Paper Introduction

Learning to Answer Complex Questions over Knowledge Base with Query Composition CIKM 2019

Nikita Bhutani* Xinyi Zheng* H V Jagadish
University of Michigan, Ann Arbor

KDE Mining Seminar

Speaker: Happy Buzaaba, D2

25th Dec 2019

Outline

- Introduction
- Preliminaries
- Query Candidate Generation Process
- Semantic Matching
- Implicit Supervision
- Experiment, Results and Discussion
- Conclusion and Future Work

Outline

- Introduction
- Preliminaries
- Query Candidate Generation Process
- Semantic Matching
- Implicit Supervision
- Experiment, Results and Discussion
- Conclusion and Future Work

Introduction

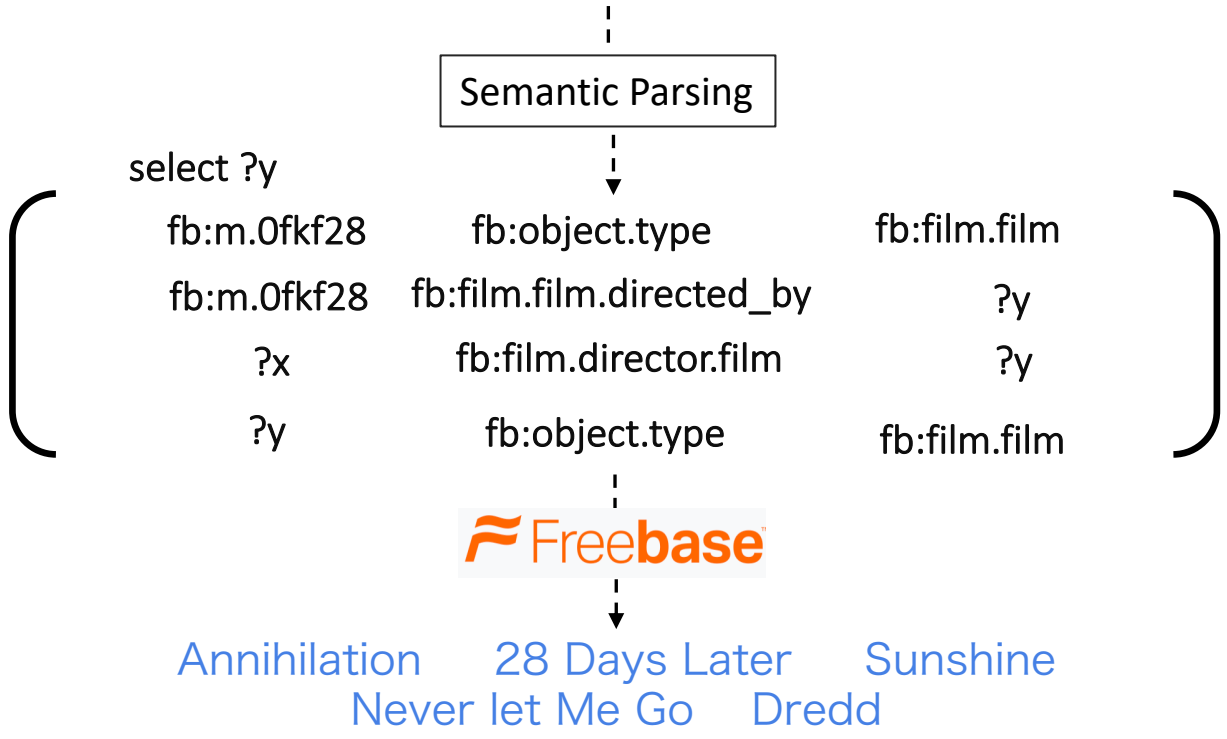
- KB-QA systems provides **crisp/precise** answers to users natural language questions (NLQ) by translating the (**NLQ**) to precise structured queries that are executed over **KB** to get the answer.

Ex Machina

British theatrical release poster

Directed by	Alex Garland
Produced by	Andrew Macdonald Allon Reich
Written by	Alex Garland
Starring	Domhnall Gleeson Alicia Vikander Oscar Isaac
Music by	Ben Salisbury Geoff Barrow

What else did the director of the movie Ex Machina direct?



NB: The advantage of using KB for question answering is that you can get more precise and accurate answers

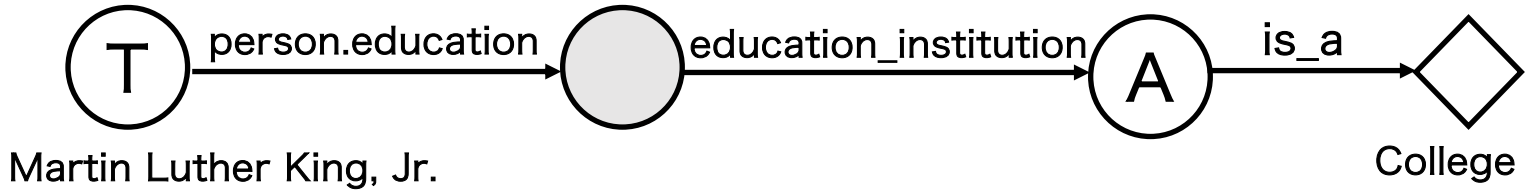
Introduction

- A major challenge in KB-QA is bridging the gap between the natural language expressions and the complex schema of the KB.

Example question: Where did Martine Luther King Junior go to College?

