Baidu Research

Knowledge Graph Embedding Based Question Answering

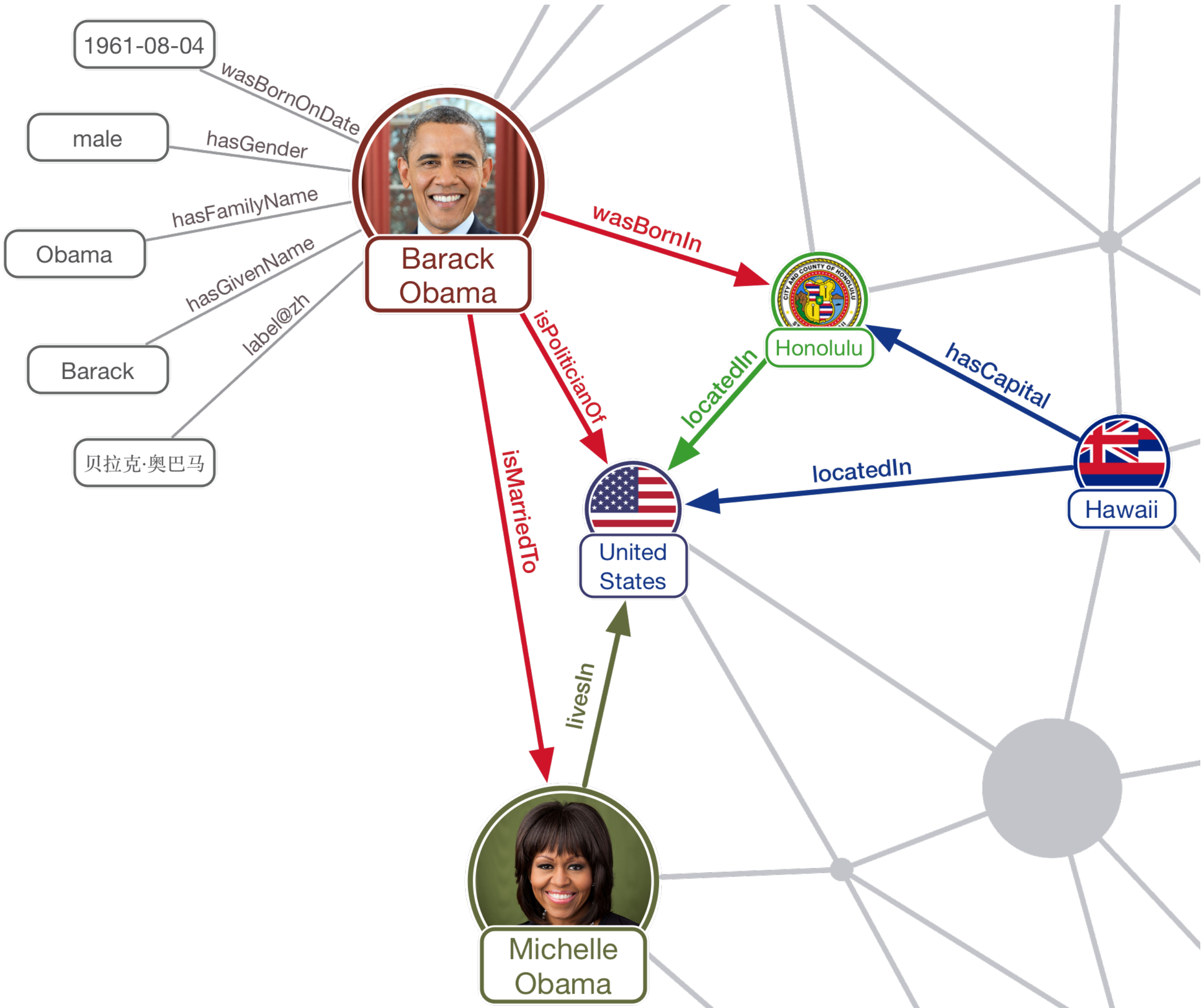
Xiao Huang, Dingcheng Li, Jingyuan Zhang, Ping Li

Cognitive Computing Lab (CCL), Baidu Research, USA

Emails: {huangxiao518, pingli98}@gmail.com, {lidingcheng, zhangjingyuan03}@baidu.com

Knowledge Graph

➢ A fact: head entity ⟶ predicate ⟶ tail entity



Question Answering Over Knowledge Graph is Crucial

- Large-scale knowledge graphs are available.
  - Difficult for regular users to find particular facts.

