

# More Accurate Entity Ranking Using Knowledge Graph and Web Corpus

Uma Sawant, Soumen Chakrabarti and Ganesh Ramakrishnan  
IIT Bombay

**Draft: Do not distribute**



**Abstract**—Recent years have witnessed some convergence in the architecture of entity search systems driven by a knowledge graph (KG) and a corpus with annotated entity mentions. However, each specific system has some limitations. We present AQQUCN, an entity search system that combines the best design principles into a public reference implementation. AQQUCN does not depend on well-formed question syntax, but works equally well with syntax-poor keyword queries. It uses several convolutional networks (convnets) to extract subtle, overlapping roles of query words. Instead of ranking structured query interpretations, which are then executed on the KG to return unranked sets, AQQUCN directly ranks response entities, by closely integrating coarse-grained predicates from the KG with fine-grained scoring from the corpus, into a single ranking model. Over and above competitive F1 score, AQQUCN gets the best entity ranking accuracy on two syntax-rich and two syntax-poor public query workloads amounting to over 8,000 queries, with 16–18% absolute improvement in mean average precision (MAP), compared to recent systems.

## 1 INTRODUCTION AND RELATED WORK

A significant fraction of queries made to commercial Web search engines are entity-seeking [Lin et al., 2012]. Search engines enhance the user experience with direct entity responses in response to queries that fit certain structured templates. Four common templates are shown below (some subsume others).

$$\begin{array}{ccc}
 e_1 \xrightarrow{r_1} e_2 & e_1 \xrightarrow{r_1} e_2 \in t_2 & e_1' \xrightarrow{r_1'} e_2' \\
 e_1 \xrightarrow{r_1} m \xrightarrow{r_2} e_2 & & e_1 \xrightarrow{r_1} m \xrightarrow{r_2} e_2
 \end{array}$$

Here  $e_1, e_1'$  are grounded entities mentioned in the query,  $m$  is a mediator (CVT) node,  $r_1, r_1', r_2$  are relations,  $t_2$  is the type of  $e_2$ , and  $e_2$  must be bound to desired response entity or entities.

**Convergent architecture:** How best to integrate signals from a structured knowledge graph (KG) and an unstructured corpus to this end has been under investigation for several years, leading to some convergence in system architecture. An entity linker marks  $e_1, e_1'$  in the query. These are located as nodes in the KG. An exploration around  $e_1, e_1'$  in the KG with bounded hops (1–2) is performed to identify candidate  $e_2$ s. Features are computed between the query and KG neighborhood around  $\{e_1, e_2\}$  or  $\{e_1, e_1', e_2\}$  as the case may be. The system learns to rank either the  $e_2$ s directly, or a structured query fitting one of the above templates. In the former case, the response is a *ranked list* of  $e_2$ s. In the latter (two-stage) case, the structured query is executed against the KG to obtain a *set* of  $e_2$ s. The vast majority of recent systems [Yao and Van Durme, 2014, Dong et al., 2015, Bast and

Haussmann, 2015, Yih et al., 2015, Yao, 2015, Xu et al., 2016b] resemble the above sketch, and generally give superior accuracy compared to semantic query parsing [Berant et al., 2013, Berant and Liang, 2015, Artzi et al., 2015]. We now highlight important characteristics of these systems.

**Two-stage systems:** AQQU [Bast and Haussmann, 2015] is a clean public implementation of the standard two-stage architecture sketched above. Apart from pretrained word2vec embeddings to measure various text similarities, it does not use any neural networks. STAGG [Yih et al., 2015] clearly specifies stages in which the structured query is built. It applies one convolutional network (convnet) to the question to predict the relation  $r_1, r_2$ . STAGG has among the best accuracy on the widely-used WebQuestions benchmark [Berant et al., 2013].

**Systems that directly rank  $e_2$ :** Sawant and Chakrabarti [2013], Joshi et al. [2014] exhaustively segment the query to find exclusive spans that hint at  $e_1, r, t_2$ . Such hard segmentation is very slow for long queries (e.g., well-formed natural language questions), and leads to missed relation and type signals. To match the segments to  $e_1, r, t_2$ , they use traditional language models over sparse lexical features. They do not harness the power of embedded representations. To our knowledge they were never tested for natural language questions. In our experiments, they were unacceptably slow and perform poorly on such queries. Another system which ranks  $e_2$  directly is Dong et al. [2015]. Their main idea is to apply three convnets on the query, matching their outputs to three families of features around  $e_2$  in the KG. They do not use a corpus at query time. Despite the elegant architecture, their performance is not at par with STAGG or AQQU, for reasons not entirely understood.

**Query-time use of corpus:** Several systems [Tymoshenko et al., 2016, Choi et al., 2017] employ a corpus for offline training of various modules and/or for answering the query independent of any KG [Krishnamurthy et al., 2016]. Relatively few use a corpus directly during query time, in conjunction with a KG. After running an inferred structured query on the KG, Xu et al. [2016b] use the corpus for filtering, improving upon STAGG’s F1 score on WebQuestions. Savenkov and Agichtein [2016] present an integrated architecture for blending corpus and KG signals. They gain over AQQU, but not over STAGG. This may not be too surprising, since WebQuestions was designed to be entirely answerable using the KG alone. This motivates the use of benchmarks where corpus is critical.