The Corresponding query is constructed by mapping question expressions to query components:

- Topic entity
- Main relation
- Any constraint



- Corresponding query:

```
select ?x where {
    Martin_Luther_King  person.education  ?c .
    ?c   education_institution  ?x .
    ?x   is_a   College .   }
```

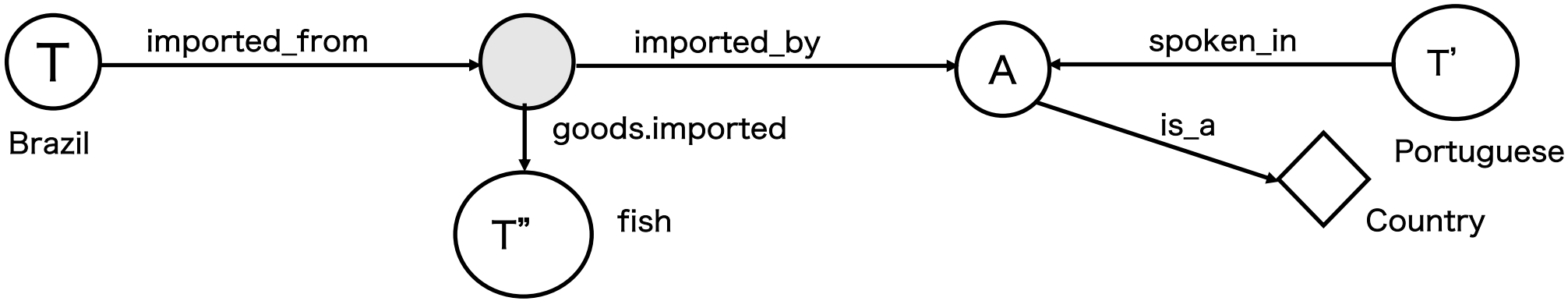
Note: This question can be answered by one main relation path in the KB

Introduction

Example question: Which Portuguese speaking countries import fish from Brazil?

The Corresponding query will have multiple query components:

- **Topic entity:** Brazil and Portuguese
- **Main relation:** import_from-imported_by and spoken_in



- Generating such a query with many expressions mapping to query components is much harder due to the structural complexity of query patterns.

Note: It becomes difficult to answer the Natural Language question because it is complex

Introduction

TEXTRAY: A KB-QA system that answers complex questions using a novel ***decompose-execute-join*** approach.

- It constructs complex query patterns using a set of simple queries (joining partial queries)
- Uses the semantic matching model to learn the semantic similarity between the question and query candidates and best candidate executed on the KB.

Main question: How to decompose the query generation process and learn the semantic matching function?

Introduction

Related work

- **[Fei Li, HV. Jagadish 2014 & Zheng et al 2018]:** Propose annotating the linguistic parse tree of a question with query components and learn to map the tree elements to a query pattern. Mapping can be learned from example questions annotated with complex query patterns.

Limitation: Obtaining complex query patterns can be cumbersome and error prone.

- **[Berant et al 2013 & H. Peng et al 2017]:** Shows that mapping can instead be learned using distant supervision from question answer pairs.

Limitation: This works for simple questions, but challenging for complex questions where only answers to complex questions are available and not the simple queries that constitute the complex query.

TEXTRAY: Proposes that by restricting each simple query to a single relation path and by leveraging some prior domain knowledge, a semantic parser can be trained to answer simple queries using only implicit supervision for simple queries.

Outline

- Introduction
- Preliminaries
- Query Candidate Generation Process
- Semantic Matching
- Implicit Supervision
- Experiment, Results and Discussion
- Conclusion and Future Work

Preliminaries

Goal: To design a KB-QA system that can map a complex NLQ Q to a matching query G, which can be executed against a KB $\mathcal{K}$ to retrieve the answers to Q.

Knowledge Base $\mathcal{K}$: Is the collection of triples of the form (s, r, o), where s: subject, r: relation and o: object.

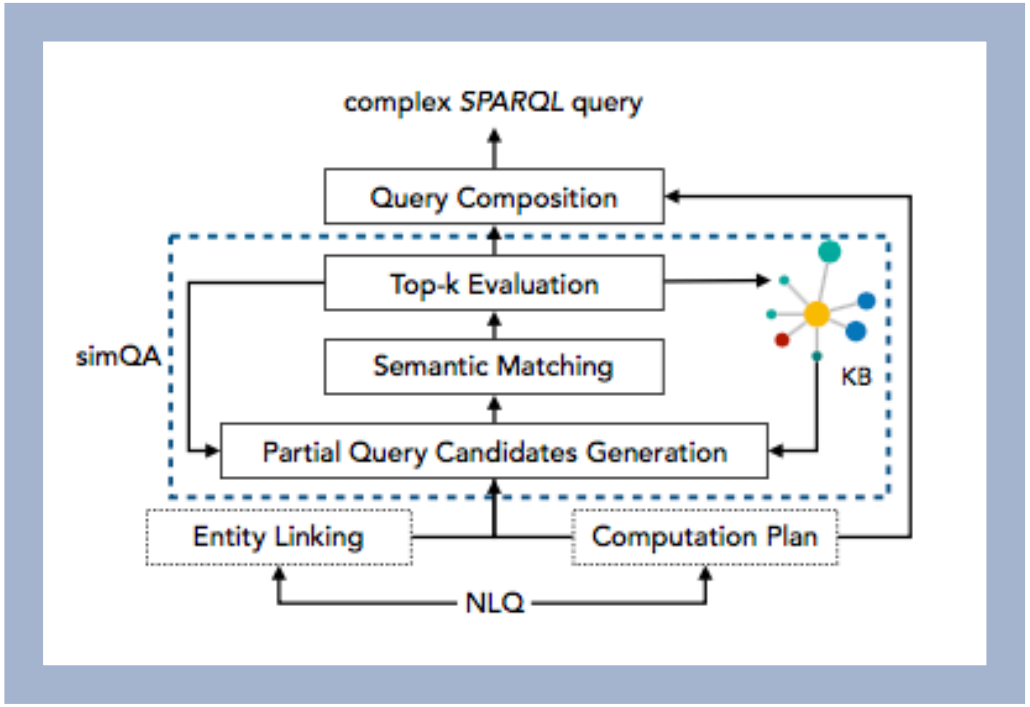
Complex Question Q : Corresponds to G over $\mathcal{K}$ such that G involves joining multiple main relations in $\mathcal{K}$.

ie: $G = (G_1, G_2, \ldots, G_o)$ a sequence of **simple partial queries** connected via different joins.

