Qsearch: Answering Quantity Queries from Text

Vinh Thinh Ho[1], Yusra Ibrahim[1], Koninika Pal[1],
Klaus Berberich[1,2], and Gerhard Weikum[1]

[1] Max Planck Institute for Informatics, Saarbrücken, Germany
[2] Saarland University of Applied Sciences, Saarbrücken, Germany

Abstract. Quantities appear in search queries in numerous forms: companies with annual revenue of at least 50 Mio USD, athletes who ran 200 meters faster than 19.5 s, electric cars with range above 400 miles, and so on. Processing such queries requires the understanding of numbers present in the query to capture the contextual information about the queried entities. Modern search engines and QA systems can handle queries that involve entities and types, but they often fail on properly interpreting quantities in queries and candidate answers when the specifics of the search condition (less than, above, etc.), the units of interest (seconds, miles, meters, etc.) and the context of the quantity matter (annual or quarterly revenue, etc.). In this paper, we present a search and QA system, called Qsearch, that can effectively answer advanced queries with quantity conditions. Our solution is based on a deep neural network for extracting quantity-centric tuples from text sources, and a novel matching model to retrieve and rank answers from news articles and other web pages. Experiments demonstrate the effectiveness of Qsearch on benchmark queries collected by crowdsourcing.

Keywords: Semantic Search, Question Answering, Information Extraction, Quantities

1 Introduction

Motivation. Quantities, such as $2B, 40 mpg or 19.19 s, are more than mere numbers; they express measures like revenue, fuel consumption or time in a race with a numeric value and a corresponding unit. The occurrence of a quantity in the text or a table of a web page is associated with an entity and interpretable only with the surrounding context. For example, in the sentence *"BMW i8 costs about 138k Euros in Germany and has a battery range between 50 and 60 km."*, the quantity €138.000 (after normalization) refers to the price of the car model BMW i8, and the quantity interval [50,60]km denotes the range for that car (note that this is in electric mode only as this is a hybrid car).

Quantities are common in search queries, for example to find a product within a specific price range, cars or mobile phones with desired technical or environmental properties, or athletes who ran a race in a certain time. When a user issues a quantity search query, such as *"Hybrid cars with price under 35,000 Euros and battery range above 100 km"*, she expects the search engine to understand the quantities and to return relevant answers as a list of entities. However, Internet search engines treat quantities largely as strings ignoring their values and unit of measurements. As a result, they cannot handle numeric comparisons, they miss out on units or scale factors (such as "k" in "138k"), do not know about necessary conversions between units, and ultimately fail. The

Table 1: Statistics on exemplary quantitative properties from Wikidata and DBpedia. *#E*: number of entities; *#P*: with property present; *#Q*: with explicit data type for the property.

Entity Type/Property	**Wikidata**			**DBpedia**		
	#E	*#P*	*#Q*	*#E*	*#P*	*#Q*
car model/range	3195	4	4	6705	0	0
car model/engine power	3195	0	0	6705	0	0
mobile phone/display size	291	0	0	1358	1309	0
marathon runner/best time	1629	18	18	3426	1346	601

exceptional cases where search engines (incl. vertical product search) provide support for coping with quantities are money and date, but this is achieved by specialized techniques and fairly limited.

One would hope that semantic search over knowledge graphs (KG) like DBpedia or Wikidata goes further, but their coverage of quantitative facts is very limited and most literals, apart from dates, are merely represented as strings; e.g., battery capacity of the BMW i3 is shown as the string *"i3 94 A·h: 33 kWh lithium-ion battery"* in DBpedia. Important properties for cars, like fuel consumption, CO2 emission, etc. are not covered at all. Table 1 gives exemplary numbers for the quantity coverage in Wikidata and DBpedia.

This paper sets out to provide support for answering quantity queries from text, over a wide variety of expressive measures, to overcome this severe limitation of today's search engines and knowledge graphs. Our method extracts quantity-centric structure from Web contents, uncovering the hidden semantics of linking quantities with entities.

Problem Statement. We define our problem as follows. Given a quantity query and a corpus of text pages, find a ranked list of entities that match the given query. A quantity query is a triple (t^*, q^*, X^*), where t^* is the semantic type of the expected answers, q^* is a quantity-centric search condition, and X^* is the context that connects the entity type t^* with quantity condition q^*. For example, for the query *"Cars with price less than €35,000 in Germany"*, the triple (t^*, q^*, X^*) is: $(cars; < €35.000; \{price, Germany\})$. Our problem has two dimensions. The first is to understand the content of the text snippets and extract the relevant quantity facts. The second is to match such extracted assertions (inevitably with noise and errors) against a query and compute a ranked list of relevant entity answers.

Approach. This paper presents Qsearch, an end-to-end system for answering quantity queries. Qsearch employs a deep neural network to extract quantity facts from text, this way lifting textual information into semantic structures. Then, it utilizes a statistical matching model to retrieve and rank answers.

We model the first component, quantity fact extraction, as a Semantic Role Labeling (SRL) task [13] and devise a deep learning method to label words in the sentences with relevant roles. We label each word as entity, quantity or context (or other). Then we use these tags to extract quantity fact triples in form of *(entity, quantity, context)*. For the second component, query matching, we devise a novel matching method to retrieve a ranked list of relevant entities that answer the user's quantity query.

Contribution. The salient contributions of this work are as follows:

- We present Qsearch, a system for answering quantity queries from text.
- We propose a deep neural network for quantity fact extraction, and a matching model for answering quantity queries.
- We present extensive experiments on benchmark queries collected by crowdsourcing.

2 Computational Model and System Overview

In this section, we introduce the computational model for our approach and give an overview of the Qsearch system and its components.

2.1 Model for Facts, Queries and Answers

Extraction Model. The *input* of this model is a corpus of text documents $\mathcal{T}$ with text snippets (e.g., sentences or paragraphs) that contain entity and quantity mentions.

The *output* of this model is a set of *quantity facts* extracted from the text corpus, $\mathbb{F} = \{\mathcal{F}_1, \mathcal{F}_2, ...\}$, where a quantity fact is defined as follows.

Definition 1 (Quantity fact). *A quantity fact (Qfact) is a triple $\mathcal{F} = (e, q, X)$, where:*
- e is an entity;
- $q = (v, u, r)$ is a quantity consisting of a numerical value v, a canonicalized unit u (e.g., km, $) and a value resolution r (exact, approximate, upper/lower bound, interval);
- $X = \{x_1, x_2, ...\}$ is a context, which is a bag of words describing the relation between e and q.