**Our contributions:** We present AQQUCN, an entity search system that combines the best design principles into a public<sup>1</sup> reference implementation. AQQUCN does not depend on formal parsing [Berant et al., 2013, Artzi et al., 2015, Krishnamurthy et al., 2016] or segmentation [Sawant and Chakrabarti, 2013, Joshi et al., 2014], but works equally well with syntax-rich (In which band did Jimmy Page perform before Led Zeppelin?) and syntax-poor (jimmy page band before led zeppelin) queries. Going beyond [Yih et al., 2015] and inspired by Dong et al. [2015], AQQUCN uses multiple well-differentiated convnets to extract diffuse, overlapping functional segments from the query. Beyond [Dong et al., 2015], AQQUCN convnets are trained using extensive offline information. Like Joshi et al. [2014] and unlike Xu et al. [2016b] and others, AQQUCN directly integrates corpus signals into a final entity ranking stage, rather than executing the best structured query to return an entity set. AQQUCN and other systems are evaluated on two syntax-rich and two syntax-poor public query workloads amounting to over 8,000 queries. Over and above competitive F1 score, AQQUCN gets state-of-the-art entity ranking accuracy, with 16–18% absolute improvement in mean average precision (MAP).

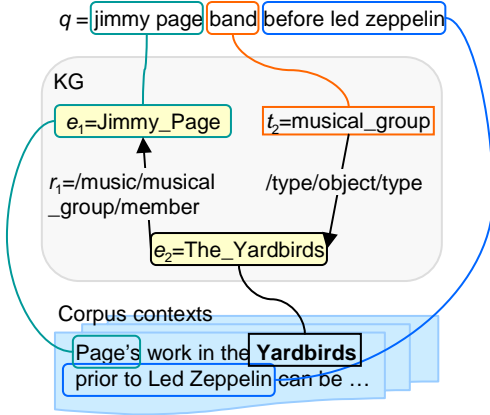


Fig. 1: Entity search example using corpus and knowledge graph.

## 2 PROPOSED ARCHITECTURE

Consider the query [jimmy page band before led zeppelin], with expected answer entity *The\_Yardbirds* (Figure 1). Accurate identification of the semantic elements mentioned in the query, viz., query entity (*Jimmy\_Page*), the type of the answer entity (*musical\_group*), and the relation (*/music/musical\_group/member*) connecting the query entity to the desired answer, allows translation of the input query into structured templates mentioned in Section 1 and returns possible answer entities from KG. Additionally, corpus snippets that match query words and mention entities provide supporting evidence in favor of another (likely overlapping) candidate set of entities. Our system evaluates these (possibly noisy and ambiguous) signals and outputs an entity ranking. Our system is divided into three main stages (Figure 2) as follows:

**Candidate generation and evidence collection:** Section 2.1 describes how potential query entities ( $e_1$ ), candidate answer entities ( $e_2$ ) and connecting relations ( $r$ ) are identified for query  $q$ , leading to a set of pairs of interpretation and candidate answer entity, denoted  $(I, e_2)$ .

**Feature generation:** Section 2.2 describes how we assemble a feature vector  $\phi(I, e_2)$  for scoring a  $(I, e_2)$  combination. The key features are contributed by three CNNs.

**Candidate entity scoring and ranking:** Section 2.3 describes how we learn a scoring model of the form  $w \cdot \phi(\dots)$  based on training data where only gold  $e_2$  is specified but interpretation  $I$  is latent.

### 2.1 Candidate generation and evidence collection

Given a query  $q$ , we first identify the set of in-query entities  $\mathcal{E}_1$  using an entity tagger TagMe [Ferragina and Scaiella, 2010]. In parallel, we gather corpus evidence for  $q$  in the form of text snippets containing mention of some entity  $e$  in the KG and the query words within a window of  $w$  words around that mention. Each such snippet is now an evidence snippet for the entity  $e$  and query  $q$ . Many prior systems e.g. [Bast and Haussmann, 2015, Yih et al., 2015, Xu et al., 2016b] limit the set of candidate answer entities  $\mathcal{E}_2$  to those occurring in the 2-hop KG neighborhood<sup>2</sup> of any entity in  $\mathcal{E}_1$ . We go a step further by also adding to  $\mathcal{E}_2$  the entities that occur in any corpus snippet. This allows the system to recover from early errors, such as when  $\mathcal{E}_1$  identified by the entity tagger is empty or wrong, or when the query entity and answer entity are more than two hops apart in KG.

No.	Description
1	Sum of QCN match scores over all snippets for $(q, e_2)$
2	Sum of $(q, e_2)$ QCN match scores for all $e_2$ connected to $(e_1, r)$ in KG
3-5	Sum, average and count of $(q, e_1)$ match score by entity tagger (TagMe)
6	Count of $e_1$ with exact match in $q$
7	$(q, r)$ match using QRN
8	No. of simple relations in $r$
9-12	Relation token match features 10-13 from [Bast and Haussmann, 2015]
13	$(q, t_2)$ match using QTN
14	$(q, t_r)$ match using QTN, where $t_r$ is the expected end type of $r$ in freebase
15	KG-only structured query template score from [Bast and Haussmann, 2015]
16	Fraction of snippets in which $e_2$ occurs over all queries.