- Each partial query G_i corresponds to a main relation in G.
- G has a single query focus "?x" corresponding to the answer of Q.

Note: partial queries may share the query focus "?x" .

System architecture



Preliminaries

Computation Plan C : Given Q, the system generates computation plan C which is a tree that decides how G is constructed and executed from partial queries G_i.

It uses two main functions:

- $SimQA$: Denotes search and execution of likely partial query candidates
- $Join$: Denotes the join condition for two partial queries.

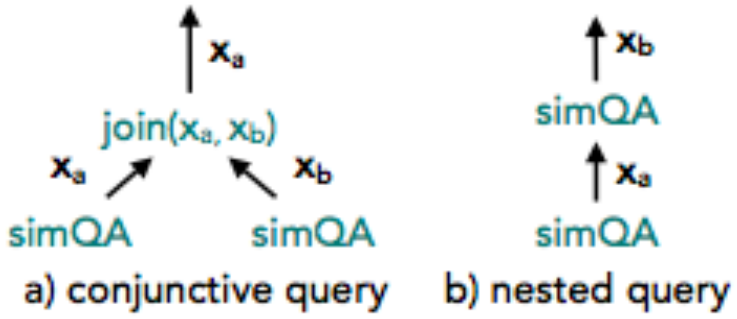


Figure 2: Example computation plan indicating how to construct the complex query given the partial queries

Preliminaries

Query Composition:

Given the complex question Q and its computation plan C, find a sequence of partial queries ($G_1, G_2, \ldots, G_o$) and construct the complex query pattern G such that executing G over $\mathcal{K}$ provides the answer to Q.

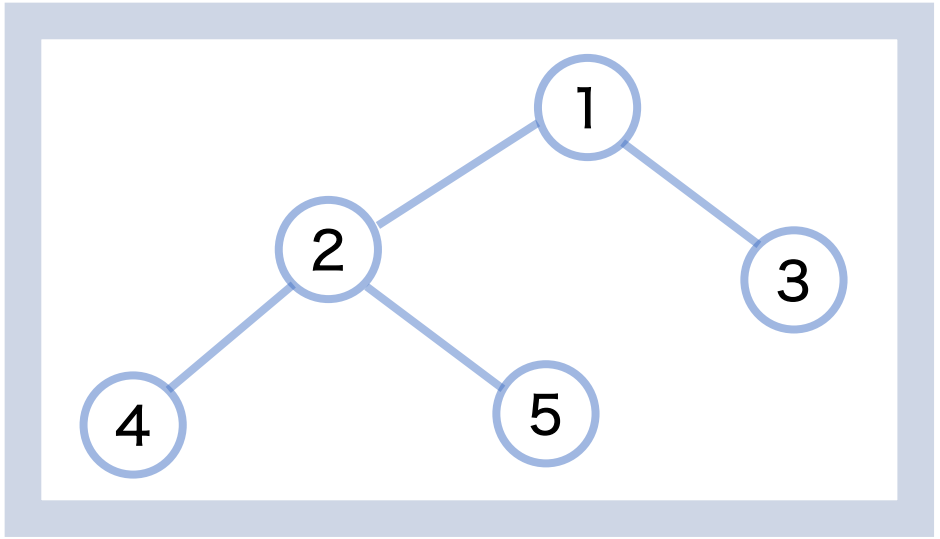
- To find the correct partial query G_i, collect candidates $\{G_i^{(k)}\}_{k=1}^{L}$, score them and reserve best candidates for finding G_{i+1}.

- In order to find candidates, start by identifying entities that are mentioned in Q from $\mathcal{K}$.

[H.Bast et al 2015] entity linking system is applied which returns possibly overlapping pairs of $E = \{(mention, entity)\}$ with confidence score and10 best pairs are considered based on score.

Preliminaries

Sequence of partial queries: Post-order tree traversal of the plan yields a sequence in which the partial queries are executed.



- Post-order traversal is usually from:
Left, Right, Root. (4,5,2,3,1)

From this, a post-order traversal of the plan yields a sequence in which partial queries are executed:

$\boldsymbol{z = z_1, z_2, \ldots, z_n,}$ where $\boldsymbol{z_i \in \{simQA, join\}}$

- [A. Talmor 2018] Augmented pointer networks are applied to predict the likelihood of the computation plan.

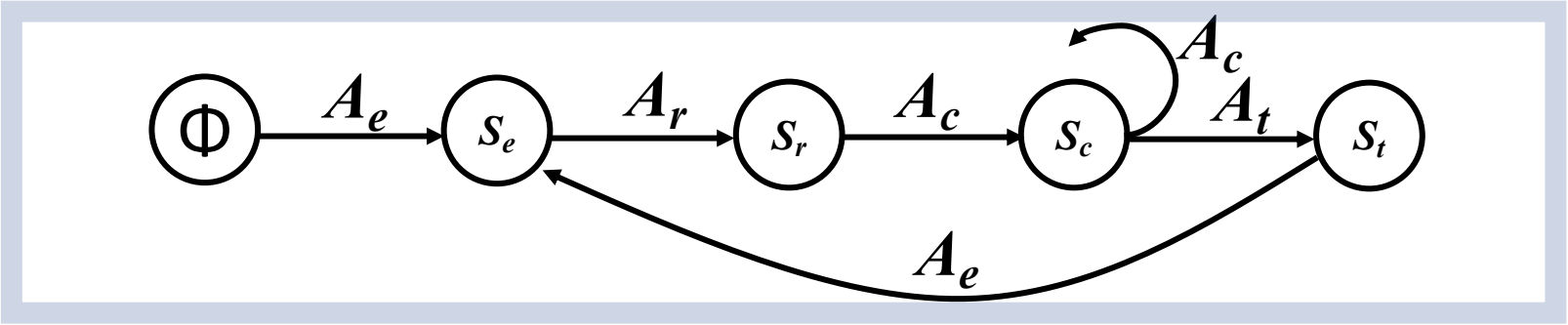
$$\boldsymbol{P(C|Q) = \Pi_{i=1}^{j} P(zi|Q, z_{1,\, i\ \_\ 1})}$$

Outline

- Introduction
- Preliminaries
- Partial Query Candidate Generation Process
- Semantic Matching
- Implicit Supervision
- Experiment, Results and Discussion
- Conclusion and Future Work

Partial Query Candidate Generation Process

The candidate generation process is described by a set of states $S = \{\emptyset, S_e, S_r, S_c, S_t\}$ and actions $A = \{A_e, Ar, Ac, At\}$



$\emptyset$: Empty query, S_e: Single entity, S_r: Main relation, S_c: Constraint, S_t: New state

Note: Action A grows the candidate query by adding one query component at a time.