Example 1. Given the text snippet *"BMW i8 costs about 138k Euros in Germany and has a battery range between 50 and 60 km."*, we can extract the following Qfacts:
- $\mathcal{F}_1 : e =$ BMW i8; $q =$ (138.000, €, approximate); $X = \{$costs, Germany$\}$
- $\mathcal{F}_2 : e =$ BMW i8; $q =$ (50-60, km, interval); $X = \{$range, battery$\}$ □

The Qfact representation is similar to the RDF model [20], which represents each fact as a *(subject, predicate, object)* triple. In the Qfact model, the entity e and the quantity q correspond to the *subject* and the *object*, respectively. The context X in Qfacts is a proxy for the *predicate* in the RDF model. However, it differs in two essential points: first, the context X can capture more than one relation between e and q; second, the context X consists of a set of non-canonicalized tokens, instead of a unique canonicalized predicate in a knowledge graph.

This relaxed representation is a judicious design choice and essential for the flexibility of our approach: first, we can represent complex n-ary facts using a simple Qfact triple; second, our model can generalize to unseen relations; third, our model can cope with the inevitable diversity and uncertainty in the language expressions of the underlying text snippets. In theory, it is conceivable that all arguments that appear in the context X are also individually extracted and canonicalized to fill the slots of a frame-like structured record. However, approaches along these lines do not work robustly and suffer from heavy propagation of noise and errors.

The Qfact model allows different representations of the same fact, and the underlying text corpus may express the same knowledge by different paraphrases. Hence, Qfacts are more expressive towards answering queries via approximate matches and related phrases.

Matching Model. The *input* of this model is a set of Qfacts $\mathbb{F} = \{\mathcal{F}_1, \mathcal{F}_2, ...\}$ extracted from the text corpus, and a *quantity query* $\mathcal{Y}$ defined as:

Definition 2 (Quantity query). *A quantity query (Qquery) is a triple $\mathcal{Y} = (t^*, q^*, X^*)$ where:*
- t^* is the semantic type of the target answers;
- $q^* = (v, u, o)$ is a quantity condition consisting of a numerical value v, a canonicalized unit u (e.g., km, $) , and a comparison operator o (exact, approximate, upper/lower bound, interval);
- $X^* = \{x_1, x_2, ...\}$ is a context condition, expressed by a bag of words that describes the relation between t^* and q^*.

Example 2. Given the query *"Cars with price less than 100k Euros in Germany"*, its corresponding Qquery is as follows:
- $\mathcal{Y}$: t^* = car; q^* = (100.000, €, upper bound); X^* = {price, Germany} □

Each part of a Qquery imposes a constraint on its counterpart in a Qfact considered as a candidate answer.

Definition 3 (Query answer). *A Qfact $\mathcal{F} = (e, q, X)$ is an answer for a Qquery $\mathcal{Y} = (t^*, q^*, X^*)$ iff (1) e is an entity of type t^*, (2) the quantity q satisfies the quantity condition q^* and (3) the context X (approximately) matches the context condition X^*.*

Example 3. Consider the Qquery in Example 2 and the two text segments *"German dealers sell the BMW X3 at a price as low as 55,000 Euros"* and *"Car dealers in Munich sell the BMW X3 starting at 55,000 Euros"*. The Qfact extracted from the first snippet with e = *BMW X3*, q = *(55.000, €, lower bound)*, and X = {*German, dealers, sell, price*} is a strong match for the query; whereas the Qfact extracted from the second snippet with e = *BMW X3*, q = *(55.000, €, lower bound)*, and X = {*car, dealers, Munich, sell*} is an approximate match (by embedding-based relatedness). □

The *output* of this model is a ranked list of entities $\mathcal{E}^* = \{e_1, e_2, e_3, ...\}$ from matching Qfacts with the Qquery, which will be discussed in Section 4.

2.2 Qsearch System

Figure 1 gives an overview of the architecture of Qsearch. The arrows in the figure depict information flow between the different system components. Qsearch consists of two main stages: *Extract* and *Answer*.

Extract. We preprocess the text corpus (Block 1) to recognize and disambiguate named entities and link them to an external knowledge base (KB). We also identify mentions of quantities in the text and normalize them into standard units. Subsequently, we run a deep neural network to extract Qfacts from the preprocessed text (Block 2). We learn and employ a specifically designed Long Short Term Memory (LSTM) network, which will be described in Section 3.

Extracted Qfacts are organized and grouped by their named entities, such that each individual entity e_i is mapped to a list of quantities and related contexts $L_{e_i} = \{(q_{i1}, X_{i1}), (q_{i2}, X_{i2}), ...\}$ (Block 3). All extracted Qfacts are stored in a data repository (Block 4, based on Elasticsearch in our implementation), where entities are linked to their semantic types from the KB.

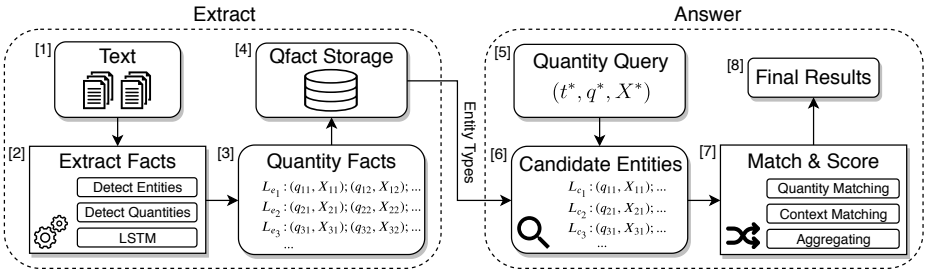

Fig. 1: Overview of Qsearch.

Answer. We answer incoming Qqueries by matching them against the Qfacts from the *Extract* stage. For a Qquery (t^*, q^*, X^*) (Block 5), we first apply an entity-type filter, eliminating entities with the wrong type. This results in a set of candidate entities $\mathcal{C} = \{c_1, c_2, ...\}$ (Block 6) satisfying the type constraint t^*, along with their quantity-context pairs $\{L_{c_1}, L_{c_2}, ...\}$. In Block 7, we discard all candidate answers that do not satisfy the quantity condition q^*. Finally, we compute a matching score for each candidate entity $c \in \mathcal{C}$ based on the contexts X in the quantity-context pairs L_c, using a statistical language model or a text embedding method, which will be described in Section 4. The candidate entities are ranked by their scores and returned to the user (Block 8).