- Question answering over knowledge graph aims to automatically identify facts in KG to answer natural language questions.
  - It provides a way for AI systems to incorporate KG as a key ingredient to answer human questions.
  - Applications: search engine design & conversational agent building.

Challenges

- A predicate often has various expressions.
  - *person.nationality*: what is …'s nationality, which country is … from, where is … from, etc.
- Ambiguity of entity names and partial names make it hard to find correct entities.
  - Many entities share the same name.
  - Partial names: how old is Obama?

- Domains of end users' questions are often unbounded.
  - Any KG is far from complete.
  - New questions might involve predicates that are different from training ones.

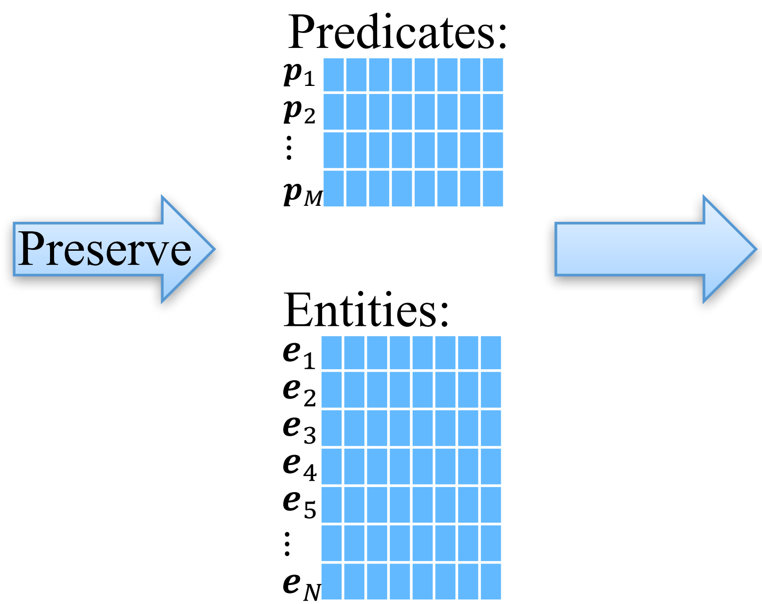
Existing Methods

- Semantic parsing based methods:
  - It converts natural language questions into logical expressions.

- Embedding based methods:
  - It projects questions and candidate facts into a unified low-dimensional space based on training questions.
  - It measure their matching scores by the similarities between their low-dimensional representations.
  - A typical way is to define a margin-based ranking criterion and train together with negative samples, i.e., wrong answers.

Opportunity: Knowledge Graph Embedding

Knowledge Graph $\mathcal{G}$



- KG Completion
- Recommendation
- Relation Extraction
- Question Answering?

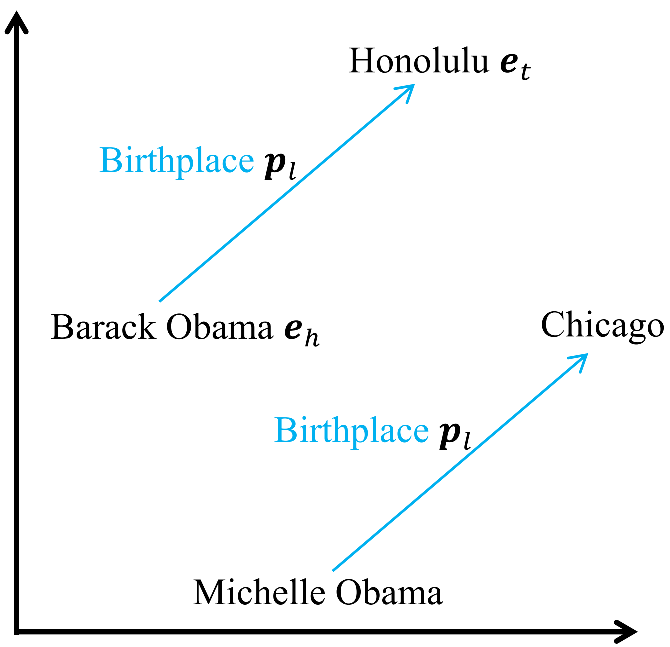
➢ The idea is to learn a low-dimensional vector representation for each predicate/entity in a KG to preserve original relations.

➢ Learn vector representations benefit downstream tasks.