TABLE 3: Features used by our ranker.

Each query interpretation  $I \in \mathcal{I}$  corresponds to one of the templates mentioned in Section 1, in conjunction with the query words. Inclusion of query words signifies the role of the corpus in an interpretation, viz., the fact that the evidence connected to this interpretation will include corpus snippets containing  $e_2$  and the query words.

### 2.2 Feature generation

In this stage, we generate a feature vector  $\phi$  to describe the match between query  $q$ , each candidate query interpretation  $I \in \mathcal{I}$  and each candidate answer entity  $e_2 \in \mathcal{E}_2$ . We generate match features through three role-differentiated convolutional networks (described in detail next), viz., QCN (query-corpus network), QRN (query-relation network) and QTN (query-type network). We also include hand-crafted features such as aggregated entity tagger

1. <http://bit.ly/2hwAeAl>

2. Two hops needed to cross mediator nodes (CVTs).

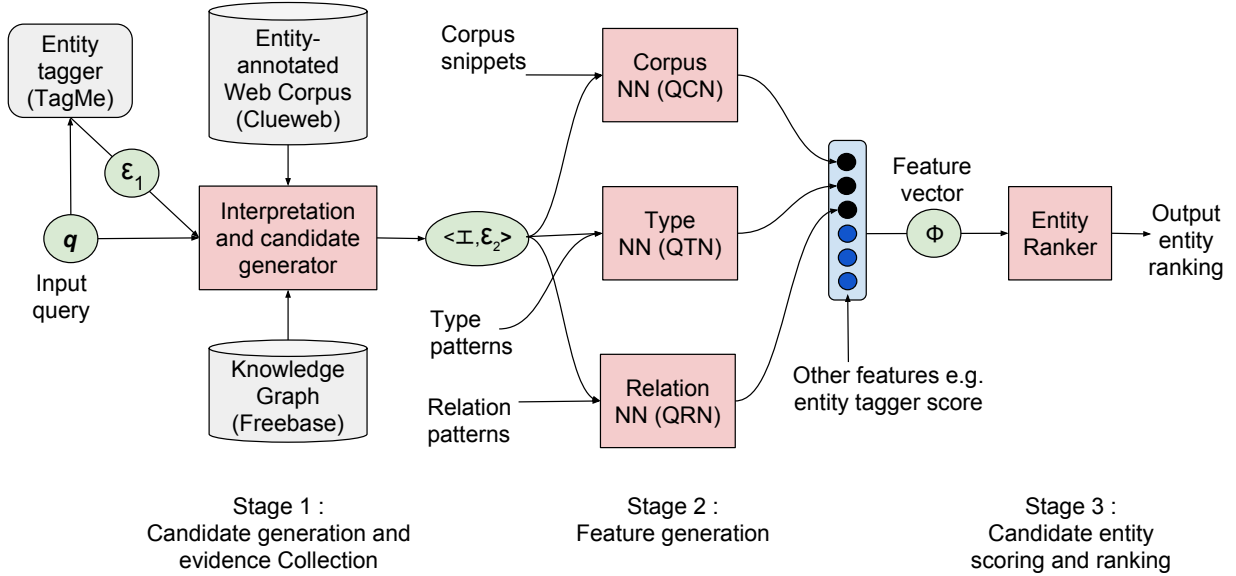


Fig. 2: Explicit query interpretation and feature generation using multiple CNNs. Variables in green circles correspond to the variables in Figure 9. Query to CNN connections are not shown to avoid clutter.

scores, count of exact-matching  $e_1$  etc. Table 3 shows the complete list of features.

### 2.2.1 Query-Type Network (QTN)

The query-type network (QTN) outputs a compatibility score between the query  $q$  and a candidate type  $t_2$ . This is a multi-class, multi-label classification problem.

Good quality training data is essential to ensure that the network is robust against diverse query syntax. The naive strategy of considering all  $(q, t)$  pairs as positive training data, where  $t$  is any type connected to the gold answer entity  $e_2$  for training query  $q$ , results in large amount of noise. For example, `/broadcast/radio_station_owner` is not the correct answer type for the query `[maya moore college]`, even if the answer entity `University_of_Connecticut` belongs to that type. We used human supervision to remove approx. 30% irrelevant  $(q, t)$  pairs and improve training data quality.

We additionally represented the type through additional zero or more lexical patterns (Table 4) obtained as follows :

**Freebase relation names:** Consider a (subject, relation, object) fact triple e.g. (Captain\_America:The\_First\_Avenger, /film/film/prequel, Thor). Relations in freebase have composite names in the form “ $x/y/z$ ” where  $y$  and  $z$  indicate the types for subject and object. E.g. `prequel` is a type indicator word for `Thor`. Meanwhile, Freebase declares expected type for endpoint entities of each  $r$ , e.g. `/film/film` for `/film/film/prequel`. We combine these two information nuggets and add `prequel` as a lexical pattern expressing the type `/film/film`.

**Freebase type names:** Ending substrings of the type name are also considered as patterns (e.g. ‘`treatment`’ for the type `medical_treatment`).