In the following Sections 3 and 4, we discuss in detail the Qfact extraction model and the matching and answering model, respectively.

3 Quantity Fact Extraction From Text

In this section, we describe our method for extracting Qfacts from natural language text. At the core of our solution is a deep-learning neural network for sequence tagging, running on individual sentences.

Input Preprocessing. In the first step, we preprocess the input text corpus by detecting entities and quantities appearing in each individual input sentence. We perform Named Entity Disambiguation (NED) using the AIDA [16] system, which links named entities to the YAGO knowledge base [34]. To achieve a better detection quality, we run NED on a per-document instead of per-sentence basis. For detecting quantities, we make use of the Illinois Quantifier [28], a state-of-the-art tool for recognizing numeric quantities in text, along with some hand-crafted rules (e.g., regular expressions). Subsequently, each identified quantity is replaced by a placeholder *"_QT_"*.

Example 4. Input and output of this preprocessing step look as follows:

sentence | *BMW i8 has price of 138k Euros in Germany and range from 50 to 60 km on battery .*

preprocessed | $\overbrace{\textit{BMW i8}}^{e_1=\text{<KB:BMW\_i8>}}$ *has price of* $\underbrace{\textit{\_QT\_}}_{q_1=(138.000,€,appr.)}$ *in* $\overbrace{\textit{Germany}}^{e_2=\text{<KB:Germany>}}$ *and range* $\underbrace{\textit{\_QT\_}}_{q_2=(50\text{-}60,km,interval)}$ *on battery .* □

Sequence Tagging Model. In the second step, we aim to extract complete Qfacts from the preprocessed sentences. For each quantity detected in the previous step, we want

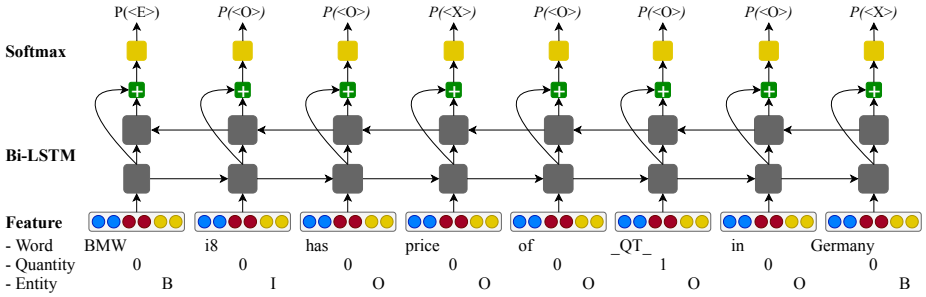

Fig. 2: The Qfact extraction model used by Qsearch.

to identify the entity to which it refers and the relevant context tokens that express the entity-quantity relation.

Example 5. Consider the preprocessed sentence in Example 4. If we use the first quantity q_1 = *(138.000, €, approximate)* as the input's pivot, we want to obtain the output $e(q_1) = e_1$ = <*KB:BMW_i8*> and $X(q_1) = \{$*price, Germany*$\}$. Analogously, with q_2 = *(50-60, km, interval)* as pivot, the desired output is $e(q_2) = e_1$ = <*KB:BMW_i8*> and $X(q_2) = \{$*range, battery*$\}$. □

We formalize this task as a sequence labeling problem as follows.

Task 1 (Quantity Fact Extraction) *Given a preprocessed sentence S with the set of detected entities $\mathcal{E} = \{e_1, e_2, ...\}$, the set of detected quantities $\mathcal{Q} = \{q_1, q_2, ...\}$ and a selected pivot quantity of interest $q_i \in \mathcal{Q}$, the task of quantity fact extraction is to label each token of the sentence with one of the following tags: (i) <E>, for denoting the entity that q_i refers to; (ii) <X>, for denoting the context tokens that relate q_i and its entity; and (iii) <O>, for all other tokens.*

Our problem resembles the Semantic Role Labeling task [13], which is typically addressed by Conditional Random Fields (CRFs) or Long Short Term Memory (LSTM) models. Figure 2 depicts the bi-directional LSTM model that we devised for this task, inspired by prior work [14]. Other models for sequence labeling (e.g., [12, 44]) could be easily incorporated as well. Our labeling network consists of three layers: Input Features, Bi-LSTM, and Softmax. While this general architecture is close to any other LSTM model, the most unique point here is the input representation, as described next.

Input Features. Each token of the preprocessed input sequence is represented as the concatenation of three input feature vectors:

(*i*) *Word*: We include word embeddings as an input feature, which enables the neural model to generalize to different words having similar meanings. In our implementation, we use Glove [24] precomputed embeddings.
(*ii*) *Quantity*: We provide the position of the pivot quantity to the model as input. When sentences contain multiple quantities (which is a relatively frequent case), our model operates one quantity at a time and we re-run the model for different quantities.

(iii) *Entity*: We also provide information about the recognized entities as input to the neural model. As entities often span multiple tokens, we employ the BIO tagging mechanism [26], where a tag *B* is used for tokens at the beginning of an entity name, *I* for tokens inside the name, and *O* for other tokens. With this representation, the output of the model only needs to tag the first token of a multi-word entity name with <*E*>, and subsequent tokens are tagged with <*O*>. Figure 2 shows an example: *"BMW i8"* is chosen as the entity connected with the pivot quantity; only the first token *"BMW"* is tagged as <*E*> in the output.

Output Constrained Decoding. The output of the model are the probabilities of each token word in the input belonging to each of the three tags <*E*>, <*X*> and <*O*>, produced by the Softmax layer. In neural models, usually the tag with the highest score will be assigned to each token word. However, this standard technique would not take into account the dependencies between output tags, and hence might give us an invalid tag sequence. To solve this issue, we impose the following two constraints on the output of the model at decoding time, and find the most probable tag sequence satisfying them: *(i)* only one tag <*E*> can appear in the output (namely, for the one entity to which the pivot quantity refers); and *(ii)* that tag <*E*> has to be at the start token of an entity name.

To find the most probable tag sequence, we use Dynamic Programming to decode from left to right. Specifically, we compute subsequences of tags $Seq_{i,j,k}$ for every $i \in \{1..n\}$ (n is the sentence length); $j \in \{$<*E*>, <*X*>, <*O*>$\}$; and $k \in \{0, 1\}$. Here, $Seq_{i,j,k}$ denotes tag subsequence with the highest probability for tokens from position 1 to position i, where the tag of token at position i is j, and the subsequence contains k <*E*> tags. Note that the probability of a tag subsequence is computed as the product of the probabilities of its constituent tags. The final tag sequence can be derived at $i = n$.