A_e: Finds candidates for seed entity, A_r: Adds main relation paths to the entity candidates, A_c: Adds any constraints.

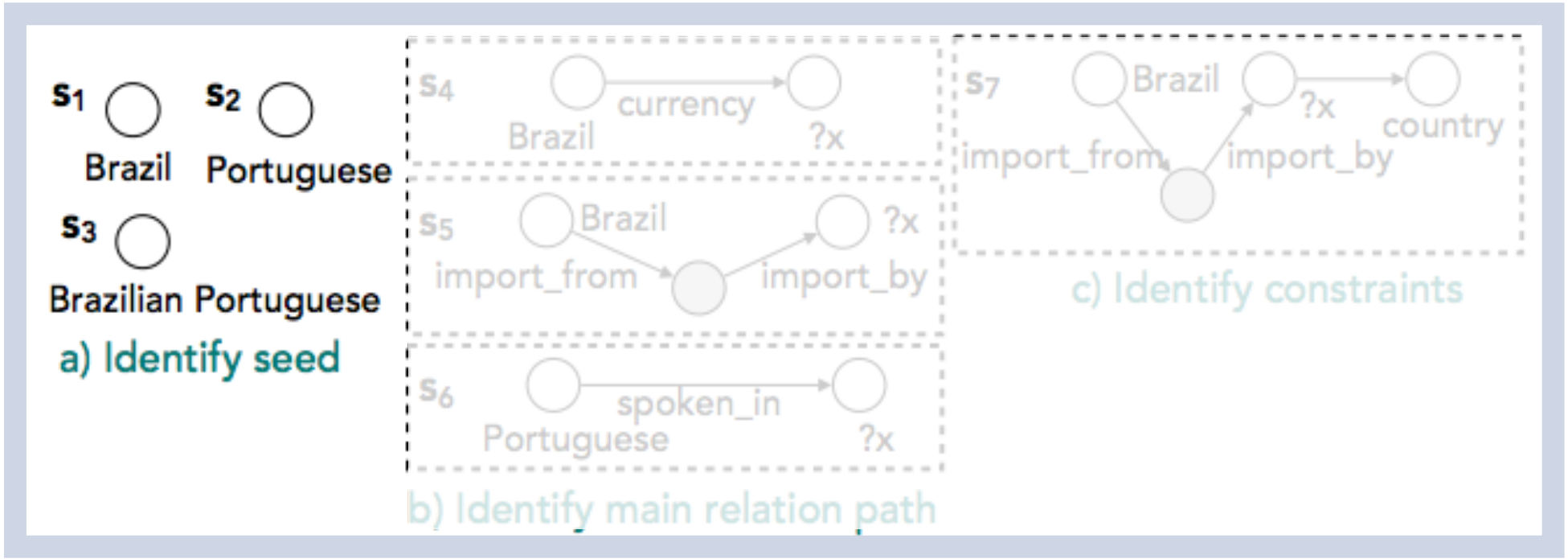
A_t: denotes termination of partial query G_i and transition to new state S_t

Partial Query Candidate Generation Process

A_e: **Finds seed entity candidates** from either entity linking step or an answer of previous partial query G_i.

Which Portuguese speaking countries import fish from Brazil?