Figure 7 illustrates the multi-class multi-label design of QTN, partly inspired by Kim [2014] and Severyn and Moschitti [2015]. For each query  $q$  input to the network, the network provides an output score vector of size equal to the number of types, as follows:

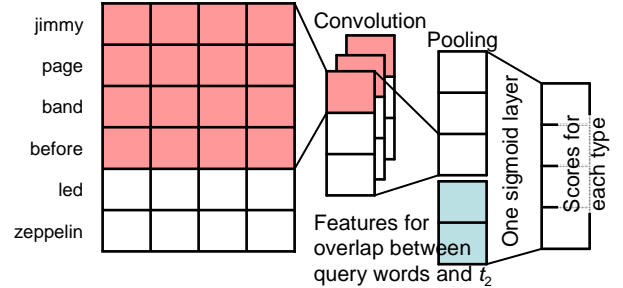


Fig. 7: Query-Type Network (QTN) architecture and inputs. Query-Relation Network (QRN) architecture is identical except that types are replaced by relations in the training data.

- 1) In the initial layer, each query word is represented as a vector embedding learnt during training. Then convolution and pooling layers are used to extract a fixed-length feature vector from the variable length input.
- 2) In a separate layer, we compute word overlap features inspired by Severyn and Moschitti [2015]. Specifically, we compute Jaccard similarity between the query and each type name, by representing each as a bag of words. We also compute Jaccard similarity between the query and each type pattern, and then take maximum score over all patterns of a given type. This process results in 2 features for each (query, type) pair.
- 3) Similar to the bag of words based overlap, we compute Jaccard similarity between the word stem (lemmatized) form for each query, type name and type pattern; resulting in additional 2 features for each (query, type) pair.
- 4) A fully connected hidden layer with sigmoid activation function is used at the last stage, to score all types using the above features.

### 2.2.2 Query-Relation Network (QRN)

The Query-Relation Network (QRN) outputs a compatibility score between a candidate relation  $r$  and the query  $q$ . As with QTN, we generate multi-class, multi-label training data in the form of  $(q, r)$

Type	Type (lexical) patterns
/book/author	dramatist, author, journalist, poet, novelist, writer, editor
/people/deceased_person	dead, deceased, late, expired, deceased person, victim, person

TABLE 4: Example path (lexical) patterns for types. Due to automatic extraction, some patterns may be idiosyncratic (e.g. ‘victim’ for /people/deceased\_person).

Relation	Relation (lexical) patterns
/film/writer/film	film, film by, by, of, written by, wrote, author of
/theater/play/composer	by, written by, music by, wrote, with music of, of

TABLE 5: Example path (lexical) patterns for relations. Notice how the same patterns (‘written by’, ‘of’) can imply different relations, causing ambiguity in query interpretation.

Query	Snippets from Entity-annotated Web corpus
spanish poet died civil war	[Positive] “ <u>Lorca</u> was executed in 1936, during the spanish civil war.” [Negative] “The murder of the spanish poet by <u>nationalists</u> in the civil war remains one of Spain’s open wounds.”
Who was the first U.S. president ever to resign?	[Positive] “ <u>Nixon</u> become the first president in American history to resign.” [Negative] “ <u>Gerald R. Ford</u> took the oath of office after the first-ever resignation by a U.S. President.”

TABLE 6: Example positive and negative corpus snippets for queries. Note how ‘Lorca’ and ‘Nixon’ are mentions of the gold answer entities `Federico_Garcia_Lorca` and `Richard_Nixon` respectively, while ‘nationalist’ and ‘Gerald R. Ford’ are mentions of non-answer entities `Francoist_Spain` and `Gerald_Ford` respectively.

pairs, where  $r$  is a relation connecting to the gold answer entity  $e_2$  to the entity  $e_1$  mentioned in the query  $q$ . There could be multiple  $r$ , for the same  $(q, e_1, e_2)$  tuple. Such training data generation process is common [Dong et al., 2015, Yih et al., 2015, Bordes et al., 2014] and human curation is not used to remove noise.

Similar to QTN, we enrich our training data using relation description lexical patterns as follows: we start with (subject, relation, object) facts in the KG and locate sentences in the annotated corpus where both subject and object are mentioned. We identify the path connecting the two in the dependency parse of the sentence, expressed as a sequence of lemmatized words. Then we retain only the most frequent paths. This gives a bag of path patterns that describe relation  $r$  (examples in Table 5). The network architecture for QRN is same as in Figure 7. The only difference is that for QRN we have (query, relation) and (relation, patterns) data as training inputs.

### 2.2.3 Query-Corpus Network (QCN)

We use the query corpus network (QCN) for scoring zero or more evidence snippets from the Web corpus, centered around various candidate answer entities  $e_2 \in \mathcal{E}_2$  for each query  $q$ . Given a query  $q$  and a text snippet from the Web corpus, QCN should assign high score to the pair if the text snippet contains evidence to correctly answer  $q$ . We generate positive and negative snippets for each training query using the following indirect supervision strategy. We treat all text snippets centered around gold answer entity and containing some or all query words as positively labeled snippets. Similarly, we treat all text snippets centered around any non-answer entity and containing some or all query words as negatively labeled snippets (examples in Table 6). To train this network, we use the state-of-the-art short text ranking system proposed by Severn and Moschitti [2015].

