Question Answering Systems

Neural KG-QA systems

Rishiraj Saha Roy
Max Planck Institute for Informatics, Germany

7

Saarland University, Summer Semester 2020

Quick word(s) on reviewing

1
- Do not pick disadvantages from future work

2
- Judge time and scope of paper

course chronological X

  - No complex, no feedback, no paraphrases are not necessarily valid points

3
- Take a stand: do not pose same thing as advantage or disadvantage

"semantic" ☺

4
A1) x
A2
x (A3
D1 D2 D3
- Write mutually exclusive points based on understanding of paper

5
- Read papers beforehand, attend class, clarify questions during class

6
- Assignment all 4's does not directly mean a grade of 4.0; don't worry ☺

Question of the day

How do we design KG-QA systems with neural learning?

You'll find this covered in

- ① Semantic Parsing via Staged Query Graph Generation: Question Answering with Knowledge Base
  - Yih et al.
  - ACL 2015

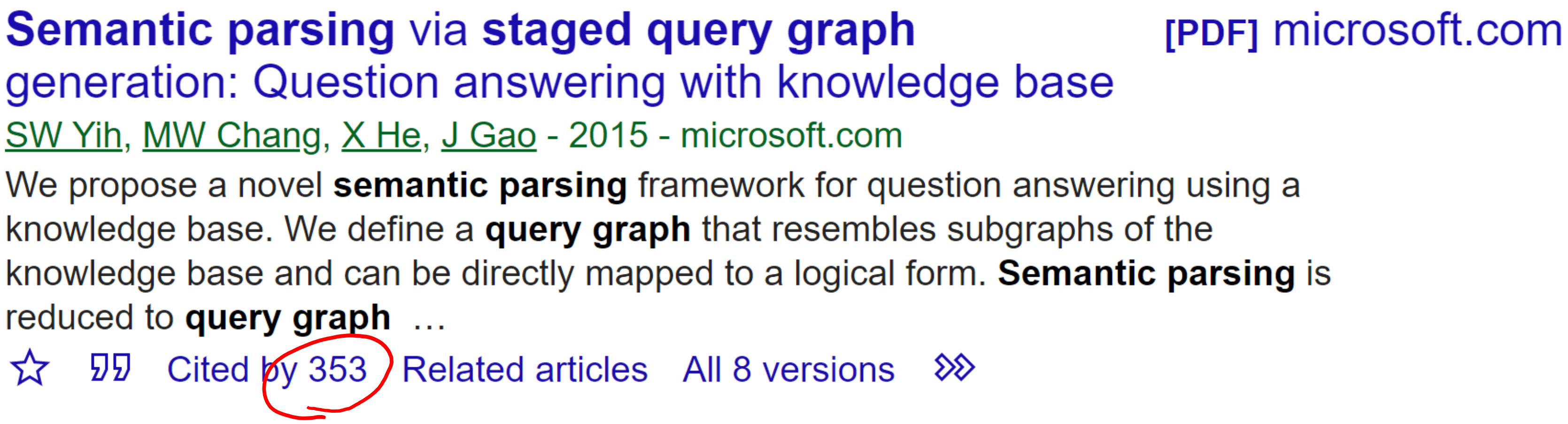
    MSR
  - → https://www.aclweb.org/anthology/P15-1128.pdf
- ② Knowledge Graph Embedding Based Question Answering
  - Huang et al.
  - WSDM 2019

    Baidu Research
  - http://research.baidu.com/Public/uploads/5c1c9a58317b3.pdf

Research paper 1

Semantic Parsing via Staged Query Graph Generation: Question Answering with Knowledge Base

STAGG



Semantic parsing via **staged query graph** generation: Question answering with knowledge base

[PDF] microsoft.com

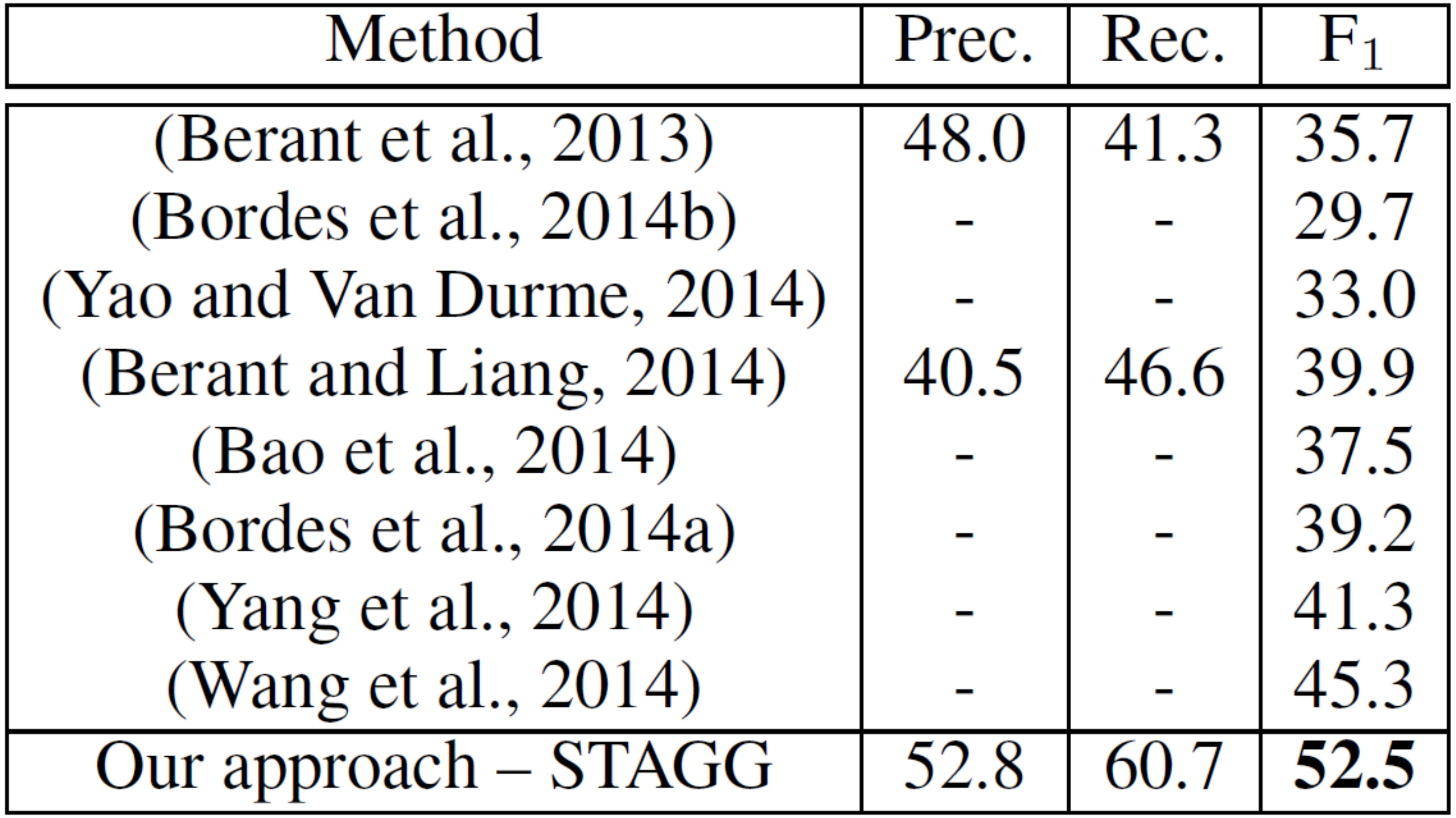
SW Yih, MW Chang, X He, J Gao - 2015 - microsoft.com

We propose a novel **semantic parsing** framework for question answering using a knowledge base. We define a **query graph** that resembles subgraphs of the knowledge base and can be directly mapped to a logical form. **Semantic parsing** is reduced to **query graph** …

☆ ❞ Cited by 353 Related articles All 8 versions »

Best performance on WebQuestions*

In paper
Leaderboard



Method	Prec.	Rec.	F_1
(Berant et al., 2013)	48.0	41.3	35.7
(Bordes et al., 2014b)	-	-	29.7
(Yao and Van Durme, 2014)	-	-	33.0
(Berant and Liang, 2014)	40.5	46.6	39.9
(Bao et al., 2014)	-	-	37.5
(Bordes et al., 2014a)	-	-	39.2
(Yang et al., 2014)	-	-	41.3
(Wang et al., 2014)	-	-	45.3
Our approach – STAGG	52.8	60.7	**52.5**

w/o add'l. evidence

~ Jul-Aug 2015

only KG

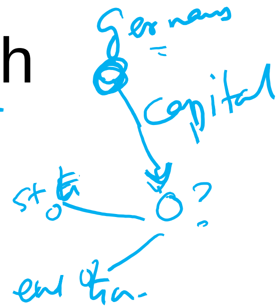
Sarthak Jain

NAACL

* Until https://www.aclweb.org/anthology/N16-2016.pdf

The STAGG System: What's new?

- First notable system to use graph representations of KBs for QA KG
- Previously restricted to RDF triples and SPARQL querying (logical form)
- Equivalence of subgraph search and logical form
- Introduces neural learning



graph

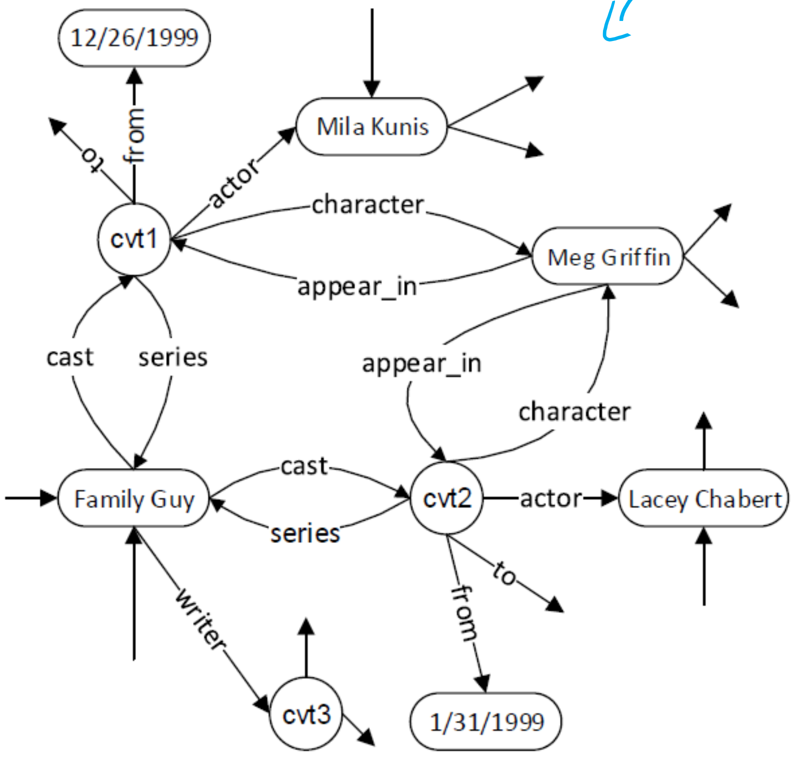


Figure 1: Freebase subgraph of Family Guy

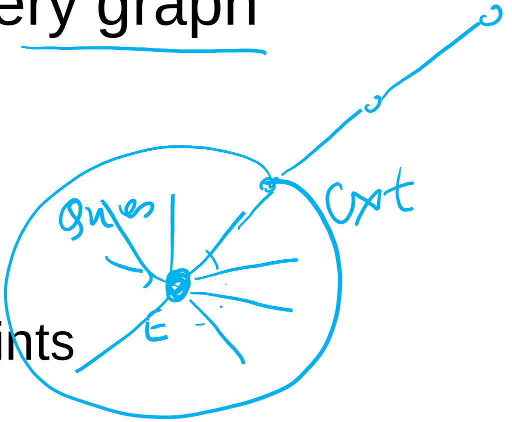
The STAGG System: What's new?