S_e: Single entity

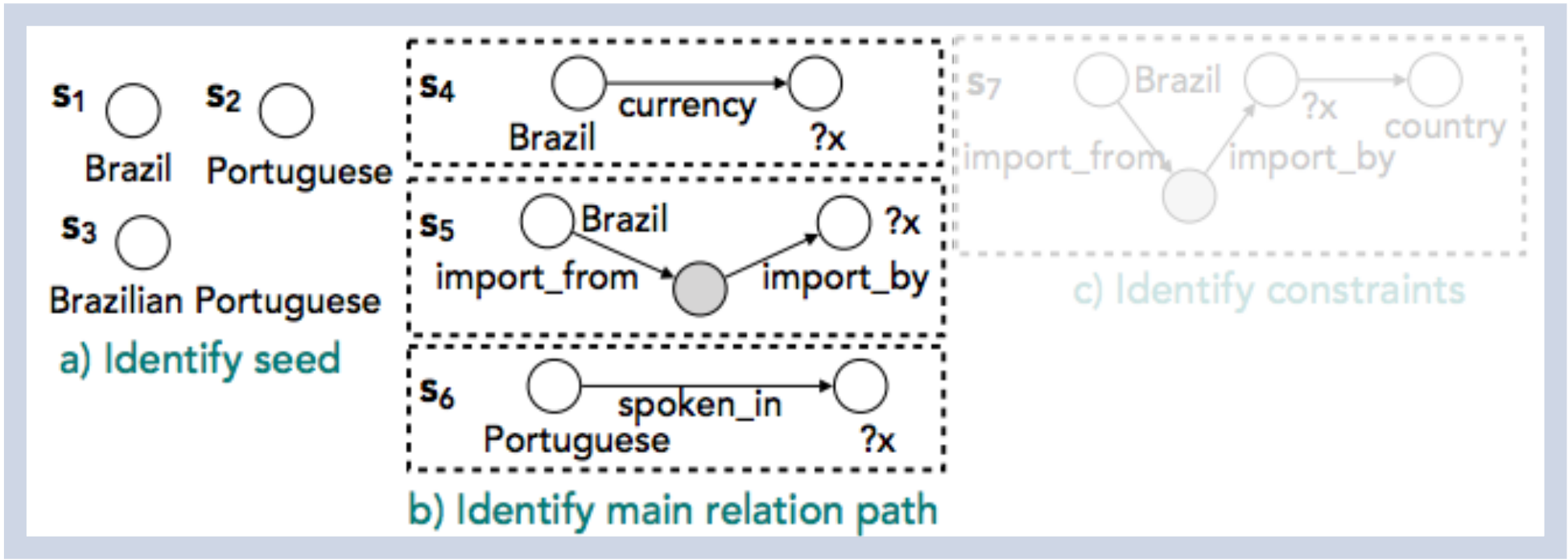


Partial Query Candidate Generation Process

A_r: **Finds different relation paths** in κ that connects seed entities to answers using relation edge or mediated 2-hope relation path.

S_e: Single entity

S_r: Main relation



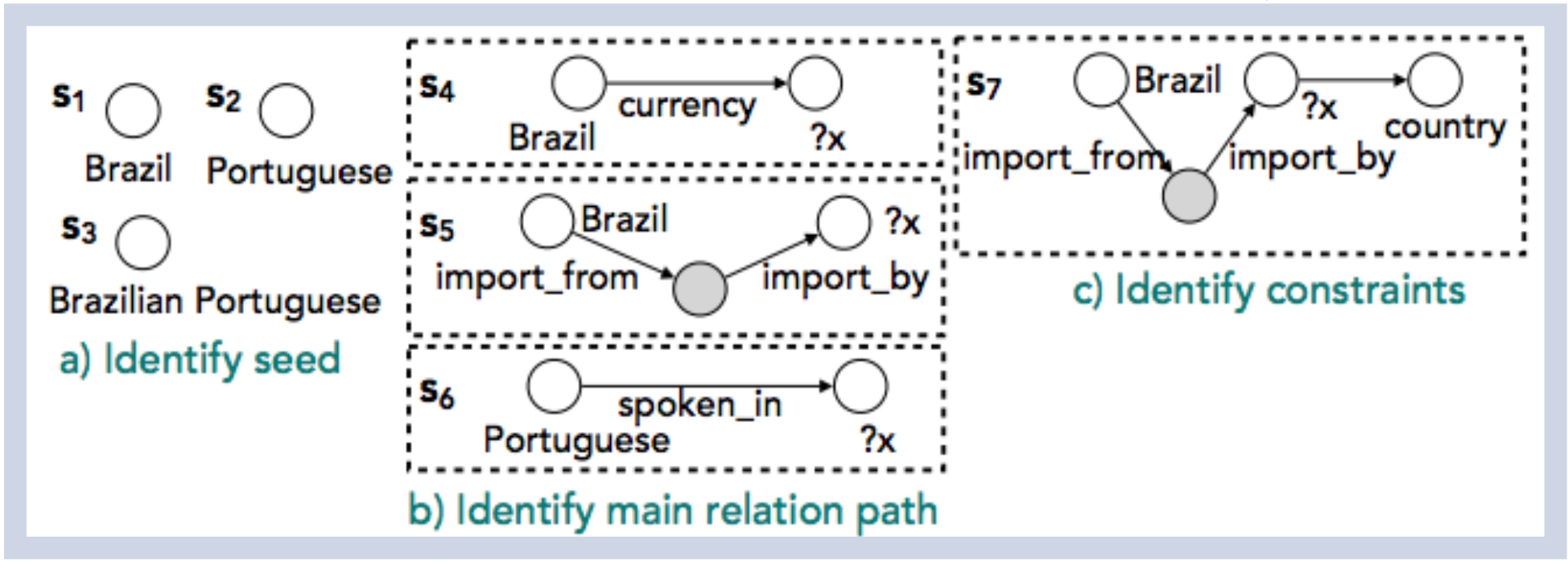
Partial Query Candidate Generation Process

A_c: **Adds any constraints.** A set of constraints $E_c = U\{E, E_t, E_o\}$ is considered:

E :Set of entity links, E_t :Entities connected to the answer via specific relations,

E_o : Named entities of type ***date, time.***

S_e: Single entity S_r: Main relation S_c: Constraint



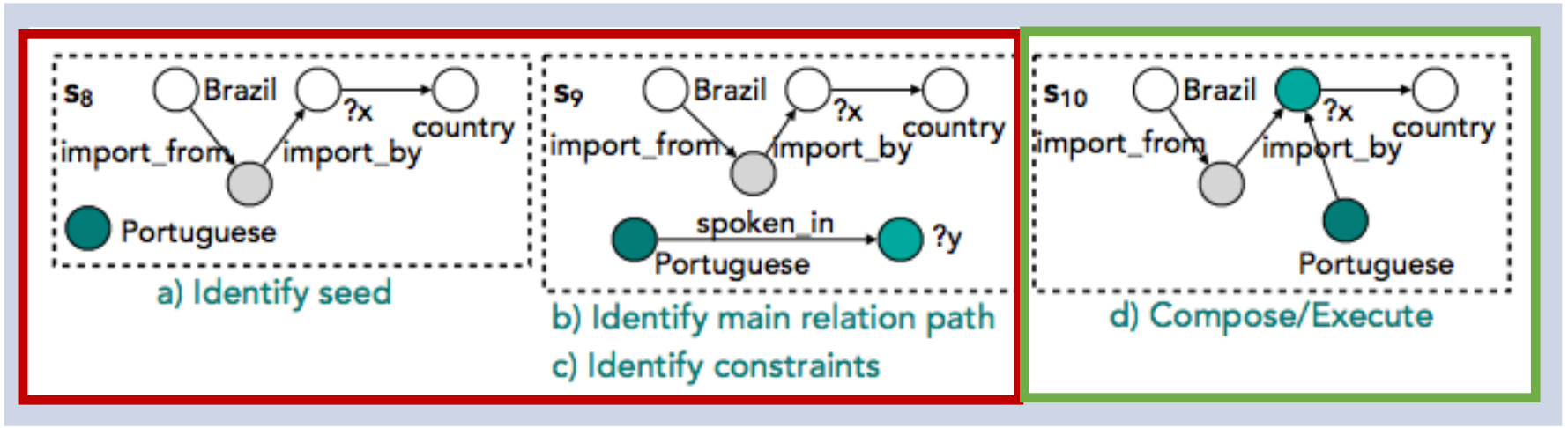
Partial Query Candidate Generation Process

A_t: Terminates partial query G_i and transition to new state S_t. In S_t, the search refers to the computation plan to determine the dependencies for G_{i+1}.