- KG completion.
- Recommender systems.
- Relation extraction.

Knowledge Graph Embedding

- Represent each predicate/entity in a KG as a low-dimensional vector, such that original relations are preserved.



- Typical Solution
  - TransE

    $$\text{minimize} \sum \|\mathbf{e}_h + \mathbf{p}_\ell - \mathbf{e}_t\|_2^2$$

  - TransH

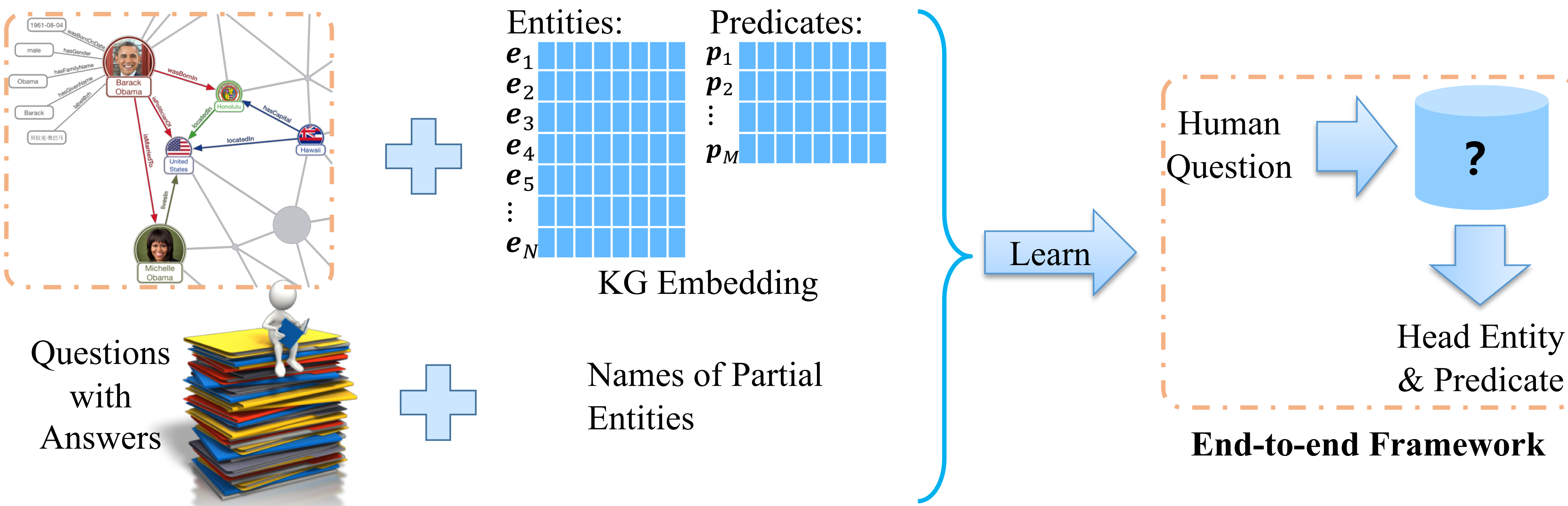
    $$\text{minimize} \sum \|\mathbf{e}_h^\perp + \mathbf{p}_\ell - \mathbf{e}_t^\perp\|_2^2,$$

    $$\text{where } \mathbf{e}_t^\perp = \mathbf{e}_t - \mathbf{m}_\ell^\top \mathbf{e}_t \mathbf{m}_\ell$$

  - TransR

    $$\text{minimize} \sum \|\mathbf{e}_h \mathbf{M}_\ell + \mathbf{p}_\ell - \mathbf{e}_t \mathbf{M}_\ell\|_2^2$$