- Defines notion of context subgraph

Question

cxt in KG (local subgraph)

- No need to search over whole KG
- Subgraph search posed as staged procedure of growing query graph
    - Find topic entity
    - Find predicate
    - Consider additional constraints



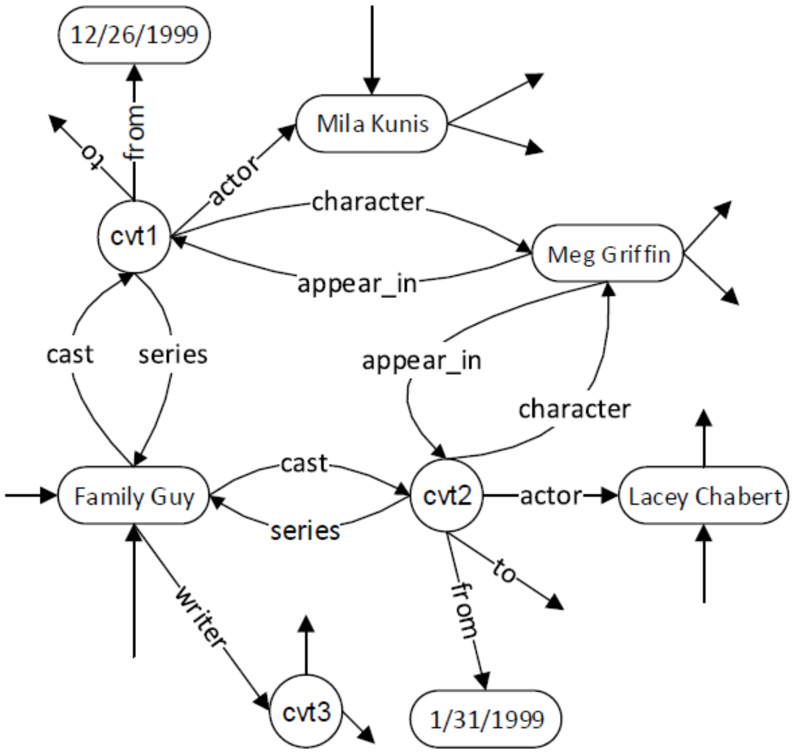


Figure 1: Freebase subgraph of Family Guy

Why focus on a context graph?

A* p* E(topic)

- e.g → *Who first voiced Meg on Family Guy?*

  we don't need to!!

- ① Focus on promising area of KG space

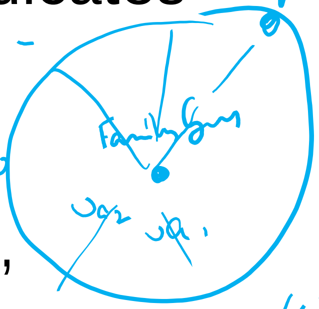
  o g E

  o Δ E

- ② Resolving other entities and predicates also becomes easier

  Meg

  meg'₂



- ③ Two sub-problems: Entity linking, QA

  2 Relation matching

  → go neural!

  ↳ classification / ranking / sim (prediction)

  → not trivial for QA

- ④ What motivates thinking on these lines?

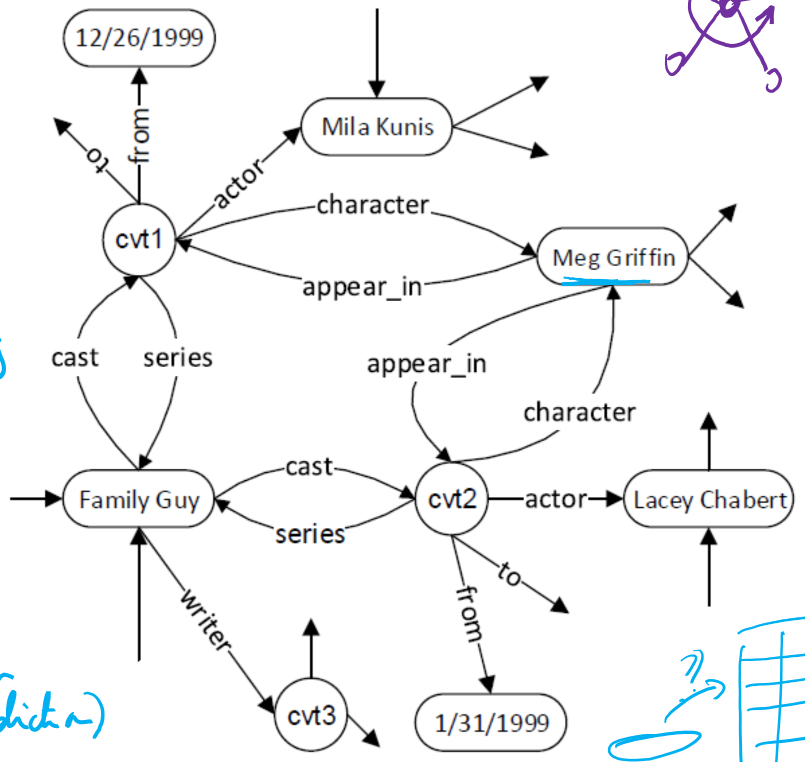


Figure 1: Freebase subgraph of Family Guy

Graph model

- 1 Entities, classes, literals, CVTs as nodes
- 2 Predicates as edge labels
- Directed edges between nodes
- No need to differentiate between entities and literals (and CVTs)
- STAGG works over Freebase
  - 46M topics, 2.6B facts

→EUC

cxt

efficiency

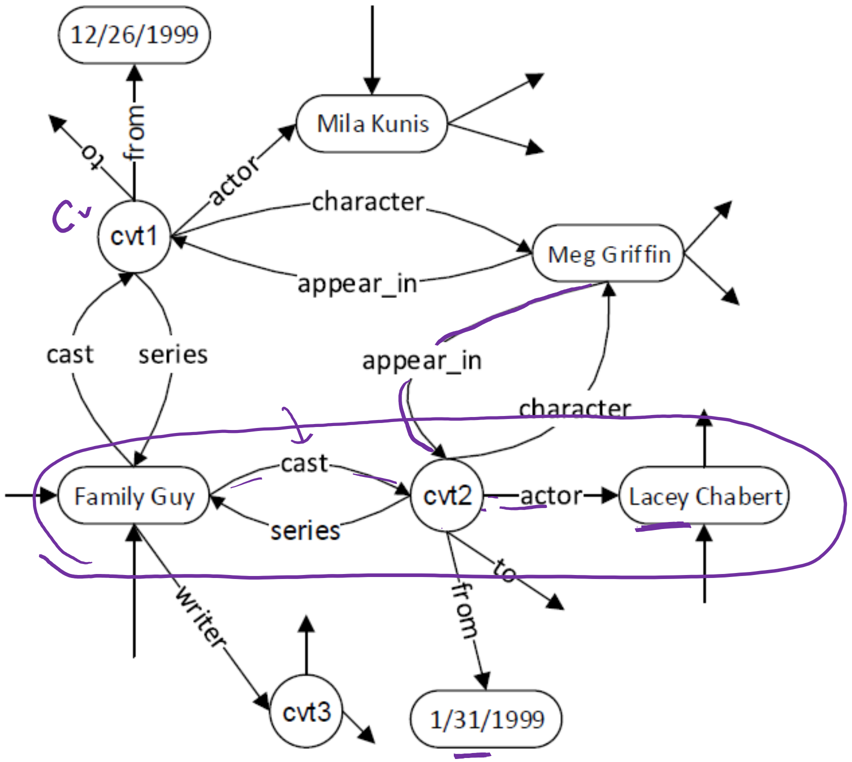


Figure 1: Freebase subgraph of Family Guy

cxt
+

Query subgraph

graphical query

1 - Grounded entity (linked)
2 - Existential variable
3 - Lambda variable
4 - Aggregation function (+ constraints)
- Answer maps to lambda variables (Berant et al. 2013)

can be executed over KGs

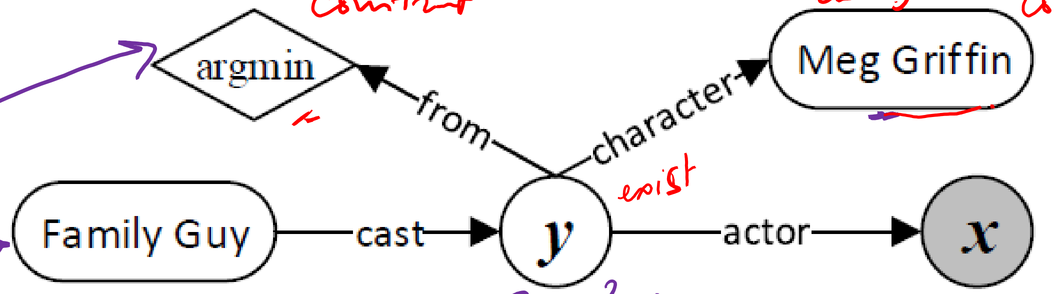


Figure 2: Query graph that represents the question "Who first voiced Meg on Family Guy?"

count

λ-calculus → | λ-DCS
simplify

graph query ≡ logical form ≡ SPARQL query

$\lambda x . \exists y . cast(FamilyGuy, y) \wedge actor(y, x) \wedge char(y, MegG)$

"first" → ??

SELECT ?x WHERE
{FG cast ?y. ?y actor ?x.
?y char MG. ?y from ?z} ORDER BY ?z LIMIT 1

∴ graph search → flexible

STAGG

Staged query graph generation

- Build a tree graph
- [illegible]
- Root = topic entity
- x
- One lambda variable = answer
- ~ tree
- One directed path from root to answer: Core inferential chain
- derivation
- ~~Possibly~~ multiple existential variables in between ?y

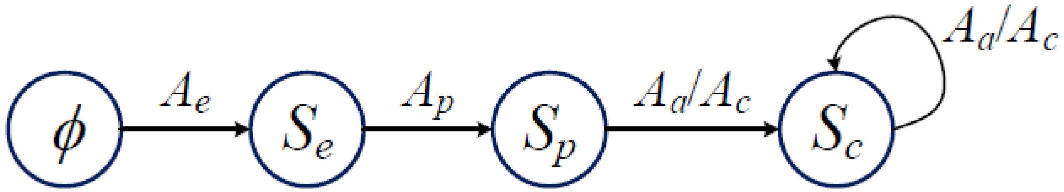


Figure 3: The legitimate actions to *grow* a query graph. See text for detail.