- If next operation is a ***join***: G_{i+1} does not share any entities from the question. A none overlapping entity from E becomes seed entity S_e for G_{i+1}.

- If the next operation is a ***SimQA***: G_{i+1} relies on the answers for G_i. G_i answers becomes seed entities for G_{i+1}. In such a case, the partial query candidates $G_i^{(k)}$ are scored based on their **semantic similarity** to the question Q and *k*-best candidates are kept. translated to **SPARQL** and executed on $\mathcal{K}$.

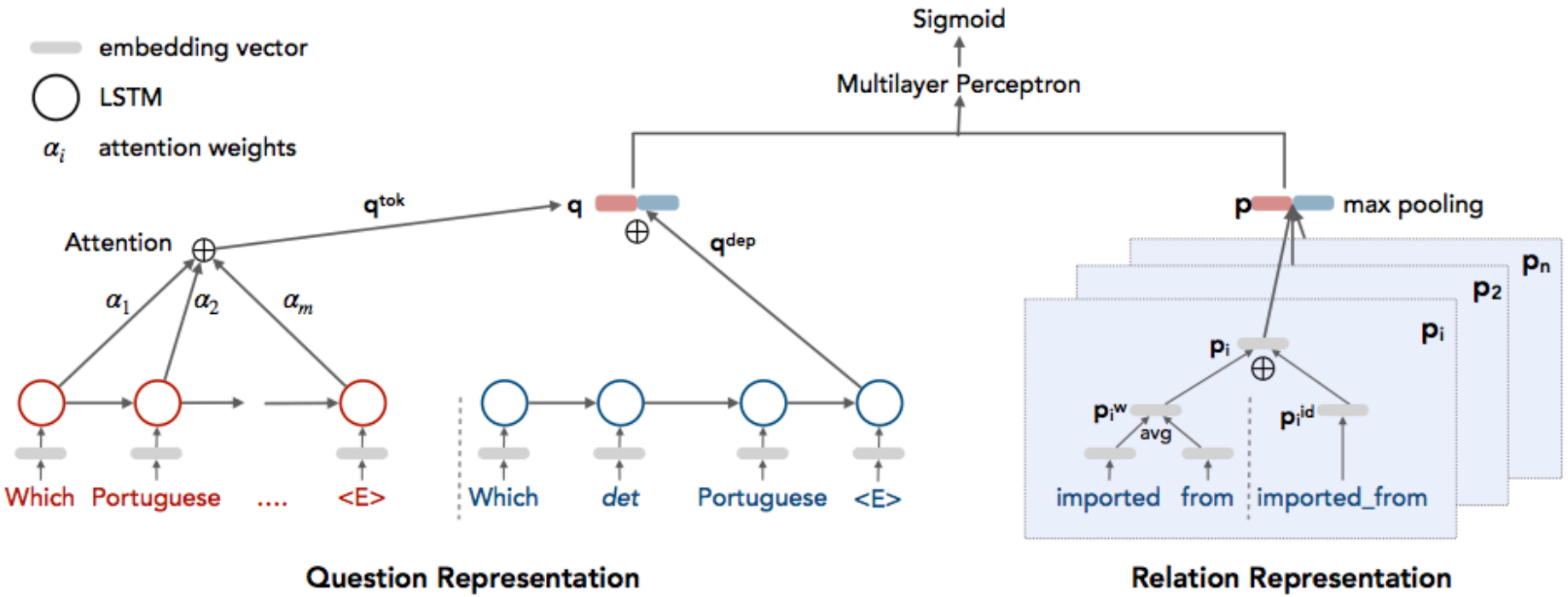
S_t: New state



Outline

- Introduction
- Preliminaries
- Query Candidate Generation Process
- Semantic Matching
- Implicit Supervision
- Experiment, Results and Discussion
- Conclusion and Future Work

Semantic Matching Model



Semantic Matching Model

Question Encoding: question is encoded using its word sequences and the dependency structure.

- Words corresponding to seed entities and constraint entities are replaced by dummy tokens $\boldsymbol{w_E}$ and $\boldsymbol{w_C}$ respectively.

- Question words are mapped to embedding vectors $(\boldsymbol{q_1^w}, \boldsymbol{q_2^w}, \ldots, \boldsymbol{q_n^w})$ by matrix $\boldsymbol{E_w}$. The last hidden state of the LSTM is taken to be the latent vector representation $\boldsymbol{q^w}$.

- Dependency encoding is done by considering the dependency tree as a sequence of question words and their dependency labels. The same embedding matrix $\boldsymbol{E_w}$ is used to map to vectors and the last hidden state of another LSTM is taken to be the latent vector $\boldsymbol{q^{dep}}$.

Semantic Matching Model

Partial query Encoding: Various relations in the query are used to encode the partial query.

- For each relation $\boldsymbol{P_i}$ in the partial query, the sequence of both **relation words** and **relation id** is considered.

For example: a relation **import_by,**
The **id** sequence is **{imported_by},** and **word sequence** is **{'imported', 'by'}**

- Word sequences are converted to a sequence of embeddings using $\boldsymbol{E_w}$ and averaged $(\boldsymbol{P^w})$ average word embeddings.
- At **id** level, the relation is translated directly using another embedding matrix of relations $\boldsymbol{E_p}$.
  The semantic vector of $\boldsymbol{P_i}$ becomes $\boldsymbol{P} = [\boldsymbol{P^w}; \boldsymbol{P^{id}}]$.