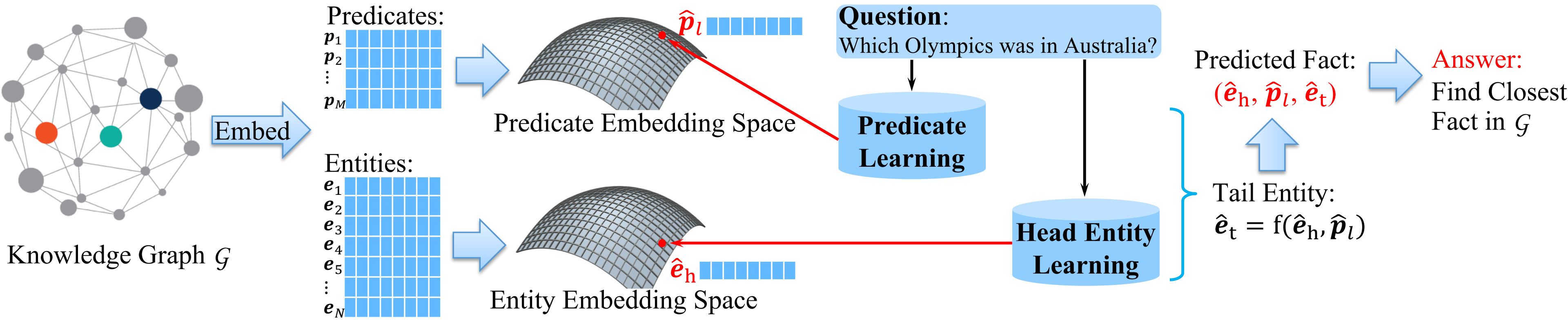
Problem Statement



- Input: a KG, predicates' and entities' names & embedding representations, training questions with answers.

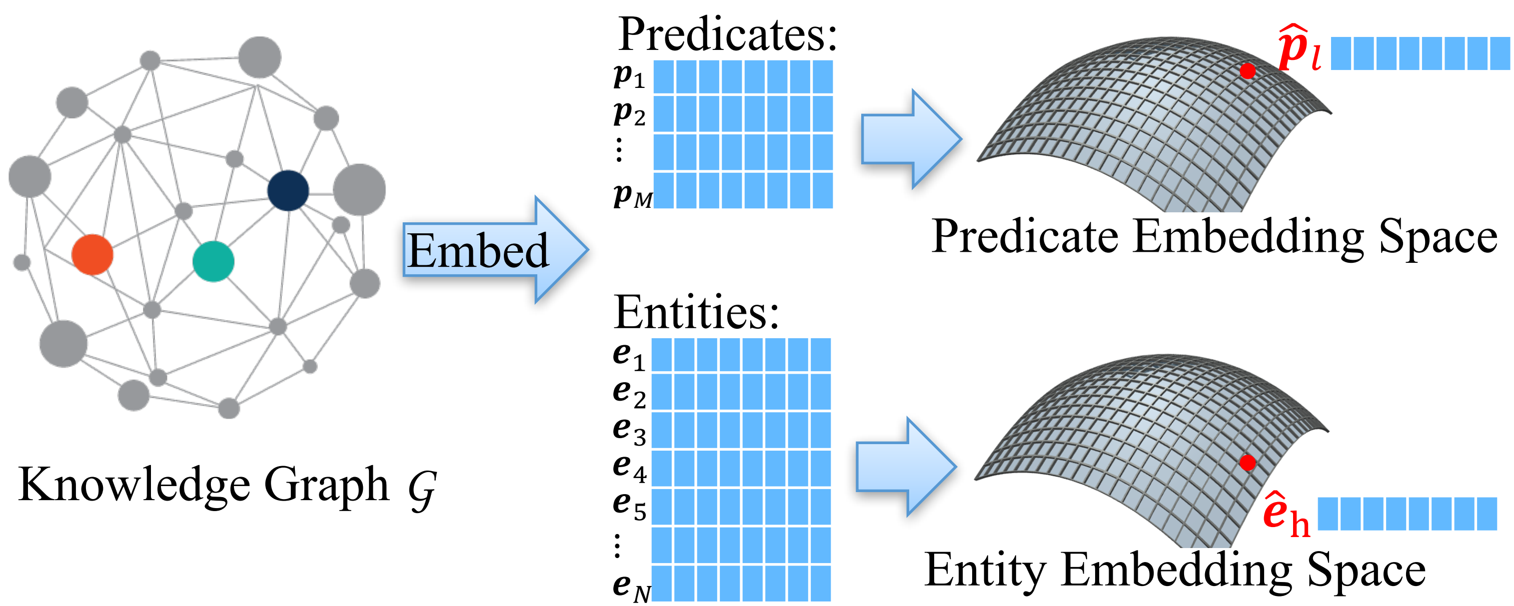
- Output: a trained end-to-end framework that takes a new simple question as input and returns its head entity & predicate.

Knowledge Embedding based Question Answering



- Each fact (h, l, t) can be represented as $(\boldsymbol{e}_h, \boldsymbol{p}_l, \boldsymbol{e}_t)$.
- Given a question, we aim to jointly predict $\boldsymbol{e}_h$, $\boldsymbol{p}_l$, and $\boldsymbol{e}_t$.
- Three components:
  - Predicate learning model & head entity learning model.
  - Head entity detection model.
  - Joint search on embedding spaces.