Staged query graph generation

→ grow

- nodes
  All are variables on core chain
  inf...
  except root (grounded)
- Meg
  Additional constraints: Entity or aggregation nodes can be attached to each variable (can also be ans) x
- Grow graph with actions
  - ① Pick entity, ② pick predicate (core chain), ③ pick condition

state of graph

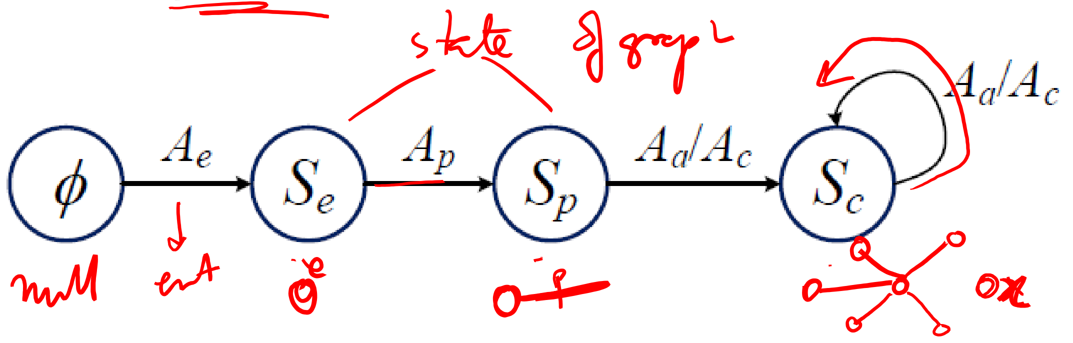
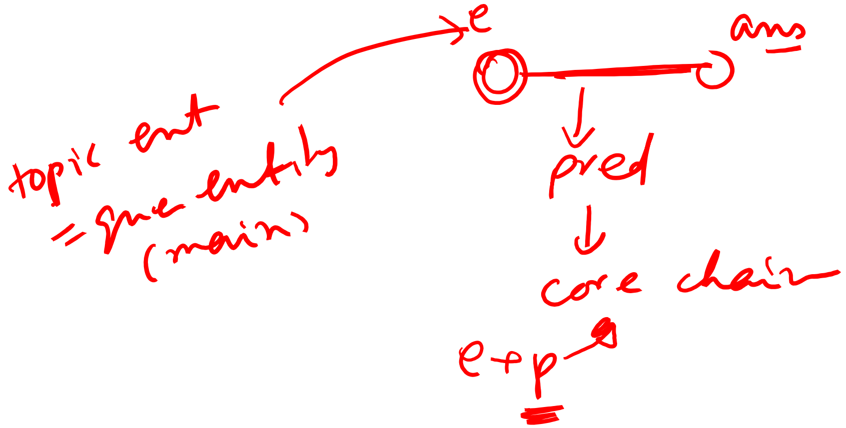


Figure 3: The legitimate actions to *grow* a query graph. See text for detail.



Why this growth order?

- Entity, predicate, conditions
- Efficiency considerations
- Implicit in graph-based QA: To the extent of being “natural”
- What happens if you choose predicates or classes first?

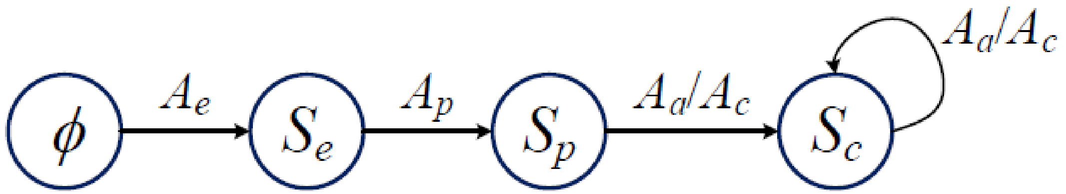
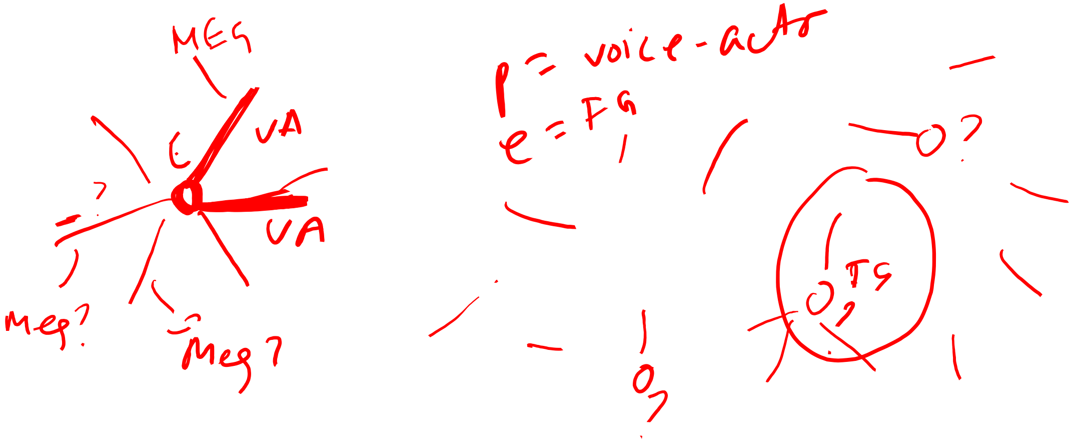


Figure 3: The legitimate actions to *grow* a query graph. See text for detail.



Linking topic entity

NERD

WebQ + SMART anno.

Yang & Chang

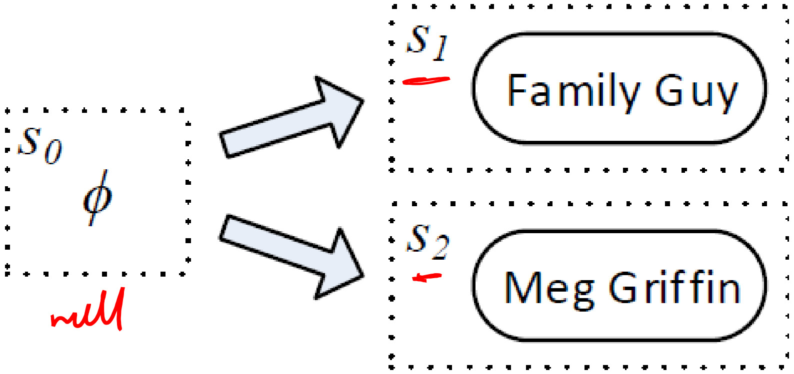
preprocessing

STAGG contrib

- S-MART: NED system for short and noisy texts (Tweets)
- Standard lexicon-based scoring with Wikipedia metadata

inside

- To account for NED mistakes, retains up to top-10 NED!



null

Figure 4: Two possible topic entity linking actions applied to an empty graph, for question "Who first voiced [Meg] on [Family Guy]?"

15:15

Identifying core inferential chain

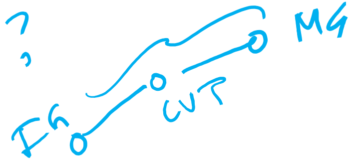
→ predicate candidates

cxt

- Only need to explore legitimate predicate sequences (paths) that start from linked entity!
- All paths of length 2 if CVT in between, length 1 otherwise

→ qualifiers WD

dist = 1?



paths / inf. chains

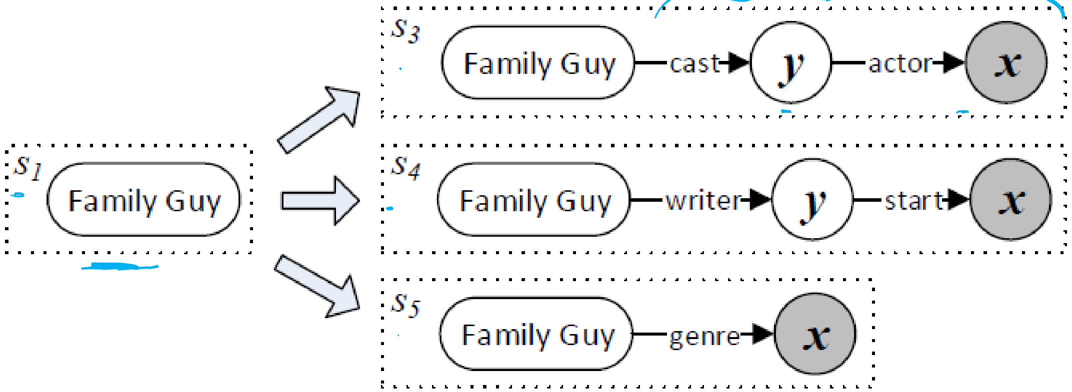


Figure 5: Candidate core inferential chains start from the entity `FamilyGuy`.

Identifying core inferential chain

- Problem boils down to matching question with path similarity
- Similarity using predictors based on neural networks

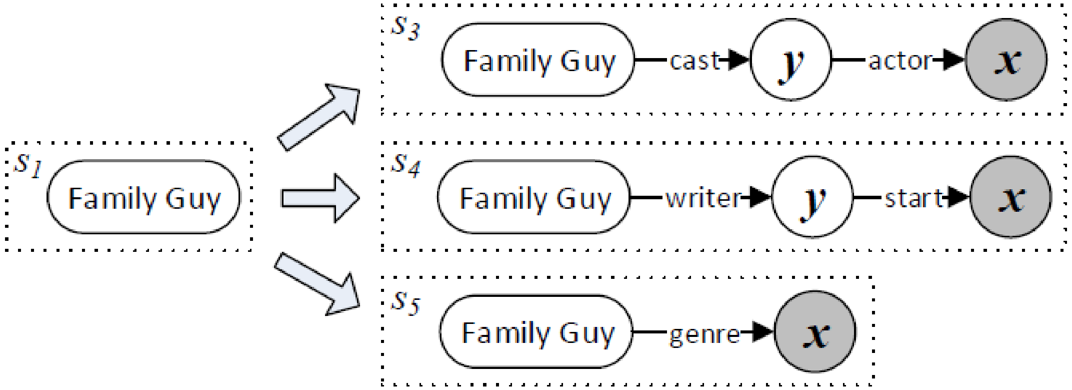


Figure 5: Candidate core inferential chains start from the entity `FamilyGuy`.