Max pooling is applied over hidden vectors of the relation to obtain compositional semantic representation of the partial query.

K. Luo et al 2018, Knowledge Base Question Answering via Encoding of Complex Query Graphs. EMNLP

Semantic Matching Model

Attention mechanism: Complex questions are very long with expressions matching multiple partial queries in the knowledge base. Irrelevant information can distract the matching.

Attention is used to improve the quality of question representation. **The partial query vector $\boldsymbol{P}$ is used as the attention vector to highlight the relevant parts of the question word vector $\boldsymbol{q^w}$.**

Given all hidden vectors $\boldsymbol{h_t}$ at time step $\boldsymbol{t} \in \{\boldsymbol{1}, \boldsymbol{2}, \ldots, \boldsymbol{n}\}$, the context vector $\boldsymbol{c}$ is the weighted sum of all hidden states:

$$\mathbf{c} = \sum_{t=1}^{n} \boldsymbol{\alpha_t h_t}$$

Where $\boldsymbol{\alpha_t}$ is the attention weight computed as: $\boldsymbol{\alpha = softmax(Wtanh(W_q q^w + W_P P))}$
where $\boldsymbol{W, W_P, W_q}$ are learnable network parameters.

K. Herman et al 2015, Teaching Machines to Read and Comprehend. NIPS
T. Luong et al 2015, Effective Approaches to Attention-based Neural Machine Translation. EMNLP

Semantic Matching Model

Objective function: The context vector $\boldsymbol{c}$, the question dependency vector $\boldsymbol{q^{dep}}$, and query vector $\boldsymbol{P}$ are concatenated and fed to a multi-layer perceptron (MLP), which outputs a semantic similarity score $\boldsymbol{S_{sem}}\ (\boldsymbol{q}, \boldsymbol{Gi})$.

The semantic matching model adopts a cross entropy loss function during the training.

$$\boldsymbol{loss = ylog\left(S_{sem}(q, G_i)\right) + (1 - y)\log(1 - S_{sem}(q, G_i))}$$

Where $\boldsymbol{y} \in \{\boldsymbol{0}, \boldsymbol{1}\}$ is the label indicating whether $\boldsymbol{G_i}$ is correct or not.

Outline

- Introduction
- Preliminaries
- Query Candidate Generation Process
- Semantic Matching
- Implicit Supervision
- Experiment, Results and Discussion
- Conclusion and Future Work

Implicit Supervision

Generate training data from question answer pairs

Main challenge: No ground truth for partial query candidates. the quality of the partial query candidate $G_i^{(k)}$ has to be indirectly measured by computing the F1 score of the derived answers compared to the labeled answers.

The quality of the partial query can be estimated based on the quality of derivations as.

$$V(G_i^{(k)}) = \max_{i \le t \le n-1} R(D_{t+1}^{(k)})$$

Where: D_t is the full query derivation at level t in the computation plan,

n is the number of partial queries,

R is the F1 score of the answers of the derivation

Implicit Supervision

Example: consider two candidate partial queries:

$G_1^{(1)}$ = ?a event.champion ?c. ?c sports_event 1946_World_Series.

$G_1^{(2)}$ = 1946_World_Series sports_event.umpire ?a .

Considering derivations starting from candidates, their retrieved answers and the ground truth answer "Bush Stadium"

$D_2^{(1)}$ = ? a event.champion ?c. ?c sports_event 1946_World_Series.
? a arena_stadium ?x.

$A\left(D_2^{(1)}\right)$: Bush Stadium

$D_2^{(2)}$ = 1946_World_Series sports_event.umpire ?a .
? a place_of_birth ?x.

$A\left(D_2^{(2)}\right)$: Texas, Missouri, Illinois, New Jersey

Implicit Supervision

The F1 scores and values are:

$$\boldsymbol{R}\left(\boldsymbol{D}_{\boldsymbol{2}}^{(\boldsymbol{1})}\right) = 1.0,\ \boldsymbol{V}(\boldsymbol{G}_{\boldsymbol{1}}^{(\boldsymbol{1})}) = 1.0,\ \boldsymbol{V}(\boldsymbol{G}_{\boldsymbol{2}}^{(\boldsymbol{1})}) = 1.0$$

$$\boldsymbol{R}\left(\boldsymbol{D}_{\boldsymbol{2}}^{(\boldsymbol{2})}\right) = 0.0,\ \boldsymbol{V}(\boldsymbol{G}_{\boldsymbol{1}}^{(\boldsymbol{2})}) = 0.0,\ \boldsymbol{V}(\boldsymbol{G}_{\boldsymbol{2}}^{(\boldsymbol{2})}) = 0.0$$

The scores of $\boldsymbol{G}_{\boldsymbol{i}}^{(\boldsymbol{k})}$ are used to approximate $\boldsymbol{y}^{(\boldsymbol{k})} \in \{\boldsymbol{0}, \boldsymbol{1}\}$ for the candidate.

Addressing Spurious Matches

The implicit supervision described is susceptible to spurious partial queries which happen to find correct answers but do not capture the semantic meaning of the question.

This can greatly affect the training data quality and the performance of the semantic matching model.

- Incorporating some domain knowledge as priors for scoring would alleviate the problem.

Addressing Spurious Matches

Qualitatively examining positive examples, found that correct candidates tend to use the same words as the natural language question.

Example question: "which government jurisdiction held the LIX Legislature of the Mexican congress in 2011?"

The partial query with main relation:

government_jurisdiction.governing_officials

Words **government** and **jurisdiction** are shared.

Thus, surface-form similarity of the main relation and the question can be used to promote true positive and false negative examples.