Once trained, QCN outputs a score for each snippet belonging to a candidate answer entity  $e_2 \in \mathcal{E}_2$ . Similar to Balog et al. [2009], Macdonald and Ounis [2006] and Joshi et al. [2014], we add up the snippet scores over all snippets belonging to an entity  $e_2$  for a query  $q$ , and use it as a feature for  $(q, e_2)$  (feature no. 1 in Table 3).

## 2.3 Candidate entity scoring and ranking

The final stage scores the combination of interpretation  $I$  and candidate entity  $e_2$ . It uses a linear discriminative model learnt from pairs of correct and incorrect answer entities for each query  $(q, e_2^+, e_2^-)$ . As we do not know the gold query interpretation as part of the training data, we treat that as a latent variable. Following the simplifications explained in [Joshi et al., 2014], we pose this problem as a bilinear discriminative optimization problem shown in Figure 8 and solve it using alternating optimization [Dai and Yuan, 2003]. Figure 9 shows the pseudocode for inference.

## 3 EXPERIMENTS

### 3.1 Testbed

**KG and Corpus:** We use Freebase as the KG and ClueWeb09B/FACC1 [Gabrilovich et al., 2013] as the entity-

$$\begin{aligned}
& \min_{\xi \geq 0, w} \frac{\|w\|^2}{2} + \frac{C}{|Q|} |\xi| \\
& \forall q, e_2^+, e_2^-, e_1', t_2', r' : \\
& \quad \sum_{e_1, t_2, r} u(q, e_1, t_2, r, e_2^+) w \cdot \phi(q, e_1, t_2, r, e_2^+) \\
& \quad \geq 1 - \xi_{q, e_2^+, e_2^-} + w \cdot \phi(q, e_1', t_2', r', e_2^-) \\
& \quad u(q, e_1, t_2, r, e_2^+) \in \{0, 1\} \\
& \quad \forall q, e_2^+ : \sum_{e_1, t_2, r} u(q, e_1, t_2, r, e_2^+) = 1.
\end{aligned}$$

Fig. 8: Discriminative training optimization problem.  $w$  is the weight vector to be learnt;  $u$  is an auxiliary convex combination vector used to simplify the non-convex constraint in the original problem [Joshi et al., 2014].  $\phi$  is the feature vector representing interpretation  $I$ , composed of  $e_1$  (one or more entities in  $q$ ),  $r$  (relation in  $q$ ),  $t_2$  (type in  $q$ ) and  $e_2$  (candidate answer entity).  $e_2^+$  is a positive candidate and  $e_2^-$  a negative candidate for  $q$ .  $Q$  is the number of queries,  $C$  a balancing parameter, and  $\xi$  non-negative slack variable vector.

```

1: input: query token sequence  $q$ 
2: Generate  $\mathcal{E}_1$  (query entity set) using entity tagger
3: Generate  $\mathcal{I} = \{I_i = (e_{1i}, t_{2i}, r_i, q); e_{1i} \in \mathcal{E}_1\}$  as potential
   interpretations, indexed by  $i$ 
4: Generate  $\mathcal{E}_2$  = candidate answer entity set reachable from
   any  $I_i \in \mathcal{I}$  in KG or corpus
5: for all  $e_{2j} \in \mathcal{E}_2$  do
6:    $bestScore_j \leftarrow -\infty$ 
7:   for all interpretation  $I_i$  do
8:     Generate CNN scores for  $(q, I_i, e_{2j})$  using QTN,
       QCN, QRN (see text)
9:     Generate other features in Table 3
10:    Create  $\phi_{ij}$ , the feature vector for  $(I_i, e_{2j})$  using the
       above features
11:    Score  $\phi_{ij}$  using a trained linear model to get  $s_{ij}$ , the
       score of  $(I_i, e_{2j})$ 
12:     $bestScore_j \leftarrow \max_i \{bestScore_j, s_{ij}\}$ 
13: output: ranking of  $e_{2j} \in \mathcal{E}_2$  according to decreasing
        $bestScore_j$ 

```

Fig. 9: High-level pseudocode for inference in our proposed system. Also see Figure 2.

annotated corpus.

**Query sets:** We use both keyword and original natural language forms of the queries from Joshi et al. [2014]<sup>3</sup>, leading to four query sets<sup>4</sup> (Table 10). By design, all WebQuestions queries can be answered using Freebase. In contrast, only 57% of TREC-INEX queries can be answered by KG queries under the restriction that  $e_1$  and  $e_2$  lie within two hops, thus rendering corpus evidence important.

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 10: Summary of different querysets. As in [Bast and Haussmann, 2015], a portion of the train set is internally set aside for dev.

**Deep network training:** Data used to train our networks is available at <http://bit.ly/2hwAeAl>. Some important design choices are specified below.

**QTN and QRN (Figure 7):** Initial word vectors are learnt using the CNN-non-static version of [Kim, 2014]. Filter sizes are set to 3,4 with 150 feature maps each. Drop-out rate is 0.5, with 100 epochs and early stopping using a validation split of 10%. Training is done through stochastic gradient descent over shuffled mini-batches with Adadelata update.