CNN

Deep convolutional neural networks

7I layer

- Map question to pattern by replacing entity with <e>

ques

wildcard tag

- Two neural networks
  - 1 Question pattern
  - 2 Inferential chain (path)

qp

ic

2 netwks

same parameter

- Siamese network architecture
- Inspired by mismatch in question and KG vocabulary , lots of paraphrases

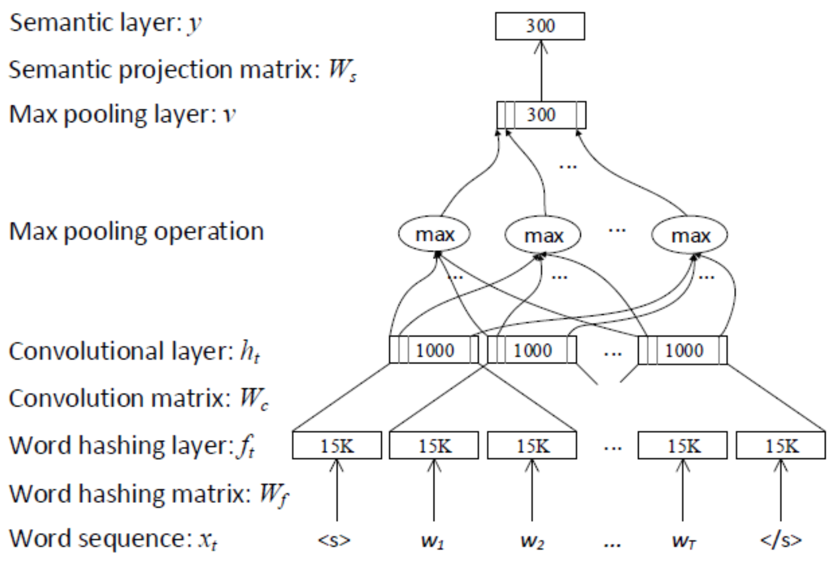


Figure 6: The architecture of the convolutional neural networks (CNN) used in this work. The CNN model maps a variable-length word sequence (e.g., a pattern or predicate sequence) to a low-dimensional vector in a latent semantic space. See text for the description of each layer.

Deep convolutional neural networks

Masterstroke

- Word hashing technique (Huang et al. 2013)
- Break word into character trigrams
- Who → #-w-h, w-h-o, h-o-#

brings

bringings

# #

- Why?
  - Reduces high-dimensionality of input space (think! #words more or #trigrams more?)

gentleness

prev : word space

now : cha trigram

1

  - Subword semantics without tricky + slow techniques like stemming, lemmatization, …

15K

noisy

2

  - Robust to typos

3

Bringsing

bringing

→ X!

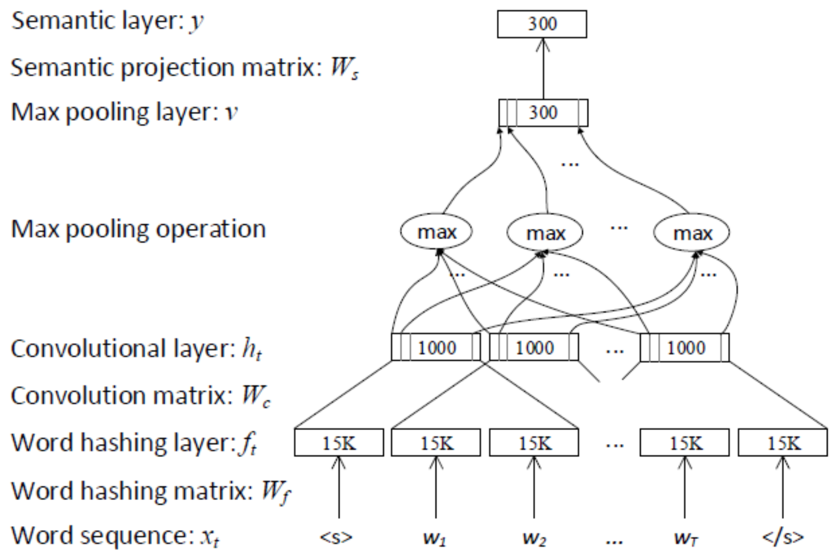


Figure 6: The architecture of the convolutional neural networks (CNN) used in this work. The CNN model maps a variable-length word sequence (e.g., a pattern or predicate sequence) to a low-dimensional vector in a latent semantic space. See text for the description of each layer.

Deep convolutional neural networks

- Uses convolution layer to project the letter-trigram vectors of words within a context window of 3 words

$f_t \rightarrow h_t$

- Creates a local contextual feature vector

h_t v

- Local to global context vector using max pooling operation
- Final fully connected feedforward layer to create final "semantic" vector

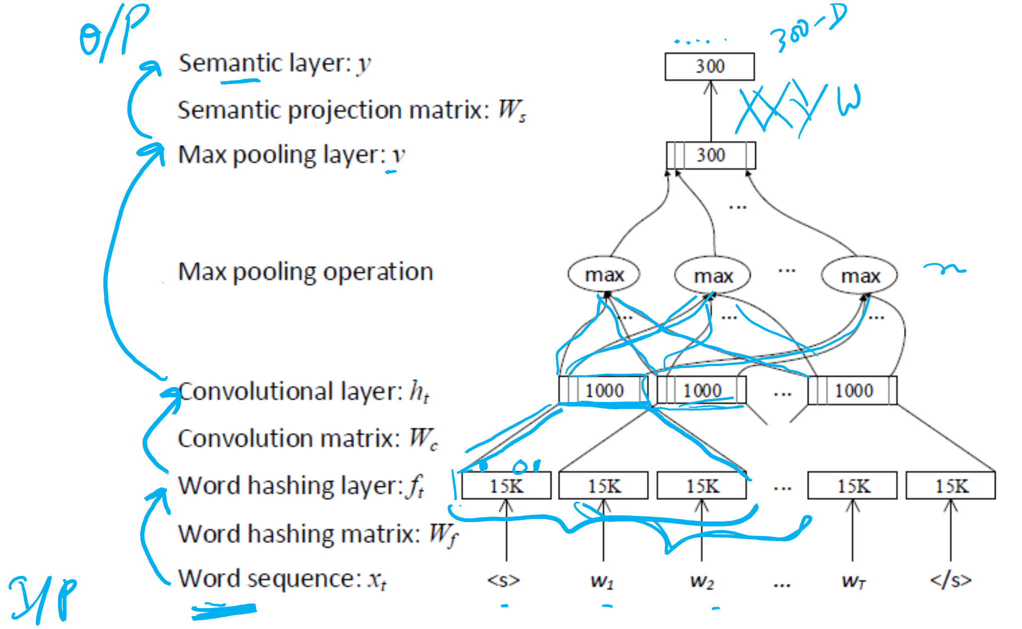
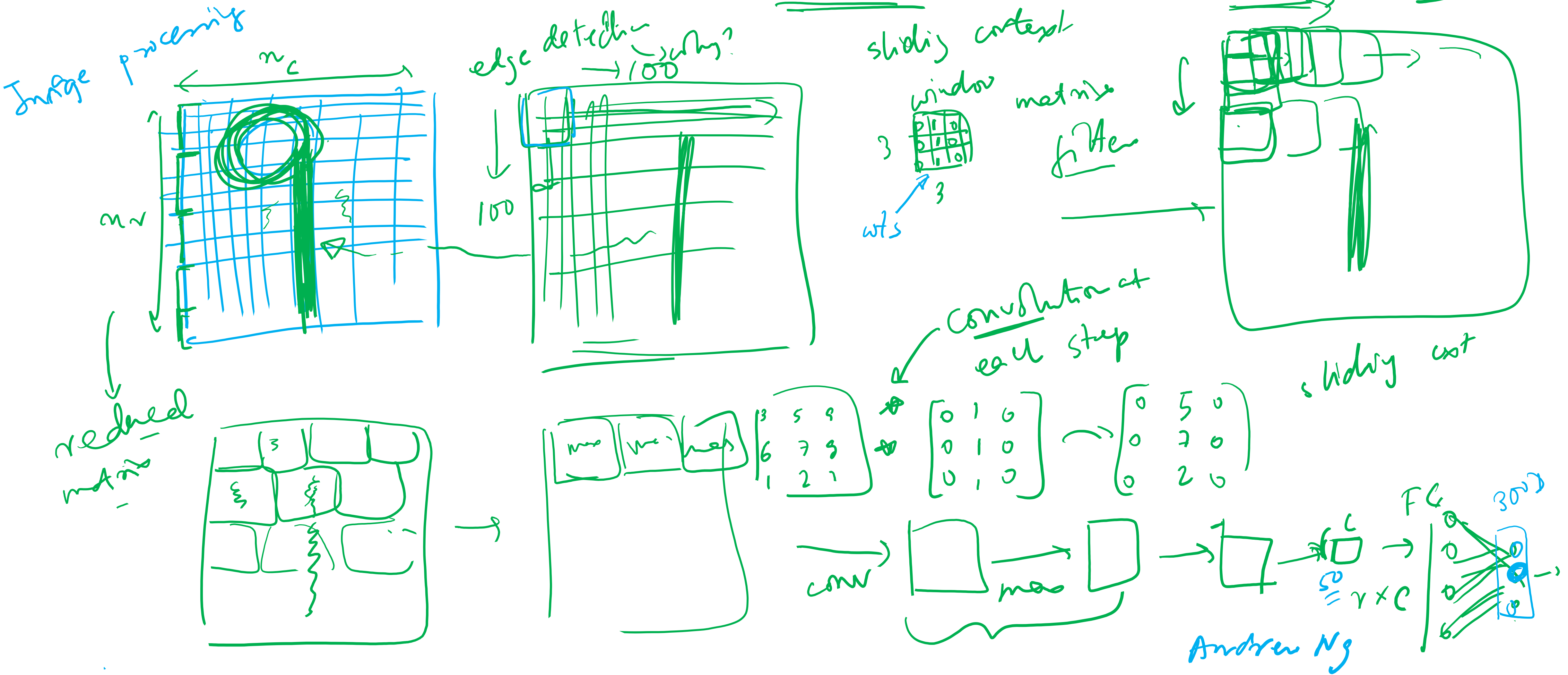


Figure 6: The architecture of the convolutional neural networks (CNN) used in this work. The CNN model maps a variable-length word sequence (e.g., a pattern or predicate sequence) to a low-dimensional vector in a latent semantic space. See text for the description of each layer.

300

avg

Convolution and max pooling



Deep convolutional neural networks

- Training model needs positive (and negative) pairs: (question, core-chain)
- Sample +ve pair

  sp ic

  (who first voiced meg on <e>, cast-actor)
- Obtained from SPARQL queries (semantic parses = correct subgraph patterns)

  cf
- If not availa~~ble, create Q-q pairs~~ by using Q-A paths in KG: Distant supervision!