Distant Supervision Training. As training data is an important factor but difficult to obtain, and manual labeling at scale is too expensive, we employ distant supervision to generate training data for the Qfact extraction model. We use unsupervised, pattern-based Open Information Extraction (Open IE) to overlay an n-tuple structure (with triples or higher-arity tuples) on the input text. We employ the OpenIE4 tool [22] to this end, and then use its output tuples to generate training data. This process consists of two steps:

- *Step 1: Capture information areas:* We define an *information area* as a subset of tokens from a sentence, which presents complete information about a fact. We run Open IE on the *unprocessed* sentence to detect all possible tuples expressed by the text. Each of these tuples has a confidence score; to ensure the quality of the generated training samples, we only keep tuples having a confidence score of at least 0.9. Each of the selected tuples corresponds to an information area.

Example 6. Consider the unprocessed sentence in Example 4, suppose the following tuples are extracted by Open IE: (1): *(BMW i8; has; price of 138k Euros; in Germany)*$^{0.95}$, (2): *(BMW i8; has; range from 50 to 60 km on battery)*$^{0.9}$, (3): *(BMW i8; has; price of 138k Euros)*$^{0.8}$, (4): *(BMW i8; has; range from 50 to 60 km)*$^{0.5}$, and (5): *(BMW i8; has; price)*$^{0.1}$. We only keep high-confidence tuples (1) and (2), which contain complete

information. Then the following information areas are chosen for training:

$$\overbrace{\underbrace{\textit{BMW i8 has price of 138k Euros in Germany}}_{(2)}}^{(1)} \textit{and} \underbrace{\textit{range from 50 to 60 km on battery}}_{(2)}. \quad \square$$

- *Step 2: Transform infomation areas into training samples:* We map the information areas obtained in Step 1 with entities and quantities detected from the preprocessing phase:

$$\overbrace{\underbrace{\textit{<KB:BMW\_i8> has price of \_QT}_{-(1)} \textit{in <KB:Germany>}}_{(2)}}^{(1)} \textit{and} \underbrace{\textit{range \_QT}_{-(2)} \textit{on battery}}_{(2)}. \quad \square$$

With this mapping, information areas yield training samples for the neural network. We apply conservative filters so that this self-training process minimizes spurious samples. First, we keep only information areas that contain exactly one quantity *_QT_*, the pivot quantity. Second, since English sentences tend to express quantity information in active voice, the entity connected to the pivot quantity should appear in the first argument (subject) of the Open IE tuple. For instance, information area (1) has two entities *<KB:BMW_i8>* and *<KB:Germany>*; we choose the former as the one to which quantity $\_QT\_(1)$ refers. Finally, we discard all information areas where the subject of the Open IE tuple contains more than one entity.

At this point, for each information area, we have a quantity and a unique entity to which it refers. The context between them is determined from the remaining tokens in the information area based on their Part-of-speech (POS) tags. We allow only the following POS patterns to form the context: noun (NN*), verb (VB*), adjective (JJ*), adverb (RB*), and foreign word (FW, to capture out-of-vocabulary names). We also use pre-defined stopwords to remove uninformative tokens from the context. The resulting Qfact, along with the *<E>*,*<X>*,*<O>* tags for its token sequence, becomes a positive training sample. As negative training samples, we collect all information areas where no entity could be identified to relate with the pivot quantity, i.e., all tokens are tagged as *<O>*.

4 Candidate Fact Matching Model

This section describes our method to answer Qqueries from the extracted Qfacts. To this end, each Qfact is assigned a score denoting its relevance to the given Qquery.

Query Parsing. Input questions are mapped into Qqueries by a rule-based parser for recognizing answer type and quantity condition; all other tokens (except stopwords) are included in the query context. The parser uses a dictionary of YAGO types and a dictionary of quantity units. An alternative to this rule-based technique would be to apply the same neural extraction method to questions that we have used to extract Qfacts from text. However, the questions are easier to handle, and the rule-based parser works well.

Task 2 (Quantity Fact Scoring) *Given Qquery* $\mathcal{Y} = (t^*, q^*, X^*)$ *and Qfact* $\mathcal{F} = (e, q, X)$*, compute a distance score* $d(\mathcal{F}, \mathcal{Y})$ *reflecting the relevance of* $\mathcal{F}$ *regarding* $\mathcal{Y}$*.*

Without loss of generality, we assume that a lower score denotes a better fact. $\mathcal{F}$ should be a high-ranked answer for $\mathcal{Y}$ iff the following three conditions hold: (1) e is an entity of type t^*, (2) q satisfies q^*, and (3) X is a good (approximate) match for X^*.

Entity - Type Matching. We only consider the Qfact $\mathcal{F}$ if the entity e has type t^*. Since the entities from text are linked to an external knowledge base, we make use of the type information from the KB to filter out unsuitable facts for $\mathcal{Y}$.

Quantity Matching. We also discard $\mathcal{F}$ if q does not satisfy q^*. This is the case when either (1) the units of q and q^* relate to different concepts (e.g. *km* (length) vs. €(money)) and are thus incomparable; or (2) their values (after conversion to the same unit) do not match the comparison operator of q^*. Since quantity matching is not the focus of our paper, we apply a simple matching method as follows. First, we use hand-crafted rules for unit conversions, re-scaling if needed (e.g., for kilo, mega, etc.), and value normalization. Second, we turn the quantity value into an interval based on its resolution. For example, when the query is about approximate matches, a quantity value v is smoothed into the interval $[v-\delta, v+\delta]$ with a configuration parameter δ. In experiments, we set δ to 5% of v. A comparison is considered a match when the two intervals overlap, and their units (after conversion and re-scaling) match.

Context Matching. If the Qfact $\mathcal{F}$ satisfies the above two constraints, we will consider the similarity between the query context X^* and the fact context X. We propose to use the following two approaches for measuring the context relevance: a *probabilistic* and an *embedding-based* approach.

