Supplementary Material: The Web as a Knowledge-base for Answering Complex Questions

Alon Talmor
Tel-Aviv University
alontalmor@mail.tau.ac.il

Jonathan Berant
Tel-Aviv University
joberant@cs.tau.ac.il

1 Dataset

Generating SPARQL queries Given a SPARQL query r, we create four types of more complex queries: conjunctions, superlatives, comparatives, and compositions. For conjunctions, superlatives, and comparatives, we identify SPARQL queries in WEBQUESTIONSSP whose denotation is a set $\mathcal{A}, |\mathcal{A}| \geq 2$, and generate a new query r' whose denotation is a strict subset $\mathcal{A}', \mathcal{A}' \subset \mathcal{A}, \mathcal{A}' \neq \phi$. We also discard questions that contain the answer within the new machine-generated questions.

For conjunctions this is done by traversing the KB and looking for SPARQL triplets that can be added and will yield a valid set $\mathcal{A}'$.

For comparatives and superlatives this is done by finding a numerical property common to all $a \in \mathcal{A}$, and adding a clause to r accordingly.

For compositions, we find an entity e in r, and replace e with a variable y and add to r a clause such that the denotation of the clause is $\{e\}$. We also check for discard ambiguous questions that yield more than one answer for entity e.

Table 1 gives the exact rules for generation.

Machine-generated (MG) questions To have AMT workers paraphrase SPARQL queries into natural language, we need to present them in an understandable form. Therefore, we automatically generate a question they can paraphrase. When we generate SPARQL queries, new predicates are added to the query (Table 1). We manually annotate 503 templates mapping predicates to text for different compositionality types (with 377 unique KB predicates). We annotate the templates in the context of several machine-generated questions to ensure that they result templates are in understandable language.

We use those templates to modify the original WEBQUESTIONSSP question according to the meaning of the generated SPARQL query. E.g., the template for ?x `ns:book.author.works_written obj` is *"the author who wrote OBJ"*. Table 2 shows various examples of such templates. *"Obj"* is replaced in turn by the actual name according to Freebase of the object at hand. Freebase represents events that contain multiple arguments using a special node in the knowledge-base called CVT that represents the event, and is connected with edges to all event arguments. Therefore, some of our templates include two predicates that go through a CVT node, and they are denoted in Table 2 with '+'.

1. Seed Question	*What movies have robert pattinson starred in?*
2. Original SPARQL	ns:rebert_pattinson ns:film.actor.film ?c . ?c ns:film.performance.film ?x .
3. New SPARQL Term	?x ns:film.film.produced_by ns:erwin_stoff
4. New Term Template	The movie that was produced by obj
5. Machine-generated	*What movies have robert pattinson starred in and is* the movie *that was produced by Erwin Stoff?*
6. Natural language	*Which Robert Pattinson film was produced by Erwin Stoff?*

Figure 1: Overview of data collection process. Blue text denotes different stages of the term addition, green represents the obj value, and red the intermediate text to connect the new term and seed question

To fuse the templates with the original WEBQUESTIONSSP natural language questions, templates contain lexical material that glues them back to the question conditioned on the compositionality type. For example, in CONJ questions we use the coordinating phrase *"and is"*, so that *"the author who wrote OBJ"* will produce *"Who was born in London and is the author who wrote OBJ"*.

First word distribution We find that in WEBQUESTIONS almost all questions start with a wh-word, but in COMPLEXWEBQUESTIONS 22%

Composit.	Complex SPARQL query r'	Example (natural language)
CONJ.	r. $?x$ pred_1 obj. **or** r. $?x$ pred_1 $?c$. $?c$ pred_2 obj.	*"What films star Taylor Lautner and have costume designs by Nina Proctor?"*
SUPER.	r. $?x$ pred_1 $?n$.ORDER BY DESC($?n$) LIMIT 1	*"Which school that Sir Ernest Rutherford attended has the latest founding date?"*
COMPAR.	r. $?x$ pred_1 $?n$. FILTER $?n < V$	*"Which of the countries bordering Mexico have an army size of less than 1050?"*
COMP.	$r[e/y]$. $?y$ pred_1 obj.	*"Where is the end of the river that originates in Shannon Pot?"*

Table 1: Rules for generating a complex query r' from a query r ('.' in SPARQL corresponds to logical and). The query r returns the variable $?x$, and contains an entity e. We denote by $r[e/y]$ the replacement of the entity e with a variable $?y$. pred_1 and pred_2 are any KB predicates, obj is any KB entity, V is a numerical value, and $?c$ is a variable of a CVT type in Freebase which refers to events. The last column provides an example for a NL question for each type.