  Web Questions sp

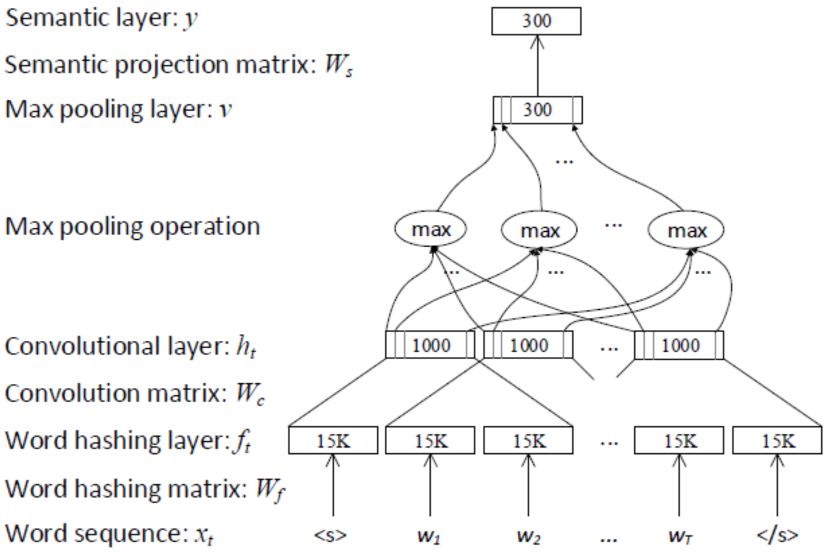
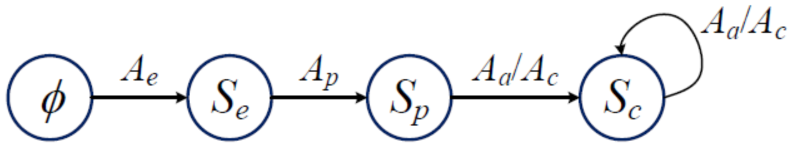
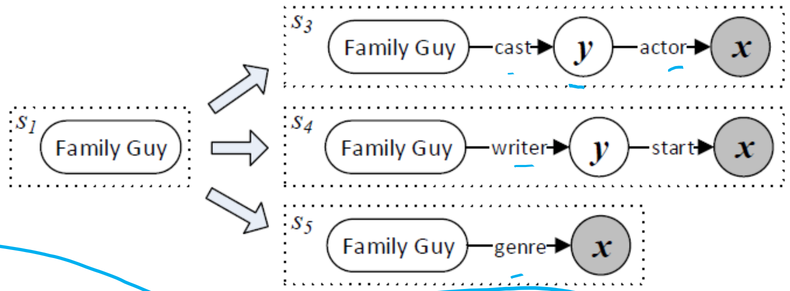


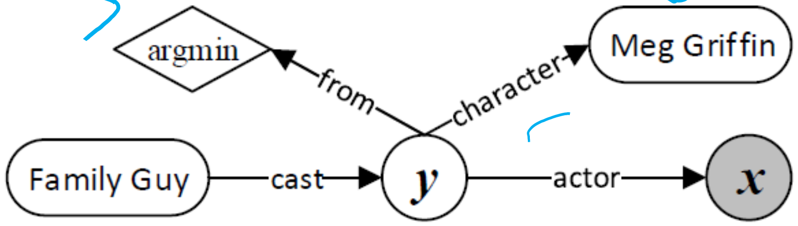
Figure 6: The architecture of the convolutional neural networks (CNN) used in this work. The CNN model maps a variable-length word sequence (e.g., a pattern or predicate sequence) to a low-dimensional vector in a latent semantic space. See text for the description of each layer.

Augmenting constraints and aggregations

- Graph with only core chain can be executed to get candidate answers: Still too many!
- Use conditions as filters
  - 1 Entity constraint Meg
  - 2 Keywords like first, last, … (rule-based)







query score

Learning reward function

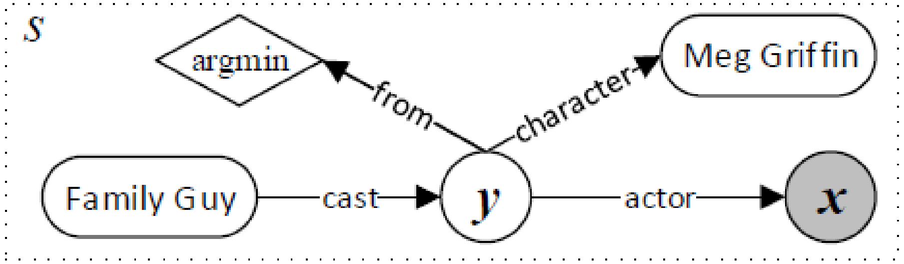
not RL

features

- Query ranking features g1 g2 g3 g4 g5
- Topic Entity E
- Core Inferential Chain P
  - PatChain (2 CNNs) } Siamese
  - QuesEP (2 more CNNs) } Siamese
  - ClueWeb (1 more CNN) FACC1
- Constraints and aggregations: Rules
- Overall: #Answers fetched and #nodes in graph

P

q = "Who first voiced Meg on Family Guy?"

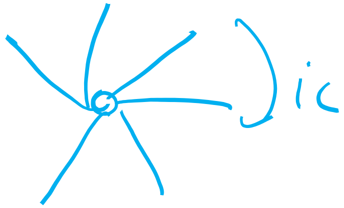


(1) EntityLinkingScore(FamilyGuy, "Family Guy") = 0.9
(2) PatChain("who first voiced meg on <e>", cast-actor) = 0.7
(3) QuesEP(q, "family guy cast-actor") = 0.6
(4) ClueWeb("who first voiced meg on <e>", cast-actor) = 0.2
(5) ConstraintEntityWord("Meg Griffin", q) = 0.5
(6) ConstraintEntityInQ("Meg Griffin", q) = 1
(7) AggregationKeyword(argmin, q) = 1
(8) NumNodes(s) = 5
(9) NumAns(s) = 1

Figure 8: Active features of a query graph s. (1) is the entity linking score of the topic entity. (2)-(4) are different model scores of the core chain. (5) indicates 50% of the words in "Meg Griffin" appear in the question q. (6) is 1 when the mention "Meg" in q is correctly linked to `MegGriffin` by the entity linking component. (8) is the number of nodes in s. The knowledge base returns only 1 entity when issuing this query, so (9) is 1.

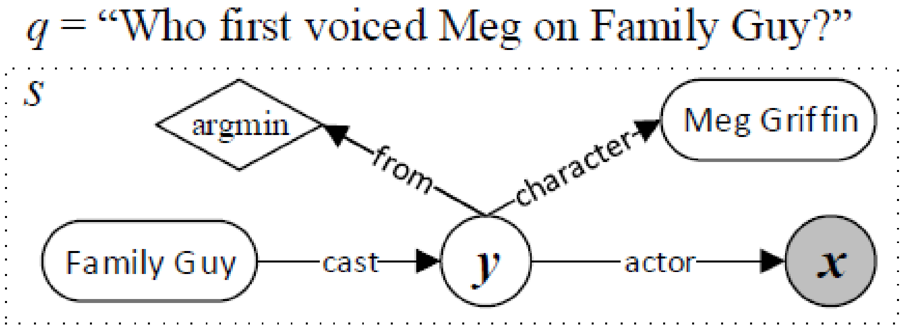
Learning reward function

- STAGG treats prediction as ranking task and not ~~a~~ binary classification problem where only the correct query graphs are labeled as positive



- Why? Closer to training data
  - Parse-graph, F1-score (ret. ans, gold ans WQ)
- One-layer neural network model based on Lambda-rank → Get best graph, execute and bingo!

LTR



(1) EntityLinkingScore(FamilyGuy, "Family Guy") = 0.9
(2) PatChain("who first voiced meg on <e>", cast-actor) = 0.7
(3) QuesEP(q, "family guy cast-actor") = 0.6
(4) ClueWeb("who first voiced meg on <e>", cast-actor) = 0.2
(5) ConstraintEntityWord("Meg Griffin", q) = 0.5
(6) ConstraintEntityInQ("Meg Griffin", q) = 1
(7) AggregationKeyword(argmin, q) = 1
(8) NumNodes(s) = 5
(9) NumAns(s) = 1

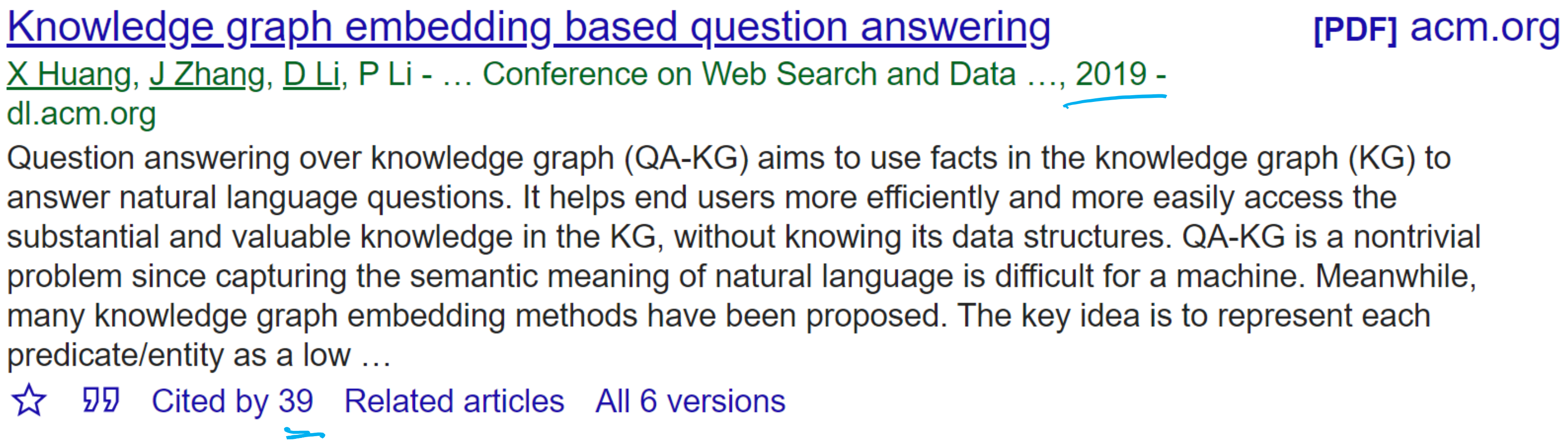
Figure 8: Active features of a query graph s. (1) is the entity linking score of the topic entity. (2)-(4) are different model scores of the core chain. (5) indicates 50% of the words in "Meg Griffin" appear in the question q. (6) is 1 when the mention "Meg" in q is correctly linked to `MegGriffin` by the entity linking component. (8) is the number of nodes in s. The knowledge base returns only 1 entity when issuing this query, so (9) is 1.

Cool ideas in STAGG

- ✓ Graph-based search
- ✓ Context subgraph
- ✓ Execution order (s, P, C)
- ✓ Top-k NED
- ✓ Character trigrams
- ✓ (Preliminary) ideas for handing complexity: core simple chain and conditions ✓
- ✓ Distant supervision for training Q-q from Q-A
- ✓ CNN models for similarity (representation)
- ✓ Pose prediction as neural LTR

Research paper 2

Knowledge Graph Embedding Based Question Answering



Knowledge graph embedding based question answering

[PDF] acm.org

X Huang, J Zhang, D Li, P Li - … Conference on Web Search and Data …, 2019 - dl.acm.org

Question answering over knowledge graph (QA-KG) aims to use facts in the knowledge graph (KG) to answer natural language questions. It helps end users more efficiently and more easily access the substantial and valuable knowledge in the KG, without knowing its data structures. QA-KG is a nontrivial problem since capturing the semantic meaning of natural language is difficult for a machine. Meanwhile, many knowledge graph embedding methods have been proposed. The key idea is to represent each predicate/entity as a low …

☆ ❞ Cited by 39 Related articles All 6 versions

The KEQA System

QA-KG or KG-QA? or KB-QA?