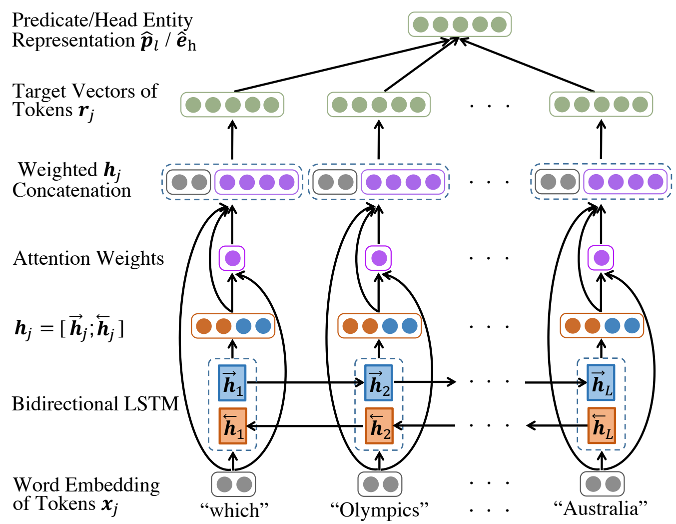
Predicate & Head Entity Learning Model



- Each fact (h, l, t) can be represented as $(\boldsymbol{e}_h, \boldsymbol{p}_l, \boldsymbol{e}_t)$.
  - For a question can be answered by KG, its predicates' vector representation lies in the predicate embedding space.

- Design a model.
  - Input: a question.
  - Output: a vector $\widehat{\boldsymbol{p}}_l$ that is as close as possible to the $\boldsymbol{p}_l$.

Predicate & Head Entity Learning Model



➢ Train on all training questions and use their $\boldsymbol{p}_l$ as the labels.

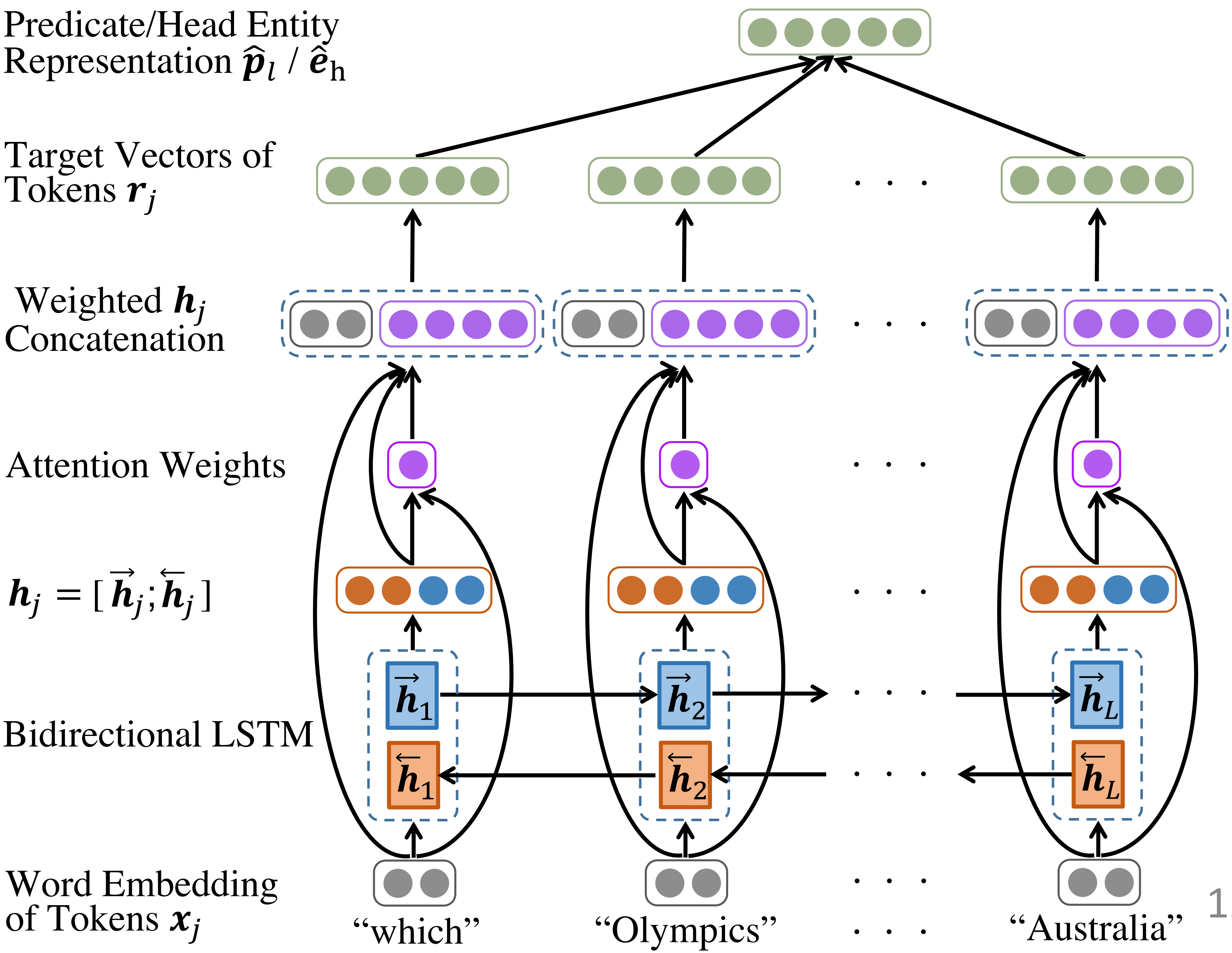
➢ Pseudocode:

for Q_i *in* Q **do**

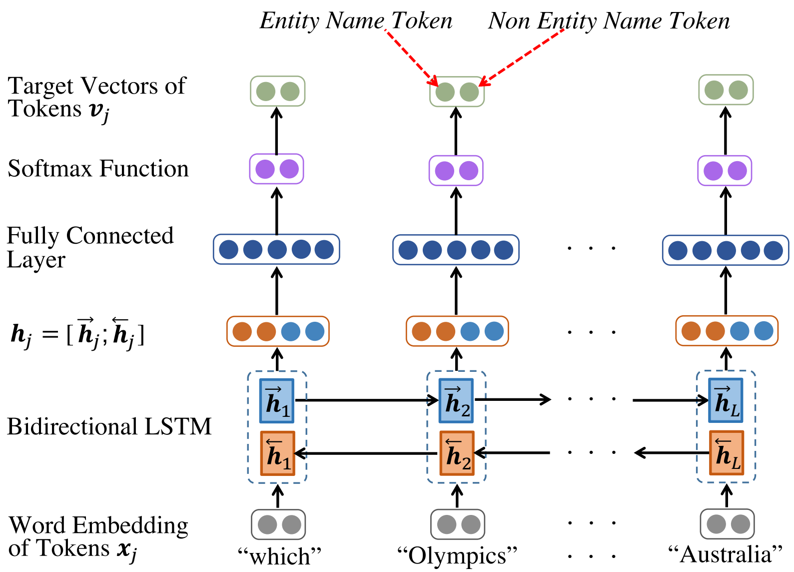
  Take the L tokens of Q_i as the input and its predicate ℓ as the label to train, as shown in Figure 2;

  Update weight matrices $\{\mathbf{W}\}$, $\mathbf{w}$, $\{\mathbf{b}\}$, and b_q to minimize the objective function $\|\mathbf{p}_\ell - \frac{1}{L}\sum_{j=1}^{L}\mathbf{r}_j^\top\|_2$;