Probabilistic Ranking Model. We adopt the Kullback-Leibler (KL) divergence between the query context X^* and the fact context X, which is typically used in statistical language models [42]. The scoring function is defined as follows:

$$\begin{aligned} d(\mathcal{F}, \mathcal{Y}) = KL(X^*, X) &= H(X^*, X) - H(X^*) \\ &\equiv H(X^*, X) = -\sum_{w \in \mathcal{V}} P(w|X^*)\ \log P(w|X) \end{aligned}$$

where $\mathcal{V}$ is the word vocabulary, $H(X^*)$ is the entropy of X^*; $H(X^*, X)$ is the cross entropy between X^* and X; and $\equiv$ indicates rank equivalence (i.e., preserving order). Since we are only interested in ranking fact contexts in response to a query context, we can omit $H(X^*)$. The word probability $P(w|X^*)$ for the query context is estimated using Maximum Likelihood Estimation (MLE) on an expanded version X_E^* of X^* as:

$$P(w|X^*) = count(w \in X_E^*)/|X_E^*|$$

To expand a query context, we resort to WordNet [23] and add all synonyms of the context words to it. For the fact context, we estimate the word probability $P(w|X)$ using Jelinek-Mercer smoothing as:

$$P(w|X) = (1-\lambda) \times count(w \in X)/|X| + \lambda \times P(w|B)$$

This linearly combines the MLE from the fact context X with the MLE obtained from a background corpus B. The smoothing parameter λ (set to $\lambda = 0.1$ in our system) controls the influence of the background corpus on the probability estimate. We construct the background corpus B from all sentences of the entire text corpus that contain at least one

quantity (total 39M sentences in our data).

Embedding-based Ranking Model: We observed on our data that the query context X^* is often shorter than the fact context X, since sentences are often more verbose than the typically short queries. Hence, to measure the distance score of X with regard to X^*, we can match tokens between X^* and X using word embedding similarity as follows:

$$d(\mathcal{F},\mathcal{Y}) = \Big(\sum_{u\in X^*} \min_{v\in X}(dist(u,v))\Big)/|X^*|$$

where $dist(u,v) \geq 0$ is the semantic distance between two words u and v estimated from their pre-computed word embedding vectors [24]. We use cosine distance in the Qsearch implementation, re-scaled for normalization to [0,1]. In the above equation, we map each word of query context X^* to its closest word in the fact context X in the embedding space. This scoring formula gives the same weight to every token in the query context X^*, which might be misleading, since they could have a different degree of importance. This issue is overcome by giving higher weight to important words and lower weight to uninformative words, using the following distance function:

$$d(\mathcal{F},\mathcal{Y}) = \frac{\sum_{u\in X^*} W(u) \min_{v\in X}(dist(u,v))}{\sum_{u\in X^*} W(u)} + 1$$

where $W(u) \geq 0$ is the importance weight of word u. There are several weighting functions that can be used for W (e.g., *inverse document frequency (idf)*, *term strength*, etc.); we use Robertson's *idf* [27]. We call the above formula the *directed embedding distance*, $ded(X^* \rightarrow X)$, between query and fact contexts.

$ded(X^* \rightarrow X)$ describes how well each word in X^* matches with some other word in X, but in many cases it fails to reflect the match between their meaning. The presence of a single word in the fact context X can totally change its meaning. Consider, as a concrete example, the two contexts $X^* = \{$*net, worth*$\}$ vs. $X = \{$*negative, net, worth*$\}$. Hence, our idea is to penalize the relevance score with an amount proportional to the directed embedding distance between X and X^*. Specifically, we define the *context embedding distance (ced)* that implements this idea:

$$\begin{aligned} d(\mathcal{F},\mathcal{Y}) &= ced(X^*,X) = ded(X^* \rightarrow X) \times ded(X \rightarrow X^*)^\alpha \\ &= \left(\frac{\sum_{u\in X^*} W(u) \min_{v\in X}(dist(u,v))}{\sum_{u\in X^*} W(u)} + 1\right) \times \left(\frac{\sum_{u\in X} W(u) \min_{v\in X^*}(dist(u,v))}{\sum_{u\in X} W(u)} + 1\right)^\alpha \end{aligned}$$

Intuitively, our ced measure is the product of two components: (1) $ded(X^* \rightarrow X)$ captures how well query context tokens match with fact context, and (2) $ded(X \rightarrow X^*)$ reflects how much additional terms in X shift its meaning, and hence, should be penalized. Parameter $\alpha \in [0, +\infty)$ controls how much the penalty scaling affects the total score.

Example 7. Consider the Qquery context $X^* = \{$*gross, domestic, product*$\}$ and two Qfact contexts $X_1 = \{$*gross, national, product*$\}$, $X_2 = \{$*gross, domestic, product,*

capita}. While we are more inclined to X_1 than X_2, the directed embedding distance $ded(X^* \rightarrow X_2)$ has a slightly better score than $ded(X^* \rightarrow X_1)$, as it does not penalize the word *"capita"* (which indicates that the GDP is per capita, not the total GDP). In contrast, $ded(X_1 \rightarrow X^*)$ is lower than $ded(X_2 \rightarrow X^*)$ (since *"national"* is close to *"domestic"*), preferring X_1 over X_2 with regard to X^*, which results in the desired ranking based on the context embedding distance ced. □

Entity Scoring. The output of Qsearch is a ranked list of entities from matching Qfacts with the Qquery. We assign a score for each candidate entity based on one of the above context distance models and aggregating over the entity's quantity-context pairs as follows:

$$\textit{score}(c \in \mathcal{C}, \mathcal{Y}) = \min_{(q,X) \in L_c} d(\mathcal{F} = (c, q, X), \mathcal{Y})$$

where $d(\mathcal{F}, \mathcal{Y})$ is either the Kullback-Leibler divergence $KL(X^*, X)$ or the context embedding distance $ced(X^*, X)$. So when the same candidate entity appears in multiple Qfacts, we pick the best-scoring Qfact context distance.

5 Evaluation

We run experiments on a Linux machine with 80 CPU cores, 500GB RAM, and 2 GPUs. To evaluate Qsearch, we perform an intrinsic evaluation of our *Qfact extraction model* and an extrinsic evaluation of the *end-to-end Qsearch system*.

Dataset. All experiments use a large collection of news articles, compiled from two real world datasets: the *STICS* project [15] with news from 2014 to 2018, and the *New York Times* archive [30] with news from 1986 to 2008. In total, our corpus consists of 7.6M documents.

5.1 Intrinsic Evaluation of the Quantity Fact Extraction Model

Training setup. We implemented the LSTM network using Theano library, largely following [14] for the training configuration: using Adadelta with $\epsilon = 1e^6$ and $\rho = 0.95$; *lstm_hidden_unit = 300*; *rnn_dropout_prob = 0.1*; *batch_size = 100*.