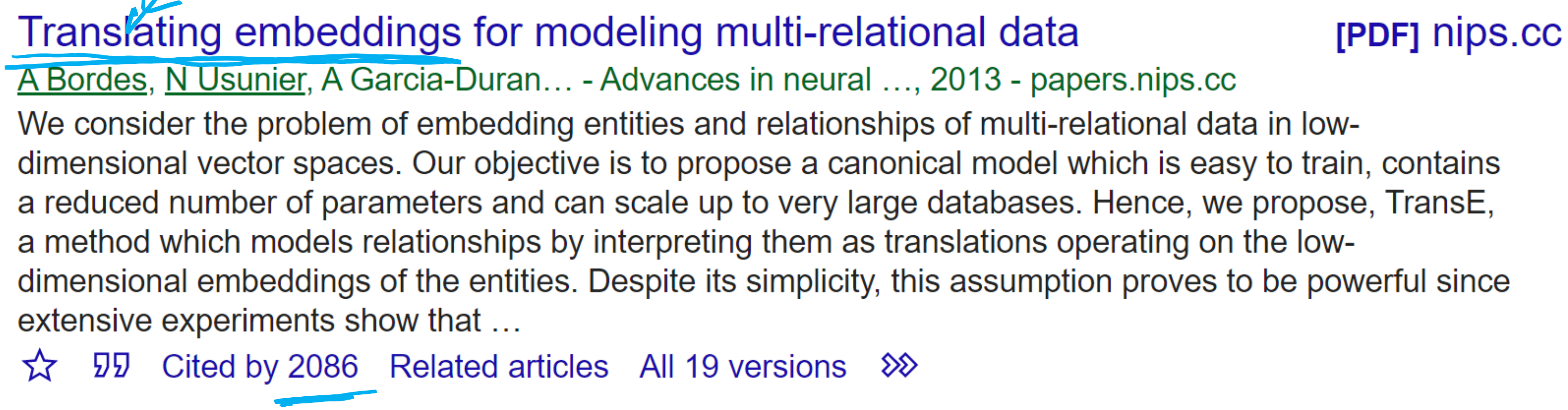
- Leverages knowledge graph embeddings! → for QA → KGC, link prediction
- Head entity, predicate, tail entity

S P O

e_h Ger — capital — Berlin e_t p

- Uses the TransE Model

TransH, TransR, RotatE



Translating embeddings for modeling multi-relational data **[PDF]** nips.cc

A Bordes, N Usunier, A Garcia-Duran… - Advances in neural …, 2013 - papers.nips.cc

We consider the problem of embedding entities and relationships of multi-relational data in low-dimensional vector spaces. Our objective is to propose a canonical model which is easy to train, contains a reduced number of parameters and can scale up to very large databases. Hence, we propose, TransE, a method which models relationships by interpreting them as translations operating on the low-dimensional embeddings of the entities. Despite its simplicity, this assumption proves to be powerful since extensive experiments show that …

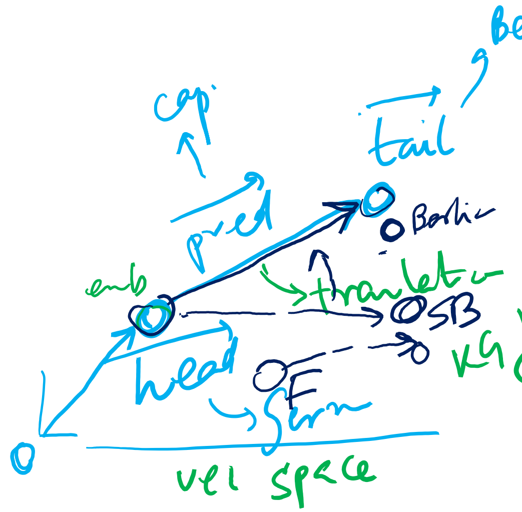
☆ 99 Cited by 2086 Related articles All 19 versions »

The TransE model

F ↓

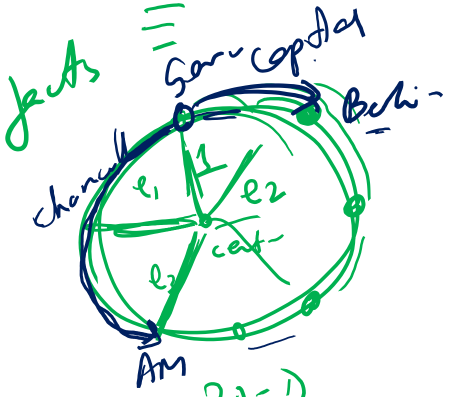
- 1 Head entity, predicate, tail entity
- 2 L2-norm of the embeddings of the entities is 1 unit vectors
- L2-norm of embeddings of predicates unconstrained

latent → vector space



all KG facts

KG facts ≡



sphere 20-D 50-D

+ve pair → KG fact
-ve pair ↳ corrupted

loss

$$\mathcal{L} = \overset{\text{correct}}{\sum_{(h,\ell,t)\in S}} \ \overset{\text{corrupt}}{\sum_{(h',\ell,t')\in S'_{(h,\ell,t)}}} \left[\gamma + d(\overset{S}{\mathbf{h}} + \overset{C}{\boldsymbol{\ell}}, \overset{B}{\mathbf{t}}) - d(\overset{F}{\mathbf{h}'} + \overset{C}{\boldsymbol{\ell}}, \overset{B}{\mathbf{t}'})\right]_+$$

Ger cap Berl
↳ France cap Berl
Germ cap SB

$$S'_{(h,\ell,t)} = \{(h',\ell,t) | h' \in E\} \cup \{(h,\ell,t') | t' \in E\}$$

The KEQA System

- Leverages knowledge graph embeddings!
- Uses the TransE Model (or TransE-like ...)
- Similar motivations
  - Name ambiguity
  - Predicate vocabulary mismatch
- From Baidu Research
- Simple questions, no qualifiers

Simple Question → FB

→ ans = obj

s p o → s p ?x

h p

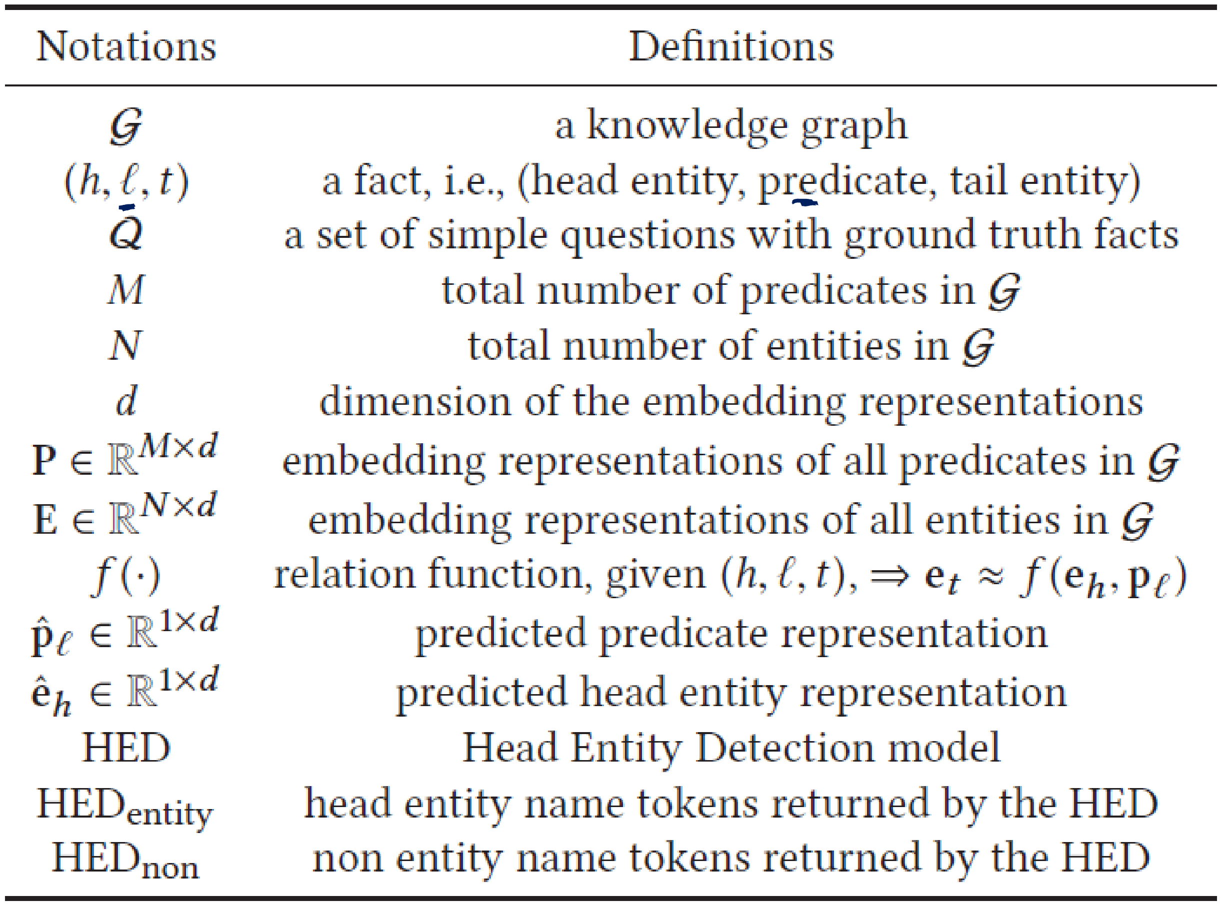
KEQA: Notations

- Given a question, we want to predict corr. fact's head entity and predicate

$$\overset{\text{tail}}{\mathbf{e}_t} \approx \overset{+}{f}(\mathbf{e}_h, \mathbf{p}_\ell)$$

$\vec{e_h} + \vec{p_l}$

Table 1: The important symbols and their definitions.



