attributed to better query interpretation (by avoiding the hard query segmentation strategy and allowing overlapping roles for query tokens) and improved feature engineering through convnets.

Data	System	F1
TI-KW	Aqqu	22.5
	Berant et.al. 2015	12.7
	AQQUCN (relative threshold)	**43.3**
	AQQUCN (ideal threshold)	60.7
WQ-KW	Aqqu	35.3
	Berant et.al. 2015	36.5
	AQQUCN (relative threshold)	**44.5**
	AQQUCN (ideal threshold)	58.7
TI-NLQ	Aqqu	23.6
	Berant et.al. 2015	10.7
	AQQUCN (relative threshold)	**39.8**
	AQQUCN (ideal threshold)	55.8
WQ-NLQ	Yao et.al. 2014	33.0
	Berant et.al. 2013	35.7
	Yao et.al. 2015	44.3
	Aqqu	49.5
	Berant et.al. 2015	49.6
	Yih et.al. 2015	52.5
	Xu et.al. 2016	**53.3**
	AQQUCN (relative threshold)	50.1
	AQQUCN (ideal threshold)	62.7

Table III. F1 comparison with recent KGQA systems. See text for relative vs ideal threshold discussion.

4.2 Entity set retrieval comparison

In Table III, we present entity *set* comparison against several KBQA systems [CodaLab 2016]. Each of those KBQA systems reports a set of entities without any ordering between them. Therefore, to compare these systems, we have to convert our ranking into a set. We thus report our results in two ways. First, we extract a set from the ranking by including all entities with score within $x\%$ of the top ranked entity's score ("relative threshold"). Tuned on held-out data, x turned out to be 0.95. Second, we also report an "ideal threshold" F1 on our results, obtained as the best case or clairvoyant F1 that can be obtained from our ranking by thresholding at any position consistent with the ground truth. As is obvious, the first one provides unfair advantage to existing KBQA systems, whereas the second provides unfair advantage to our system.

Comparing with KBQA systems in Table III, we see that AQQUCN also performs better than all the related work in three out of four query sets. On the fourth query set (WQ-NLQ), our system performs better than many related works even when at a disadvantage (relative threshold). The superior performance of our system is again attributed to better feature engineering through convnets, as well as the use of corpus as a supplementary information source.

5. CONCLUSION

We presented AQQUCN, a system that unifies structured interpretation of queries with ranking of response entities. Apart from seamlessly integrating corpus and KG information, AQQUCN has two salient features: it can deal with the full spectrum of query styles between keyword queries and well-formed questions; and it directly ranks response entities, rather than 'compile' the input to a structured query and execute that on the KG alone. AQQUCN code was based on the AQQU code base [Bast and Haußmann 2015] and is available from the CSAW project [CSAW 2017] site.

Acknowledgment. Thanks to Saurabh Sarda for initial adaptation of [Joshi et al. 2014] code leading to our move to AQQU code base. Thanks to Elmar Haußmann for generous help with AQQU code.

REFERENCES

BAST, H. AND HAUSSMANN, E. 2015. More accurate question answering on freebase. In *CIKM*. 1431–1440.

BERANT, J., CHOU, A., FROSTIG, R., AND LIANG, P. 2013. Semantic parsing on Freebase from question-answer pairs. In *EMNLP Conference*. 1533–1544.

CODALAB. 2016. Webquestions benchmark for question answering. `http://bit.ly/2kvXroJ`.

CORNOLTI, M., FERRAGINA, P., CIARAMITA, M., RUED, S., AND SCHUETZE, H. 2014. The SMAPH system for query entity recognition and disambiguation. In *ERD Challenge Workshop*.

CSAW. 2017. The csaw project at iit bombay. `http://www.cse.iitb.ac.in/~soumen/doc/CSAW/`.

JOSHI, M., SAWANT, U., AND CHAKRABARTI, S. 2014. Knowledge graph and corpus driven segmentation and answer inference for telegraphic entity-seeking queries. In *EMNLP Conference*. 1104–1114. Download `http://bit.ly/1OCKbVW`.

LIN, T., PANTEL, P., GAMON, M., KANNAN, A., AND FUXMAN, A. 2012. Active objects: Actions for entity-centric search. In *WWW Conference*. ACM, 589–598.

ROTH, D. 2017. On the necessity of learning and reasoning: A perspective from natural language understanding. McCarthy award acceptance speech at IJCAI 2017: `https://www.youtube.com/watch?v=tAKn3Gt75rg`.

SEVERYN, A. AND MOSCHITTI, A. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 373–382.

WEST, R., GABRILOVICH, E., MURPHY, K., SUN, S., GUPTA, R., AND LIN, D. 2014. Knowledge base completion via search-based question answering. In *WWW Conference*. 515–526.

YIH, S. W.-T., CHANG, M.-W., HE, X., AND GAO, J. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *ACL Conference*. 1321–1331.

BIOGRAPHIES

Uma Sawant is a PhD candidate in computer science and engineering at IIT Bombay. Her interests include question answering, search and recommendation systems. She has previously worked at Yahoo and is currently working at LinkedIn.

Soumen Chakrabarti is professor of computer science and engineering at IIT Bombay. His interests include robust query interpretation, continuous knowledge graph representation, corpus annotation with types, entities and relations, and the integration of these capabilities into question answering systems. During 2014–2016 he worked in the NLP group at Google. He is author of a popular book titled *Mining the Web*.

Ganesh Ramakrishnan is an associate professor of computer science and engineering at IIT Bombay. His interests include incorporation of domain knowledge and human interaction in machine learning models and multi-instance multi-label learning. He worked at IBM India research labs between 2004 and 2009.