**QCN:** The width of the convolution filters is set to 5, the number of convolutional feature maps to 150 and batch size to 50. L2 regularization term is  $10^{-5}$  for the parameters of convolutional layers and  $10^{-4}$  for all the others. The dropout rate is set to 0.5. We initialize the word vectors using the word embeddings trained by [Huang et al., 2012].

3. Available at <http://bit.ly/1OCKbVW>

4. WQ-NLQ is exactly the same as WebQuestions queryset

**End-to-end training:** We use the pruning process of Bast and Haussmann [2015] to restrict the set of KG queries (and hence  $\mathcal{E}_2$ ) to a practical size. The corpus snippet window size  $w$  is set to 20. We normalize all feature values in  $[0, 1]$ . AQQUCN builds upon AQQU and is written in Python.

### 3.2 Benefits of corpus evidence and effectiveness of QCN

In this set of experiments, we systematically evaluate the benefit of using the Web corpus as an additional resource along with the KG, across multiple querysets of diverse nature.

The corpus is used both to potentially increase coverage of candidate  $e_2$ s, and to find snippets supporting them. We first report AQQUCN’s performance with (“No corpus”) as against (“With corpus”). As evident from Table 11, corpus support is beneficial, if not critical, for all query sets.

Data	System	map	mrr	n@10
TI	No corpus	32.9	33.1	35.8
KW	With corpus	<b>56.5</b>	<b>59.1</b>	<b>61.7</b>
WQ	No corpus	54.4	54.4	58.5
KW	With corpus	<b>55.6</b>	<b>55.8</b>	<b>59.3</b>
TI	No corpus	28.9	29.2	31.9
NLQ	With corpus	<b>51.4</b>	<b>53.6</b>	<b>56.4</b>
WQ	No corpus	58.5	59.1	61.2
NLQ	With corpus	<b>60.0</b>	<b>61.2</b>	<b>62.8</b>

TABLE 11: Evidence from entity-annotated corpus significantly improves entity ranking.

Data	System	map	mrr	n@10
TI-KW	BM25	46.8	48.9	52.6
	QCN	<b>52.7</b>	<b>55.3</b>	<b>58.1</b>
WQ-KW	BM25	54.0	54.4	58.2
	QCN	<b>54.2</b>	<b>54.4</b>	<b>58.3</b>
TI-NLQ	BM25	46.1	48.3	52.2
	QCN	<b>50.5</b>	<b>52.9</b>	<b>54.8</b>
WQ-NLQ	BM25	59.7	60.9	62.4
	QCN	<b>59.9</b>	<b>61.0</b>	<b>62.5</b>

TABLE 12: QCN is much more effective than BM25.

We then compared the effectiveness of QCN over more traditional query-document matching algorithm such as BM25, keeping other features unchanged (results in Table 12). QCN showed a relative gain of 9.5% and 12.5% in MAP over BM25 for TI- queries. For WQ-\* queries, the gain was mild (around 1%) but positive.

### 3.3 Effectiveness of type and relation networks

To evaluate the effectiveness of QTN and QRN convolutional networks, we compared them with Language Models [Zhai, 2008].

In the first (“No CNN”) setting, we used three features: query-corpus match using BM25, query-type match using a type language model and query-relation match using a relation language model. In the next setting (“QTN”), we replaced the type language model scores with QTN scores and repeated the experiment. Similarly, in the last setting, we replaced the relation language model scores with QRN scores, and repeated the experiment. Results in Table 13 reassure that the deep network’s performance is much superior to the traditional hand tuned models.

Data	System	map	mrr	n@10
TI KW	No CNN	44.3	46.0	49.6
	QTN	<b>47.0</b>	<b>49.2</b>	<b>52.9</b>
	QRN	<b>47.5</b>	<b>49.3</b>	<b>53.9</b>
WQ KW	No CNN	51.0	51.6	55.5
	QTN	<b>54.1</b>	<b>54.8</b>	<b>57.8</b>
	QRN	<b>51.9</b>	<b>52.6</b>	<b>55.7</b>
TI NLQ	No CNN	45.3	48.0	49.6
	QTN	<b>46.3</b>	<b>49.0</b>	<b>51.0</b>
	QRN	<b>47.4</b>	<b>49.9</b>	<b>51.9</b>
WQ NLQ	No CNN	47.6	49.4	52.8
	QTN	<b>52.9</b>	<b>54.5</b>	<b>59.0</b>
	QRN	<b>51.9</b>	<b>52.9</b>	<b>55.8</b>

TABLE 13: QTN and QRN effectiveness.

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

TABLE 14: Entity ranking performance comparison with Joshi et al. [2014]. WQ-NLQ scores are not reported for their system due to scaling issue (Refer text).

### 3.4 End-to-end comparison with recent systems

Next, we present the comparison of our end-to-end system with existing systems, in two parts: (i) entity ranking comparisons against [Joshi et al., 2014] in Table 14 and (ii) F1 score comparison against several KBQA systems listed in <http://bit.ly/2kvXroJ><sup>5</sup> in Table 15.

Since both our system and [Joshi et al., 2014] output a *ranking* over entities, the former is a more direct comparison. In contrast, all the KBQA systems in <http://bit.ly/2kvXroJ> report a *set* of entities without any ordering between them. There is no correct way to compare a *set* to a *ranking* without being unfair to one of the systems. 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. 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 [Yazdani et al., 2015] 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.