Notations	Definitions
$\mathcal{G}$	a knowledge graph
(h, ℓ, t)	a fact, i.e., (head entity, predicate, tail entity)
Q	a set of simple questions with ground truth facts
M	total number of predicates in $\mathcal{G}$
N	total number of entities in $\mathcal{G}$
d	dimension of the embedding representations
$\mathbf{P} \in \mathbb{R}^{M \times d}$	embedding representations of all predicates in $\mathcal{G}$
$\mathbf{E} \in \mathbb{R}^{N \times d}$	embedding representations of all entities in $\mathcal{G}$
$f(\cdot)$	relation function, given (h, ℓ, t), $\Rightarrow \mathbf{e}_t \approx f(\mathbf{e}_h, \mathbf{p}_\ell)$
$\hat{\mathbf{p}}_\ell \in \mathbb{R}^{1 \times d}$	predicted predicate representation
$\hat{\mathbf{e}}_h \in \mathbb{R}^{1 \times d}$	predicted head entity representation
HED	Head Entity Detection model
$\text{HED}_{\text{entity}}$	head entity name tokens returned by the HED
HED_{non}	non entity name tokens returned by the HED

KEQA: Outline

- 1 Based on Q and their corresponding predicates' embeddings, KEQA trains predicate learning model
- 2 Takes a question as the input and returns a predicate vector that lies in KG embedding space

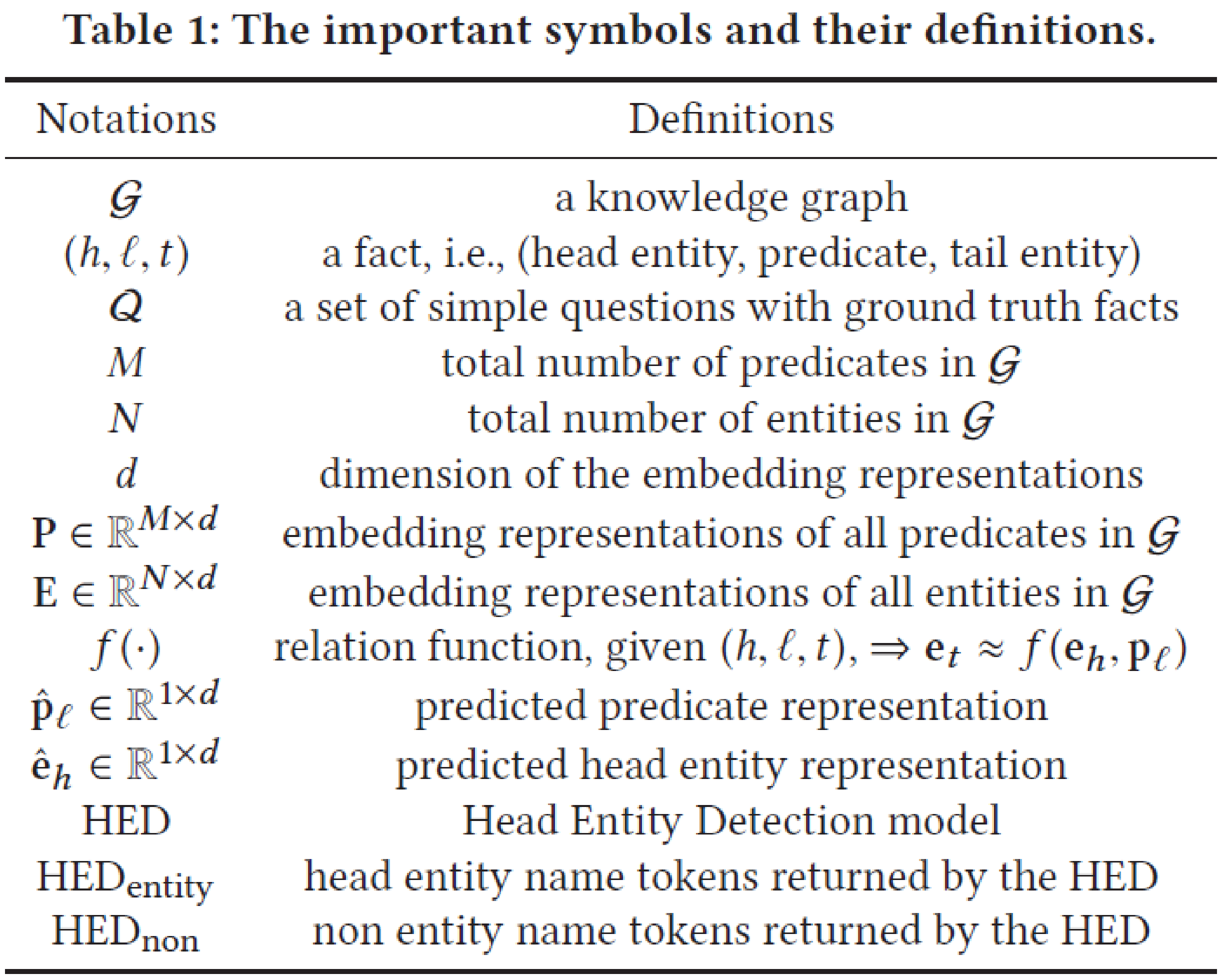


Table 1: The important symbols and their definitions.

Notations	Definitions
$\mathcal{G}$	a knowledge graph
(h, ℓ, t)	a fact, i.e., (head entity, predicate, tail entity)
$\mathcal{Q}$	a set of simple questions with ground truth facts
M	total number of predicates in $\mathcal{G}$
N	total number of entities in $\mathcal{G}$
d	dimension of the embedding representations
$\mathbf{P} \in \mathbb{R}^{M \times d}$	embedding representations of all predicates in $\mathcal{G}$
$\mathbf{E} \in \mathbb{R}^{N \times d}$	embedding representations of all entities in $\mathcal{G}$
$f(\cdot)$	relation function, given (h, ℓ, t), $\Rightarrow \mathbf{e}_t \approx f(\mathbf{e}_h, \mathbf{p}_\ell)$
$\hat{\mathbf{p}}_\ell \in \mathbb{R}^{1 \times d}$	predicted predicate representation
$\hat{\mathbf{e}}_h \in \mathbb{R}^{1 \times d}$	predicted head entity representation
HED	Head Entity Detection model
$\text{HED}_{\text{entity}}$	head entity name tokens returned by the HED
HED_{non}	non entity name tokens returned by the HED

KEQA: Outline

- Similarly for head entity: head entity learning

pe+

- Head entity detection ~NED
- Get tail entity

predicted representation

- Get closest fact in KG using distance function

$$\underline{\hat{\mathbf{e}}_t} = \overset{*}{f}(\underline{\hat{\mathbf{e}}_h}, \underline{\hat{\mathbf{p}}_\ell})$$

$$(\hat{\mathbf{e}}_h, \hat{\mathbf{p}}_\ell, \hat{\mathbf{e}}_t) \Rrightarrow \text{KG}$$

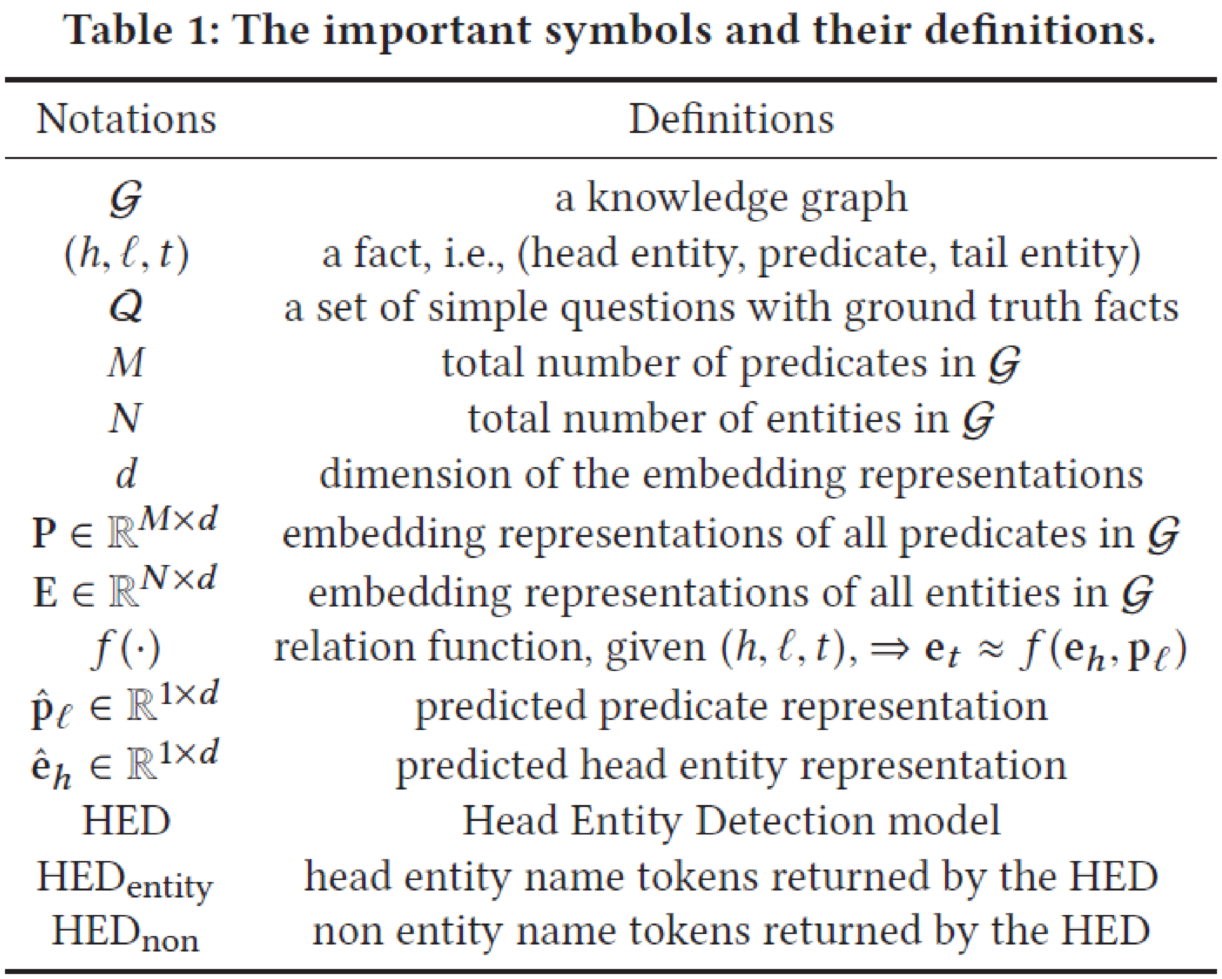


Table 1: The important symbols and their definitions.