We extracted training samples from the corpus using the distant-supervision technique as described in Section 3 and conducted the training process with different settings. In the *General* setting, we use all available training data of 3.2M training samples, where we maintain the ratio 3:1 between the number of positive and negative samples. We also train our model for three other *measure-specific* settings, where only a subset of the training samples is used. In particular, we classify training samples into different categories based on the quantity unit. For example, training samples containing quantities with unit *Kilometer* or *Meter* are chosen to train the model in the *Length* setting, while the ones with unit *US dollar*, *Euro*, etc. are picked for the *Money* setting. Among many such categories, we selected the three most prevalent measures *Money*, *Percentage* and *Length*, containing 307K, 235K and 41K training samples, respectively (also with ratio 3:1 between positive and negative samples). The trained models are then applied to the entire corpus to extract more Qfacts.

Table 2: Evaluation of Qfact extraction model of Qsearch on different settings.

Tag	Length			Money			Percentage			General		
	Prec.	*Rec.*	*F1*	*Prec.*	*Rec.*	*F1*	*Prec.*	*Rec.*	*F1*	*Prec.*	*Rec.*	*F1*
E	0.860	0.860	0.860	0.850	0.850	0.850	0.794	0.770	0.782	0.882	0.820	0.850
X	0.650	0.849	0.736	0.717	0.844	0.776	0.659	0.827	0.734	0.728	0.713	0.721
O	0.958	0.886	0.920	0.942	0.886	0.913	0.947	0.888	0.917	0.895	0.906	0.900
Macro-avg.	0.823	0.865	0.839	0.836	0.860	0.846	0.800	0.828	0.811	0.835	0.813	0.824

Table 3: Statistics of benchmark queries from each domain.

Domain	Distribution of queries based on unit of quantity				
	Money	*Length*	*Percentage*	*Others*	*Examples for Others*
Finance	76 %	-	12 %	12 %	no. of sales, albums, etc.
Transport	4 %	32 %	-	64 %	MPG, mph, horsepower, etc.
Sports	8 %	32 %	-	60 %	sec, years, kg, no. of medals, etc.
Technology	20 %	20 %	8 %	52 %	megapixels, Watt, mAh , etc.

Performance of Extraction model. As the test data does not have any ground-truth labels, we randomly selected 100 samples that contain at least two entities from the output tag sequences, for each training model, and manually assessed their validity. We evaluate the quality of the three output labels <*E*>, <*X*> and <*O*> by three measures: *Precision*, *Recall*, and *F1 score*. The results are shown in Table 2. We observe that all training models perform very well on entity tagging with more than 85% *F1 score*. We also see that the measure-specific training variants for Length and Money have slight advantages.

5.2 Extrinsic Evaluation of the End-to-End Qsearch System

We performed the extrinsic evaluation of Qsearch on a benchmark of 100 quantity queries, collected by crowdsourcing and covering four domains: *Finance*, *Transport*, *Sports* and *Technology*. These queries capture a wide diversity of measures and units as well as variety in query formulations (e.g., phrases for the comparison operators); see Table 3. Anecdotal examples of user queries and their answers produced by Qsearch are shown in Table 4. We also considered queries from the QALD-6-task-3 statistical QA benchmark [37], but out of total 150 training and test queries, we found only 6 with quantity conditions (as opposed to simpler property lookups).

In this evaluation, we use the Qfact extraction model trained under the *General* setting, as it generalizes to different measures and units.

Setup. For each Qquery, we consider top-10 results returned by Qsearch and evaluate their relevance and validity by judgements from crowd-workers (using Figure-Eight platform, formerly known as CrowdFlower). The judges were shown the query, the top-10 entity answers, and the corresponding 10 sentences from which the answers were

Table 4: Anecdotal examples of quantity queries and results from Qsearch.

Domain	Query
Finance	**Q1:** Coal companies with more than 200 Million dollar annual profit
Transport	**Q2:** Sport utility vehicles with engine power at least 150 horsepower
Sports	**Q3:** Sprinters who ran 100 meter in less than 10 seconds
Technology	**Q4:** Digital cameras with focal length of lens more than 18 mm

Query	Result	Corresponding Sentence
Q1	Duke Energy	Duke Energy had revenue of $ 23.9 billion and profit of $ 1.9 billion last year.
Q2	Ford Escape	Its V-6 engine (the Escape is a four-cylinder) has 270 horsepower, 20 percent more than the Lexus RX330.
Q3	Andre Grasse	Andre De Grasse, a 20-year-old from Markham, Ont., has run the 100 metre in under 10 seconds three times this year.
Q4	Nikon D7100	For example, the D7100 can be found in a kit with 18-140 mm and 55-300 mm lenses , so you'll want to use the 55-300 mm and zoom in to 300 mm.

extracted. Each result was annotated as *relevant* or *irrelevant* to the query based on the cue given in its corresponding sentence. For each query, we collected three judgements and used the majority label as gold standard. Overall, we obtained a high inter-annotator agreement with Fleiss' Kappa value of 0.54.

Baselines. Although our Qsearch system produces entities as main result, we still want to compare it with standard search systems, which produce snippets. As there is no other system that can handle quantity queries with crisp entity answers, we use search systems as baselines that produce text snippets as answers. Specifically, we ran all benchmark queries on Elasticsearch, locally indexing all sentences of our news corpus, and on Google web search retrieving the top-10 result snippets. Elasticsearch uses a text-oriented state-of-the-art ranking model based on BM25.

The baselines were given certain advantages, to avoid that Qsearch could be viewed as an unfair competitor. For Elasticsearch, we consider only sentences that contain an entity and a quantity. For the evaluation, we asked crowd-workers to annotate top-10 results, retrieved from Elasticsearch, as relevant or irrelevant based on whether they spotted a reasonable result for the quantity query. To evaluate result snippets from Google search, we instructed annotators to be generous, as the result snippets are not well-formed sentences (but could be synthesized from non-contiguous text segments with ellipses). For example, a text snippet that contains a correct entity and its quantity is considered relevant even if it also contains other entities or quantities. Such instructions to annotators give Google results an advantage because Qsearch results are considered relevant only if both entity and quantity are correctly extracted.