Predicate & Head Entity Learning Model

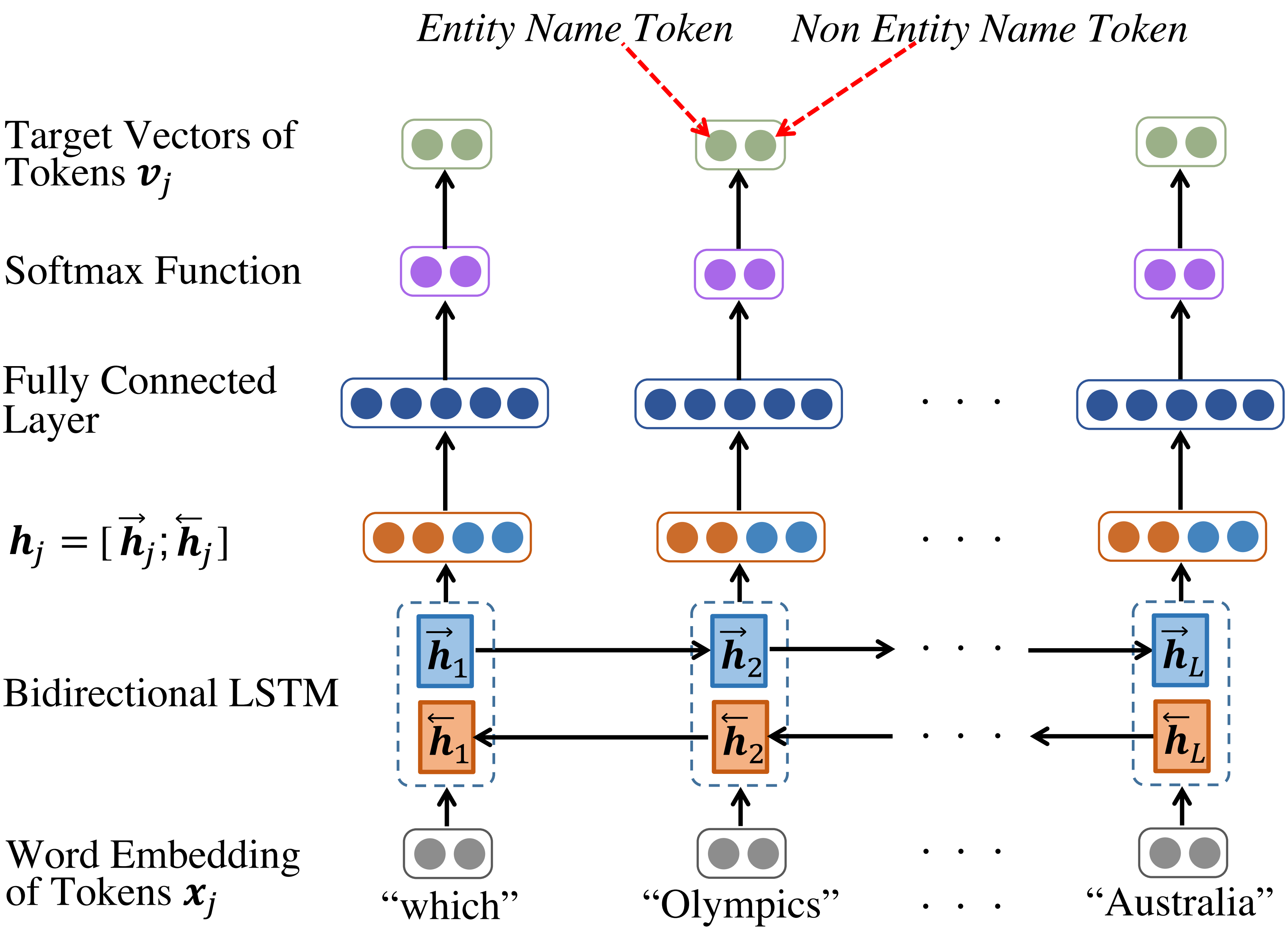


Head Entity Detection Model (HED)



- Select successive tokens as the name of head entity.
- Reduce the search space from entire entities to a number of entities with the same or similar names.
- Head entity name position is used as the label.
- $\hat{e}_h$ is used to handle the ambiguity.

Head Entity Detection Model (HED)



Joint Search on Embedding Spaces

- $$\underset{(h,\ell,t)\in\mathcal{C}}{\text{minimize}} \quad \|\mathbf{p}_\ell - \hat{\mathbf{p}}_\ell\|_2 + \beta_1\|\mathbf{e}_h - \hat{\mathbf{e}}_h\|_2 + \beta_2\|f(\mathbf{e}_h, \mathbf{p}_\ell) - \hat{\mathbf{e}}_t\|_2$$
$$- \beta_3 sim[n(h), \text{HED}_{\text{entity}}] - \beta_4 sim[n(\ell), \text{HED}_{\text{non}}].$$