For systems with proprietary or unavailable components, we limited the comparison to WQ-NLQ and obtained corresponding results from <http://bit.ly/2kvXroJ><sup>6</sup>. We replaced the now-deprecated Freebase entity tagger API with TagMe in FixedOrder, and hence in Aquu.

Our superior performance as compared to many KBQA systems is mainly because: (a) We get the benefit of corpus evidence<sup>7</sup>. (b) We trained our convNets on both keyword and natural language queries, thus making them more robust than systems which are trained only on natural language queries.

5. FixedOrder version from Berant and Liang [2015].

6. Savenkov and Agichtein [2016] is absent from this list as the system and its predictions were not released at the time of this manuscript.

7. Also see Table 11.

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

TABLE 15: F1 comparison with recent KGQA systems. See text for set vs ranking discussion.

The modest gap between AQQUCN and Yih et al. [2015] or Xu et al. [2016b] is more a reflection on our thresholding strategy than ranking. AQQUCN achieves much higher F1 with an ideal threshold.

Our wins over Joshi et al. [2014] result from (a) the use of convNets instead of hand created models and (b) the fact that they designed their system for syntax-poor queries, choosing to not invest in any natural language parsing.

## 4 RELATED WORK

There is a large literature on text-based QA, but here we focus on entity search with critical involvement of a large KG. Berant et al. [2013] used CCG parsing for semantic query interpretation, and introduced the WebQuestions data set. They followed up with enhancements, e.g., question paraphrases for robust learning [Berant and Liang, 2014] and improved candidate search techniques [Berant and Liang, 2015].

Apart from parsers, related work has also explored word embeddings and neural networks for better query interpretation. Bordes et al. [2014] modeled the query as a bag of words and simply added up their word embeddings. Yang et al. [2014] used embeddings to translate questions into relational predicates. More refined sentence/query embeddings have been created [Severyn and Moschitti, 2015, Iyyer et al., 2014] via recurrent and convolutional networks (CNNs), but usually applied to syntax-rich, well-formed questions. Dong et al. [2015] obtained better accuracy than Bordes et al. [2014] by replacing the aggregated word vector query representation with multiple parallel CNNs for extracting deep representations for  $r$ ,  $t_2$  and the KG neighborhood of  $e_1$ . Yih et al. [2015] used a proprietary short-text linker to extract  $e_1$ , and a CNN in the style of Severyn and Moschitti [2015] to predict  $r$ . Bast and Haussmann [2015] fit the input query to one of three possible “query templates” (each having a direct translation to a SPARQL query) through traditional hand-crafted features. Xu et al. [2016b] ranked knowledge graph interpretations using a CNN, and further refined the ranking using Wikipedia text as evidence.

All the above approaches answer the input query by finding the best matching structured query to be executed on the KG, whose



response is a *set*. In contrast, we are interested in direct entity *ranking*, using any number and variety of KG (and corpus) queries if need be. In this aspect, the closest work is that of Joshi et al. [2014], who explore segmentations of the input keyword query into spans with (latent) labels for entity, types or relations. Like us, they use an entity-annotated corpus to get evidence in support of candidate entities. However, like [Bast and Haussmann, 2015], they rely on hand-crafted features rather than neural networks. The system of Joshi et al. [2014] also performs worse than ours on syntax-rich queries.

Joshi et al. [2014], Xu et al. [2016b], Savenkov and Agichtein [2016] and Xu et al. [2016a] are among the few to harness corpus evidence. Xu et al. [2016b] do so indirectly, as a post-processing filter on the response entity set, to improve precision. Xu et al. [2016a] use OpenIE [Etzioni et al., 2011] to process the corpus into textual triples, which enhance the selection of the best structured query. Text2KB [Savenkov and Agichtein, 2016] adds several informative features to an Aquu-like architecture [Bast and Haussmann, 2015], but has comparable accuracy to STAGG [Yih et al., 2015], which our system surpasses.

## 5 CONCLUSION

We presented AQQUCN, an entity search system that works well with both syntax-poor keyword queries and syntax-rich well-formed questions. It uses of multiple convnets over a KG and a corpus, to generate features for a discriminative ranker that aggregates over latent structured interpretations and scores response entities directly. AQQUCN has competitive F1 scores for entity set responses, and state-of-the-art MAP and MRR for entity rankings. We demonstrate this using four query workloads amounting to over 8,000 queries. Our data and code will be placed in the public domain.

## REFERENCES

Y. Artzi, K. Lee, and L. Zettlemoyer. Broad-coverage CCG semantic parsing with AMR. In *EMNLP Conference*, pages 1699–1710, 2015.

K. Balog, L. Azzopardi, and M. de Rijke. A language modeling framework for expert finding. *Inf. Process. Manage.*, 45(1): 1–19, Jan. 2009. ISSN 0306-4573.

H. Bast and E. Haussmann. More accurate question answering on freebase. In *CIKM*, pages 1431–1440, 2015. URL <http://dl.acm.org/citation.cfm?id=2806472>.