Notations	Definitions
$\mathcal{G}$	a knowledge graph
(h, ℓ, t)	a fact, i.e., (head entity, predicate, tail entity)
$\mathcal{Q}$	a set of simple questions with ground truth facts
M	total number of predicates in $\mathcal{G}$
N	total number of entities in $\mathcal{G}$
d	dimension of the embedding representations
$\mathbf{P} \in \mathbb{R}^{M \times d}$	embedding representations of all predicates in $\mathcal{G}$
$\mathbf{E} \in \mathbb{R}^{N \times d}$	embedding representations of all entities in $\mathcal{G}$
$f(\cdot)$	relation function, given (h, ℓ, t), $\Rightarrow \mathbf{e}_t \approx f(\mathbf{e}_h, \mathbf{p}_\ell)$
$\hat{\mathbf{p}}_\ell \in \mathbb{R}^{1 \times d}$	predicted predicate representation
$\hat{\mathbf{e}}_h \in \mathbb{R}^{1 \times d}$	predicted head entity representation
HED	Head Entity Detection model
$\text{HED}_{\text{entity}}$	head entity name tokens returned by the HED
HED_{non}	non entity name tokens returned by the HED

KEQA: Outline

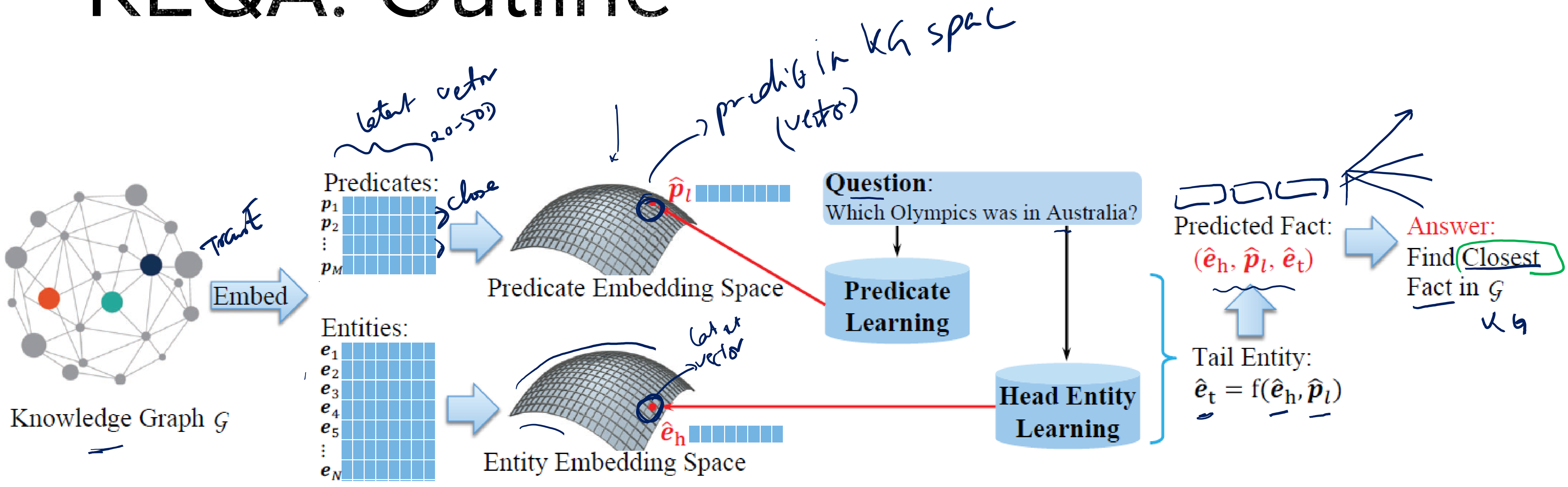


Figure 1: Instead of inferring the head entity and predicate directly, KEQA targets at jointly recovering the question's head entity, predicate, and tail entity representations $(\hat{e}_h, \hat{p}, \hat{e}_t)$ in the knowledge graph embedding spaces.

①

Predicate and head entity learning models

- Bi-LSTM for word order → in Q
  - Useful model for sequence models for NLP
- imp wts
  Attention for word importance
- Learning representations generalization to unseen concrete predicates at test time!! ✓✓

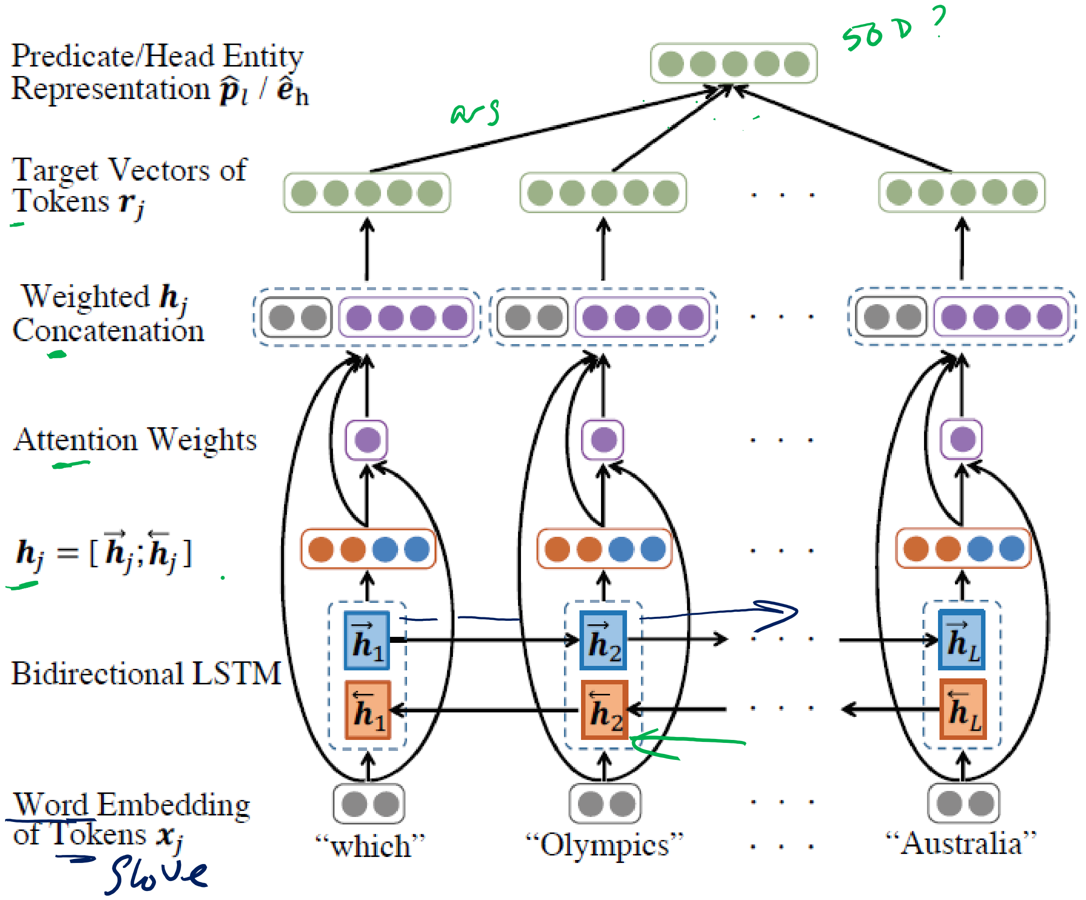


Figure 2: The architecture of the proposed predicate and head entity learning models.

Head entity detection model

- Getting the right entity (embedding) is crucial
- Need to reduce entity embedding search space
- Only some words matter
- No attention component
- 2 values predicted per token

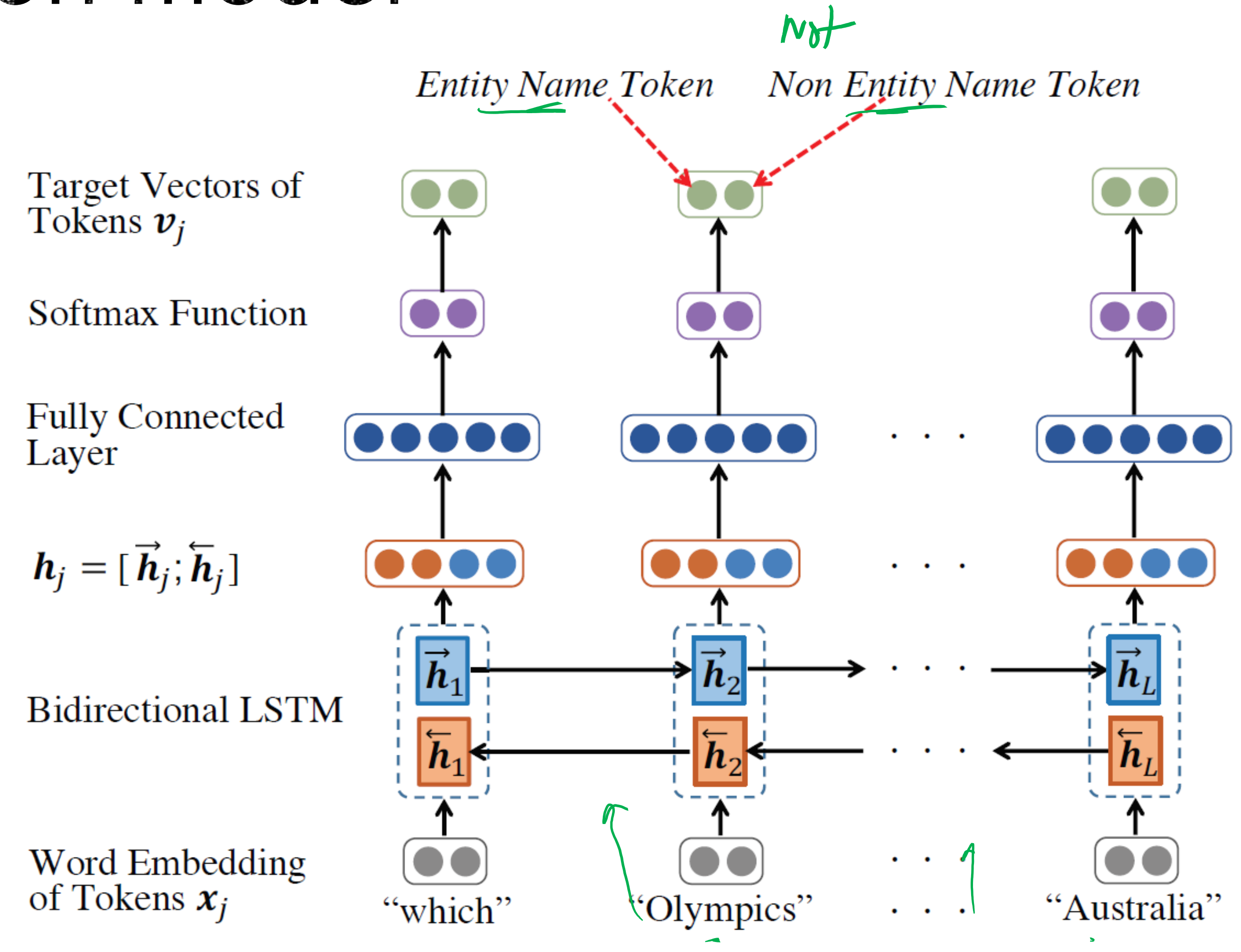


Figure 3: Structure of Head Entity Detection (HED) model.

Joint search on embedding spaces

- We now have predicted entity and predicate vectors
- Also, the head entities
- Now all we need is to define a distance metric for search!

β_1, β_2 ...

embedding sim

[illegible]

$$\underset{(h,\ell,t)\in C}{\text{arg minimize}} \quad \|\mathbf{p}_\ell - \hat{\mathbf{p}}_\ell\|_2 + \beta_1\|\mathbf{e}_h - \hat{\mathbf{e}}_h\|_2 + \beta_2\|\overbrace{f(\mathbf{e}_h, \mathbf{p}_\ell)}^{\text{tail}} - \hat{\mathbf{e}}_t\|_2$$

$$\underbrace{- \beta_3 sim[n(h), \mathrm{HED}_{\mathrm{entity}}] - \beta_4 sim[n(\ell), \mathrm{HED}_{\mathrm{non}}]}_{\text{string similarity}},$$

Conclusions

- Neural methods have shown great promise in KG-QA
- Think: Guidelines for where to apply neural learning in QA pipeline
- Graph representations of KBs very flexible and efficient for search
- Need large training data
- Lot depends on effective sampling of positive and negative pairs to train loss function