Freebase Predicate	Template
ns:book.author.works_written	*"the author who wrote obj"*
ns:aviation.airport.airlines + ns:aviation.airline_airport_presence.airline	*"the airport with the obj airline"*
ns:award.competitor.competitions_won	*"the winner of obj"*
ns:film.actor.film + ns:film.performance.film	*"the actor that played in the film obj"*

Table 2: Template Examples

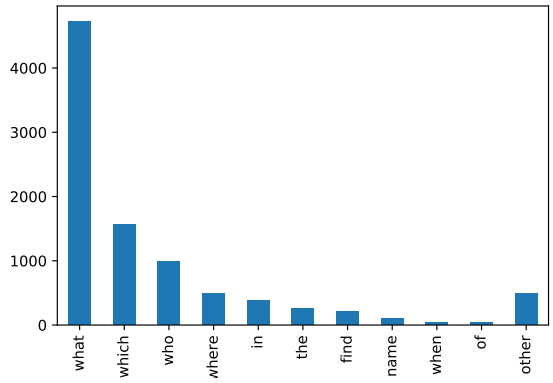

Figure 2: First word in question distribution

of the questions start with another word, again showing substantial paraphrasing from the original questions. Figure 2 Shows the distribution of first words in questions.

2 Generating noisy supervision

We created a heuristic for approximating the amount of global word re-ordering performed by AMT workers and creating noisy supervision. For every question, we constructed a matrix A, where A_{ij} is the similarity between token i in the MG question and token j in the NL question. Similarity is 1 if lemmas match, or the cosine distance according to GloVe embedding, when above a threshold, and 0 otherwise. This allows us to compute an approximate word alignment between the MG question and the NL question tokens and assess whether word re-ordering occurred.

For a natural language CONJ question of length n and a machine-generated question of length m with a known split point index r, the algorithm first computes the best point to split the NL question assuming there is no re-ordering. This is done iterating over all candidate split points p, and returning the split point p_1^* that maximizes:

$$\sum_{0\le i<p} \max_{0\le j<r} A(i,j) + \sum_{p\le i<n} \max_{r\le j<m} A(i,j) \quad (1)$$

We then compute p_1^* by trying to find the best split point, assuming that there is re-ordering in the NL questions:

$$\sum_{0\le i<p} \max_{r\le j<m} A(i,j) + \sum_{p\le i<n} \max_{0\le j<r} A(i,j) \quad (2)$$

We then determine the final split point and whether re-ordering occurred by comparing the two values and using the higher one.

In COMP questions, two split points are returned, representing the beginning and end of the phrase that is to be sent to the QA model. Therefore, if r_1, r_2 are the known split points in the machine-generated questions, we return p_1, p_2 that maximize:

$$\sum_{0\le i<p_1} \max_{0\le j<r_1} A(i,j) + \sum_{p_1\le i<p_2} \max_{r_1\le j<r_2} A(i,j) + \sum_{p_2\le i<n} \max_{r_2\le j<m} A(i,j).$$

Figure 3 illustrates finding the split point for a CONJ questions by using equation (2). The red line in Figure 3 corresponds to the known split point in the MG question, and the blue one is the estimated split point p^* in the NL question.

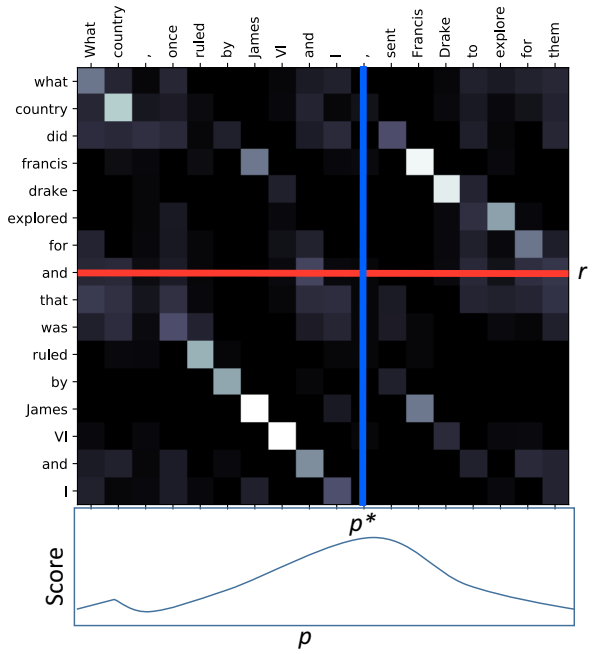

Figure 3: Heat map for similarity matrix between an MG and NL question. The red line indicates a known MG split point. The blue line is the approximated NL split point. Below is a graph of each candidate split point score.