J. Berant and P. Liang. Semantic parsing via paraphrasing. In *Proceedings of ACL*, volume 7, page 92, 2014.

J. Berant and P. Liang. Imitation learning of agenda-based semantic parsers. *TACL*, 3, 2015.

J. Berant, A. Chou, R. Frostig, and P. Liang. Semantic parsing on freebase from question-answer pairs. In *EMNLP*, pages 1533–1544, 2013.

A. Bordes, S. Chopra, and J. Weston. Question answering with subgraph embeddings. *arXiv preprint arXiv:1406.3676*, 2014.

E. Choi, D. Hewlett, A. Lacoste, I. Polosukhin, J. Uszkoreit, and J. Berant. Coarse-to-fine question answering for long documents. In *ACL*, 2017.

Y.-H. Dai and Y.-X. Yuan. Alternate minimization gradient method. *IMA Journal of Numerical Analysis*, 23(3):377–393, 2003.

L. Dong, F. Wei, M. Zhou, and K. Xu. Question answering over freebase with multi-column convolutional neural networks.

In *ACL/IJCNLP*, 2015. URL <http://www.aclweb.org/anthology/P15-1026>.

O. Etzioni, A. Fader, J. Christensen, S. Soderland, and M. Mausam. Open information extraction: The second generation. In *IJCAI*, volume 11, pages 3–10, 2011.

P. Ferragina and U. Scaiella. TagMe: On-the-fly annotation of short text fragments. In *CIKM*, pages 1625–1628. ACM, 2010. URL <http://doi.acm.org/10.1145/1871437.1871689>.

E. Gabrilovich, M. Ringgaard, and A. Subramanya. Facc1: Freebase annotation of clueweb corpora, version 1 (release date 2013-06-26, format version 1, correction level 0, june 2013).

E. H. Huang, R. Socher, C. D. Manning, and A. Y. Ng. Improving Word Representations via Global Context and Multiple Word Prototypes. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2012.

M. Iyyer, J. Boyd-Graber, L. Claudino, R. Socher, and H. Daumé III. A neural network for factoid question answering over paragraphs. In *EMNLP Conference*, pages 633–644, 2014.

M. Joshi, U. Sawant, and S. Chakrabarti. Knowledge graph and corpus driven segmentation and answer inference for telegraphic entity-seeking queries. In *EMNLP Conference*, pages 1104–1114, October 2014. URL <http://www.aclweb.org/anthology/D14-1117>.

Y. Kim. Convolutional neural networks for sentence classification. In *EMNLP*, pages 1746–1751, 2014. URL <http://aclweb.org/anthology/D/D14/D14-1181.pdf>.

J. Krishnamurthy, O. Tafjord, and A. Kembhavi. Semantic parsing to probabilistic programs for situated question answering. In *EMNLP*, 2016.

T. Lin, P. Pantel, M. Gamon, A. Kannan, and A. Fuxman. Active objects: Actions for entity-centric search. In *WWW Conference*, pages 589–598. ACM, 2012.

C. Macdonald and I. Ounis. Voting for candidates: adapting data fusion techniques for an expert search task. In *CIKM*, pages 387–396, 2006. URL <http://doi.acm.org/10.1145/1183614.1183671>.

D. Savenkov and E. Agichtein. When a knowledge base is not enough: Question answering over knowledge bases with external text data. In *SIGIR*, pages 235–244, 2016. URL <http://doi.acm.org/10.1145/2911451.2911536>.

U. Sawant and S. Chakrabarti. Learning joint query interpretation and response ranking. In *WWW Conference*, 2013. URL <http://dl.acm.org/citation.cfm?id=2488484>.

A. Severyn and A. Moschitti. Learning to rank short text pairs with convolutional deep neural networks. In *SIGIR*, 2015.

K. Tymoshenko, D. Bonadiman, and A. Moschitti. Convolutional neural networks vs. convolution kernels: Feature engineering for answer sentence reranking. In *NAACL-HLT*, pages 1268–1278, 2016.

K. Xu, Y. Feng, S. Huang, and D. Zhao. Hybrid question answering over knowledge base and free text. In *COLING 2016, December 11-16, 2016, Osaka, Japan*, pages 2397–2407, 2016a. URL <http://aclweb.org/anthology/C/C16/C16-1226.pdf>.

K. Xu, S. Reddy, Y. Feng, S. Huang, and D. Zhao. Question answering on freebase via relation extraction and textual evidence. In *ACL*, 2016b. URL <http://sivareddy.in/papers/kun2016question.pdf>.

M.-C. Yang, N. Duan, M. Zhou, and H.-C. Rim. Joint relational embeddings for knowledge-based question answering. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 645–650, 2014.

- X. Yao. Lean question answering over freebase from scratch. In *HLT-NAACL*, pages 66–70, 2015.
- X. Yao and B. Van Durme. Information extraction over structured data: Question answering with freebase. In *ACL (1)*, pages 956–966. Citeseer, 2014.
- M. Yazdani, M. Farahmand, and J. Henderson. Learning semantic composition to detect non-compositionality of multiword expressions. In *EMNLP*, pages 1733–1742, 2015.
- W. Yih, M.-W. Chang, X. He, and J. Gao. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *ACL Conference*, July 2015. URL <http://www.aclweb.org/anthology/P15-1128>.
- C. Zhai. Statistical language models for information retrieval. *Synthesis Lectures on Human Language Technologies*, 1(1):1–141, 2008.