We also explored several state-of-the-art QA systems over linked open data: Frankenstein [32], QAnswer [9], Platypus [35], AskNow [10], Quint [1], SPARKLIS [11]. None of these systems is geared for handling quantity questions, except SPARKLIS, however it can only process quantities without associated unit. Moreover, their underlying KBs have poor coverage of quantities. They failed on almost all of our benchmark queries; so we excluded these systems from our comparative evaluation.

Table 5: End-to-end evaluation of Qsearch.

Metric	Finance		Transport		Sports		Technology		All	
	KL-div.	*Emb.*	*KL-div.*	*Emb.*	*KL-div.*	*Emb.*	*KL-div.*	*Emb.*	*KL-div.*	*Emb.*
Pr.@1	0.720	0.800	0.480	0.600	0.560	0.680	0.640	0.680	0.600	0.690
Pr.@3	0.667	0.747	0.480	0.480	0.507	0.587	0.627	0.653	0.570	0.617
Pr.@5	0.632	0.672	0.412	0.412	0.480	0.528	0.550	0.624	0.519	0.559
Pr.@10	0.604	0.608	0.333	0.379	0.412	0.432	0.500	0.547	0.462	0.492
Hit@3	0.880	0.920	0.760	0.760	0.760	0.800	0.840	0.880	0.810	0.840
Hit@5	0.880	0.960	0.760	0.760	0.920	0.840	0.840	0.920	0.850	0.870
MRR	0.792	0.870	0.621	0.678	0.685	0.746	0.747	0.783	0.711	0.769

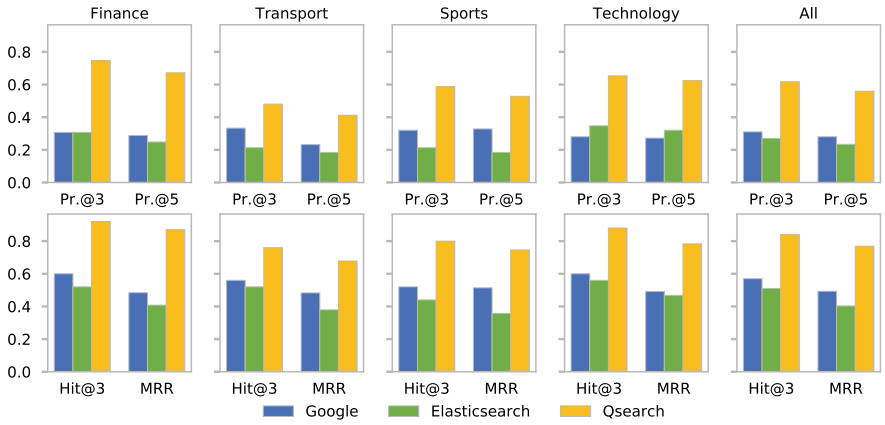

Fig. 3: Comparison of Qsearch against baselines.

Performance of Qsearch. Table 5 shows the performance of Qsearch for the four domains and for all 100 queries together, using the two variants of our ranking models: KL divergence and context embedding distance (ced). For ced we empirically tune the parameter $\alpha = 3$ based on results from 10 validation queries disjoint from the 100 test queries. We report three metrics: *Precision@k*, *Hit@k* and *Mean-Reciprocal-Rank (MRR)*, macro-averaged over queries. We do not discuss metrics like Recall or MAP, as these would require exhaustively annotating a huge pool of candidate answers.

Overall, Qsearch performs amazingly well, typically with MRR around 0.7 or better. The best results are for the Finance domain, which has the highest share in the corpus and is most represented in the Qfact extraction training. *Precision@1* is pretty good, but precision drops substantially when going deeper in the rankings. The embedding-based ranking model clearly outperformed the KL-divergence method by a significant margin.

Figure 3 presents the comparison of Qsearch with the ced ranking model against Elasticsearch and Google, showing the metrics *Prec.@3*, *Prec.@5*, *Hit@3* and *MRR*. The results clearly indicate that Qsearch outperforms both baselines by a large margin.

6 Related Work

Question Answering. QA over knowledge graphs and other linked data sources has received great attention over the last years; see [8, 36] for surveys. State-of-the-art methods (e.g., [2, 5, 39, 41, 43]) translate questions into SPARQL queries, bridging the gap between question vocabulary and the terminology of the underlying data by means of templates and/or learning from training collections of question-answer pairs. Benchmarks like the long-standing QALD series and other competitions have shown great advances along these lines [38]. However, these benchmark tasks hardly contain any quantity queries of the kind addressed here (even in QALD-6-task-3, only 6 out of 150 questions are of this kind, others are mostly about quantity lookup). Note that look-ups of quantity attributes of qualifying entities (e.g., Jeff Bezos's net worth, 10 richest people, or fastest sprinter over 100m) are of a different nature, as they do not contain quantity comparisons between query and data (e.g., worth more than 50 million USD, running faster than 9.9 seconds). Moreover, the scope and diversity of the benchmark queries is necessarily restricted to relatively few numeric properties, as knowledge graphs hardly capture quantities in their full extent (with value and unit properly separated and normalized). This is our motivation to tap into text sources with more extensive coverage.

QA over text has considered a wide range of question types (e.g., [6, 7, 40]), but there is again hardly any awareness of quantity queries. Keyword search, including telegraphic queries, with quantity conditions have been considered by [18], and have been applied to web tables [25, 31].

[4] and its follow-up work [31] focused on a specific kind of quantity query, namely, retrieving and aggregating numerical values associated with an attribute of a given entity (e.g., Bezos's net worth or GDP of India). To this end, learning-to-rank techniques over value distributions were developed to counter the uncertainty in the retrieved values, where web pages often contain crude estimates and lack exact values. In contrast to our setting, that work did not consider quantities in search conditions.

Information Extraction. Recognizing and extracting numeric expressions from text has been addressed using techniques like CRFs and LSTMs (e.g., [3, 21, 29]). However, this alone does not turn numbers into interpretable quantities, with units and proper reference to the entity with that quantity. Only few works attempted to canonicalize quantities by mappings to hand-crafted knowledge bases of measures [17], but these efforts are very limited in scope. The special case of temporal expressions has received substantial attention (e.g., [33]), but this solely covers dates as measures.

Most related to our approach are the works of [31] and [28]. The former used probabilistic context-free grammars to infer units of quantities, but focused specifically on web tables as inputs. The latter extended semantic role labeling (see below) to extract quantities and their units from natural language sentences. Neither of these can be readily applied to extracting quantities and their reference entities from arbitrary textual inputs.