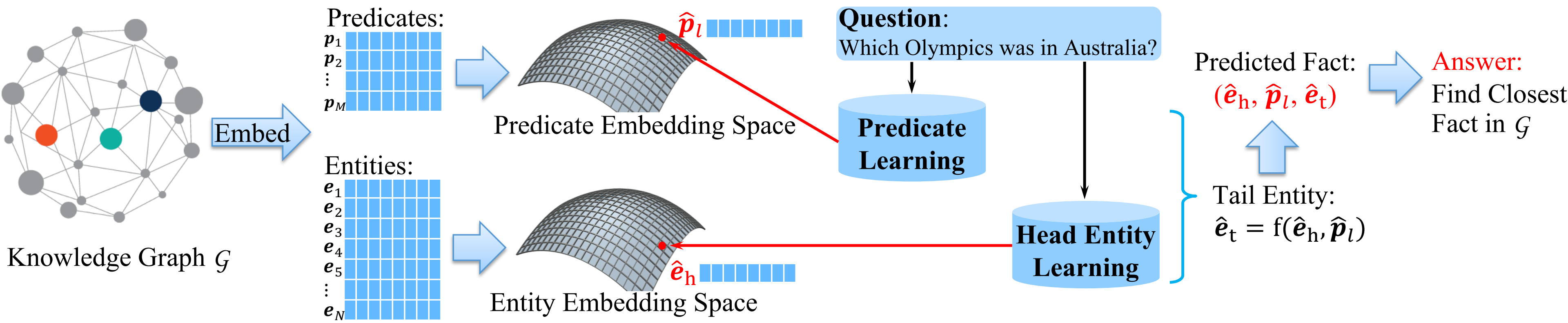
- $\hat{\mathbf{e}}_t = f(\hat{\mathbf{e}}_h, \hat{\mathbf{p}}_\ell)$.

- Function $n(\cdot)$ returns the name of the entity or predicate.

- $\text{HED}_{\text{entity}}$ and HED_{non} denote the tokens that are classified as entity name and non entity name by the HED model.

- sim$[\cdot,\cdot]$ measures the similarity of two strings.

Advantages of Proposed Framework



- KEQA could handle questions with predicates and entities that not exist in training data.
- KG embedding enables KEQA to perform head entity, predicate, and tail entity predictions jointly.
- KEQA is general to all KG embedding algorithms. It might be further improved by more effective embedding algorithms.

Datasets

- SimpleQuestions: Benchmark for most recent methods.
- FB2M & FB5M: subsets of Freebase knowledge graph.

	FB2M	FB5M	SimpleQuestions
# Training	14,174,246	17,872,174	75,910
# Validation	N.A.	N.A.	10,845
# Test	N.A.	N.A.	21,687
# Predicates (M)	6,701	7,523	1,837
# Entities (N)	1,963,130	3,988,105	131,681
# Words	733,278	1,213,205	61,336

Effectiveness of KEQA

	FB2M (Accuracy)	FB5M
Bordes et al. (2015) [6]	0.627	0.639
Dai et al.[3] (2016) [10]	N.A.	0.626
Yin et al. (2016) [46]	0.683 (+8.9%)	0.672
Golub and He (2016) [18]	0.709 (+13.1%)	0.703
Bao et al. (2016) [2]	0.728 (+16.1%)	Entire Freebase
Lukovnikov et al. (2017) [27]	0.712 (+13.6%)	N.A.
Mohammed et al.[5] (2018) [29]	0.732 (+16.7%)	N.A.
KEQA_noEmbed	0.731 (+16.6%)	0.726
KEQA	**0.754 (+20.3%)**	**0.749**

- KEQA outperforms all baselines.
- KEQA achieves 3.1% higher accuracy than KEQA_noEmbed.
- KEQA decreases 0.7% when applied to FB5M.

Experimental Results

	SimpleQuestions	SimpleQ_Missing
KEQA_noEmbed	0.731	0.386
KEQA_TransE	0.754 (+3.1%)	0.418 (+8.3%)
KEQA_TransH	0.749 (+2.5%)	0.411 (+6.5%)
KEQA_TransR	0.753 (+3.0%)	0.417 (+8.0%)

- Apply different KG embedding algorithms to learn the predicate and entity embedding representations.
- SimpleQ_Missing: All predicates in test have never been mentioned in the training and validation.
- KEQA is general and robust.

Conclusions

- We formally define knowledge graph embedding based question answering problem.

- KEQA could answer a natural language question by jointly recovering its head entity, predicate, and tail entity representations in the KG embedding spaces.

- We design a joint distance metric that takes the structures and relations preserved in the KG embedding representations into consideration.

- We empirically demonstrate that the separate task KG embedding indeed could help the question answering task.

Acknowledgement

- Cognitive Computing Lab
- Baidu Research
- Everyone attending the talk