Addressing Spurious Matches

1. Compute the ratio of number of words in the main relation of $\boldsymbol{G_i^{(k)}}$ that are mentioned in the question $\boldsymbol{Q}$:

$$lexical\_score\ (\boldsymbol{Q}, \boldsymbol{G_i^{(k)}})$$

2. **Co-occurrence:** Use small-hand crafted lexicon that contains lexical pairs $(\boldsymbol{w_G}, \boldsymbol{w_Q})$ of words from the relation $\boldsymbol{w_G}$ and keywords from the question $\boldsymbol{w_Q}$ based on their co-occurrence frequency.

$$co\_occur\_score\ (\boldsymbol{Q}, \boldsymbol{G_i^{(k)}})$$

The quality of a partial query candidate $\boldsymbol{G_i^{(k)}}$ is given by:

$$\boldsymbol{VG_i^{(k)}} + \boldsymbol{\gamma lexical\_score}\left(\boldsymbol{Q}, \boldsymbol{G_i^{(k)}}\right) + \boldsymbol{\delta co\_occur\_score}\ (\boldsymbol{Q}, \boldsymbol{G_i^{(k)}})$$

Where $\boldsymbol{\gamma}\ and\ \boldsymbol{\delta}$ are hyperparameters denoting the strength of priors

Outline

- Introduction
- Preliminaries
- Query Candidate Generation Process
- Semantic Matching
- Implicit Supervision
- Addressing Spurious Matches
- Experiment, Results and Discussion
- Conclusion and Future Work

Experiments

Item	Datasets	Samples	Train	Dev	Test	Content
KB-QA Benchmark	**CompQWeb**	34, 689	27,734	3,480	3,475	Complex question, answer & SPARQL query
	WebQSP	4,737	3,098		1,639	Mostly simple questions, answer, & SPARQL query
Knwledge base	**Freebase**					
Evaluation Metric	Average F1 score for the predicted answer (KB-QA system effectiveness) Precision @1 as the fraction of questions that were answered with exact gold answer					
Comparision	**CompQA:** Generates full query candidates for complex questions **Parasempre:** Parses questions to logical forms that are executed agains Freebase **SplitQA & MHQA:** Handle complex questions but rely on noisy textual data sources. **STAGG:** Uses a similar approach but improves results by feature engineering & augment entity linking with external knowledge. **MulCQA** **AQQU**					

Results

Effectiveness on Complex Questions

Method	Average F_1	Precision@1
TextRay	**33.87**	**40.83**
CompQA [18]	4.83	4.83
Parasempre [4]	7.05	12.37
SplitQA (web) [24]	-	27.50
MHQA (web) [23]	-	30.10

Table 1: Average F_1 scores and Precision@1 on CompQWeb.

Effectiveness on Simple Questions

Method	Average F_1	Precision@1
TextRay	60.3	**72.16**
CompQA [18]	59.5	61.64
Parasempre [4]	46.9	51.5
STAGG [30]	**66.8**	67.3
MulCQA [2]	52.4*	-
AQQU [3]	49.4*	-

Table 2: Average F_1 scores and Precision@1 on WebQSP dataset. * are results on the WebQ dataset.

Effectiveness of Candidate Generation

Method	CompQWeb F_1^**	WebQSP F_1^*
TextRay	50.83	84.57
CompQA [18]	31.30	75.03
Parasempre [4]	19.15	60.95

Table 3: Upper bound F_1 scores for candidate generation.

	CompQWeb		WebQSP	
	%	Avg. *best* F_1	%	Avg. *best* F_1
Top-1	48.7	33.87	66.5	60.31
Top-2	55.6	36.05	79.4	69.73
Top-5	65.6	39.97	89.1	76.5
Top-10	71.5	42.42	93.7	79.9

Table 4: Percentage of questions with the highest F_1 score in the top-k candidate derivations, and the average best F_1.

Ablation Study

Investigate the contributions of various components in complex question answering.

The components include:
- Constraints
- Attention mechanism
- Prior knowledge

Setup	CompQWeb F_1	WebQSP F_1
TEXTRAY (Full System)	**33.87**	**60.31**
No constraints	28.16	58.39
No attention	29.92	58.31
No priors	31.28	59.43

Table 5: Component-wise ablation results (Average F_1).

Note: In table 5, the improvement due to prior knowledge is 2.6% on CompQWeb. While this may seem marginal, It was observed that Improvements were significant over top-k (k >1) results.
- Best F1 for top-10 was 29.47 when no prior knowledge incorporated, and it was 40.06 with prior knowledge.

Error Analysis

50 randomly sampled queries with F1 scores less than 0.1 were analyzed.

- About 36% of errors were due to incorrect entity identification. 88% out of these were made when finding the first partial query. Errors made early in the process propagate and affect future partial queries.

- 45% of errors were because a wrong or ambiguous relation was scored high by the semantic matching model.

- 19% of errors were made when constraints or value nodes were missed.

Outline

- Introduction
- Preliminaries
- Query Candidate Generation Process
- Semantic Matching
- Implicit Supervision
- Addressing Spurious Matches
- Experiment, Results and Discussion
- Conclusion and Future Work

Future work

There are two directions in which the work can be extended:

- While it is beneficial to resolve multiple query components simultaneously, the inference could be improved if the question representation reflected all prior instances.

- More operations such as value comparisons and aggregations could be included in the computation plan and incorporated in the semantic matching model.

Conclusion

We have presented TEXTRAY, a new KB-QA system that answers complex questions over a knowledge base by constructing complex queries from simpler partial queries.

It integrates the novel query candidate generation strategy and a semantic matching model learned from implicit supervision to find and join partial queries efficiently.

Appendix

What is the capital of India?

Dependency tree structure.

- Capital-NN (root)
- What-WP (nsubj)
- is-VBZ (cop)
- The-DT (det)
- of-IN (prep)
- India-NNP (pobj)

Implicit Supervision

- Intuitively, candidates with scores greater than the threshold can be considered as positive examples for the question and negative otherwise.

But using a fixed threshold might lead to many false negatives for questions where no candidate has a score greater than threshold.