Semantic Role Labeling. Semantic role labeling (SRL) has been intensively researched as a building block for many NLP tasks [13]. Given a verb phrase of a sentence viewed as a central predicate, SRL identifies phrases that are assigned to pre-defined roles to form a frame-like predicate-arguments structure. Modern SRL methods make use

of pre-computed word embeddings and employ deep neural networks for role filling (e.g., [12, 14, 44]). Our approach differs from this state-of-the-art SRL, as we are not primarily focused on the verb-phrase predicate, but consider the numeric quantity in a sentence as the pivot and aim to capture quantity-specific roles.

To support exploration of quantitative facts in financial reports, [19] proposed a semantic representation for quantity-specific roles. [28] devised a quantity representation as an additional component of an SRL method, which is part of the Illinois Curator software suite. Our approach makes use of this technique, as a preprocessing step. However, we go further by learning how to connect quantities with their respective entities and to collect relevant context cues that enable our matching and ranking stage for query answering.

7 Conclusion

Awareness of entities and types has greatly advanced semantic search both for querying the web of linked data and for Internet search engines. In contrast, coping with quantities in text content and in query constraints has hardly received any attention, yet is an important case. This paper has presented the Qsearch system for full-fledged support of quantity queries, through new ways of information extraction and answer matching and ranking. We capture quantities in their full extent, including units of measures, reference entities and the relevant contexts. The model for Qfacts and Qqueries is relatively simple but highly versatile and effective. A key asset of Qsearch is its high quality in extracting Qfacts, recognizing the right entity-quantity pairs even in complex sentences.

Future work includes devising additional ways of aggregating Qfacts with the same candidate answer, so as to obtain strong signals from many noisy cues (i.e., when the same entity-quantity pair occurs in many pages, but mostly in the form of crude estimates or vague hints). Also, we plan to extend the Qquery model to incorporate queries that contain multiple quantity conditions (e.g., hybrid SUVs with range above 500 miles and energy consumption above 40 MPGe).

References

1. A. Abujabal et al. QUINT: interpretable question answering over knowledge bases. EMNLP 2017.
2. A. Abujabal et al. Never-ending learning for open-domain question answering over knowledge bases. WWW 2018.
3. O. Alonso, T. Sellam. Quantitative information extraction from social data. SIGIR 2018.
4. S. Banerjee et al. Learning to rank for quantity consensus queries. SIGIR 2009.
5. H. Bast, E. Haussmann. More accurate question answering on freebase. CIKM 2015.
6. D. Chen et al. Reading wikipedia to answer open-domain questions. ACL 2017.
7. C.Clark, M.Gardner. Simple and effective multi-paragraph reading comprehension. ACL 2018.
8. D. Diefenbach et al. Core techniques of question answering systems over knowledge bases: a survey. *Knowl. Inf. Syst.* 2018.
9. D. Diefenbach et al. Qanswer: A question answering prototype bridging the gap between a considerable part of the LOD cloud and end-users. WWW 2019.
10. M. Dubey et al. Asknow: A framework for natural language query formalization in SPARQL. ESWC 2016.

11. S. Ferré. Sparklis: An expressive query builder for SPARQL endpoints with guidance in natural language. *Semantic Web* 2017.
12. N. FitzGerald et al. Semantic role labeling with neural network factors. EMNLP 2015.
13. D. Gildea, D. Jurafsky. Automatic labeling of semantic roles. *Comp. Linguistics* 2002.
14. L. He et al. Deep semantic role labeling: What works and what's next. ACL 2017.
15. J. Hoffart et al. STICS: searching with strings, things, and cats. SIGIR 2014.
16. J. Hoffart et al. Robust disambiguation of named entities in text. EMNLP 2011.
17. Y. Ibrahim et al. Making sense of entities and quantities in web tables. CIKM 2016.
18. M. Joshi et al. Knowledge graph and corpus driven segmentation and answer inference for telegraphic entity-seeking queries. EMNLP 2014.
19. M. Lamm et al. Qsrl : A semantic role-labeling schema for quantitative facts. arXiv 2018.
20. O. Lassila, R. R. Swick. Resource description framework (RDF) model and syntax specification. 1999.
21. A. Madaan et al. Numerical relation extraction with minimal supervision. AAAI 2016.
22. Mausam. Open information extraction systems and downstream applications. IJCAI 2016.
23. G. A. Miller. Wordnet: A lexical database for english. *CACM* 1995.
24. J. Pennington et al. Glove: Global vectors for word representation. EMNLP 2014.
25. R. Pimplikar, S. Sarawagi. Answering table queries on the web using column keywords. *PVLDB* 2012.
26. L. Ramshaw, M. Marcus. Text chunking using transformation-based learning. VLC@ACL 1995.
27. S. Robertson. Understanding inverse document frequency: on theoretical arguments for IDF. *Journal of Documentation* 2004.
28. S. Roy et al. Reasoning about quantities in natural language. *TACL* 2015.
29. S. Saha et al. Bootstrapping for numerical open IE. ACL 2017.
30. E. Sandhaus. The new york times annotated corpus, 2008.
31. S. Sarawagi, S. Chakrabarti, Open-domain quantity queries on web tables: annotation, response, and consensus models. KDD 2014.
32. K. Singh et al. Why reinvent the wheel: Let's build question answering systems together. WWW 2018.
33. J. Strötgen, M. Gertz. *Domain-Sensitive Temporal Tagging*. Morgan & Claypool 2016.
34. F. M. Suchanek et al. Yago: a core of semantic knowledge. WWW 2007.
35. T. P. Tanon et al. Demoing platypus - A multilingual question answering platform for wikidata. ESWC 2018.
36. C. Unger et al. An introduction to question answering over linked data. Reasoning Web 2014.
37. C. Unger et al. 6th Open Challenge on Question Answering over Linked Data (QALD-6). SemWebEval@ESWC 2016.
38. R. Usbeck et al. Benchmarking question answering systems. *Semantic Web* 2019.
39. K. Xu et al. Question answering on freebase via relation extraction and textual evidence. ACL 2016.
40. Z. Yang et al. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. EMNLP 2018.
41. W. Yih et al. Semantic parsing via staged query graph generation: Question answering with knowledge base. ACL 2015.
42. C. Zhai. Statistical language models for information retrieval. *F&T in IR* 2008.
43. W. Zheng et al. Question answering over knowledge graphs: Question understanding via template decomposition. *PVLDB* 2018.
44. J. Zhou, W. Xu. End-to-end learning of semantic role labeling using recurrent neural networks. ACL 2015.