KBQA: Learning Question Answering over QA Corpora and Knowledge Bases

PVLDB 2017

Wanyun Cui

SIME@SUFE

Wanyun Cui

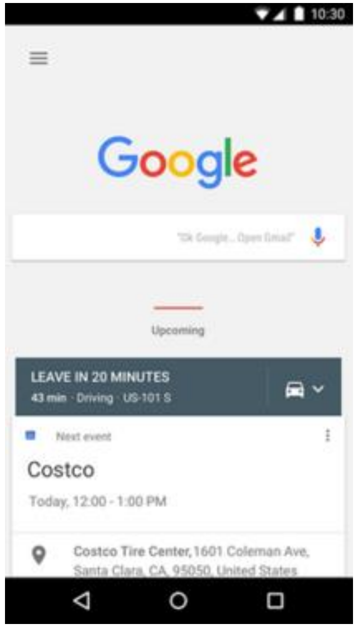
- Where do I come from?
  - 2017-Present, assistant professor, SUFE
  - 2013-2017 PhD candidate, Fudan University
  - 2009-2013 BS, Fudan University
- What do I work on?
  - Question answering
    - 2012.1 – 2012.11 Microsoft Research Asia
    - 2014.7 – 2014.11 Baidu DeepQA project (小度机器人)
    - 2015.10 – 2016.12 Xiaoi Robot (小i机器人)
  - Automatic domain knowledge base construction
    - Knowledge priority
    - Domain-aware knowledgeable sentence extraction
    - Domain adaptation for NLP
    - Chinese mention2entity / verb pattern
  - AI + finance

Backgrounds

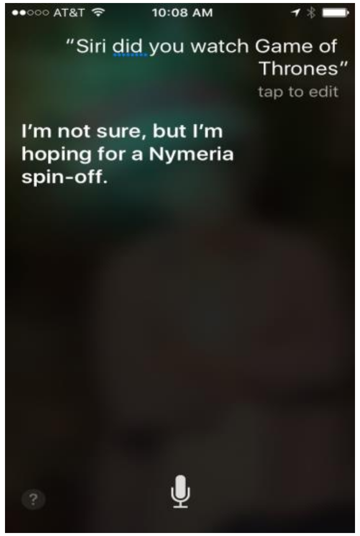
- Question Answering (QA) systems answer natural language questions.

IBM Watson

Google Now

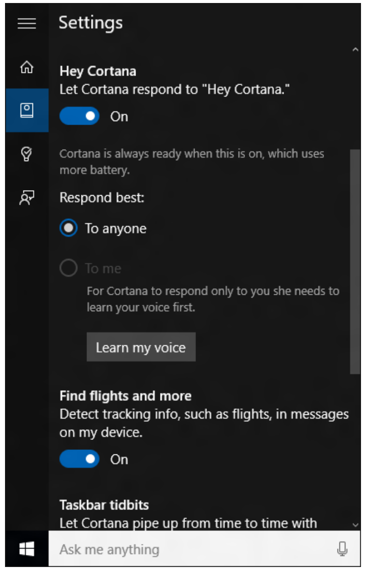


Apple Siri



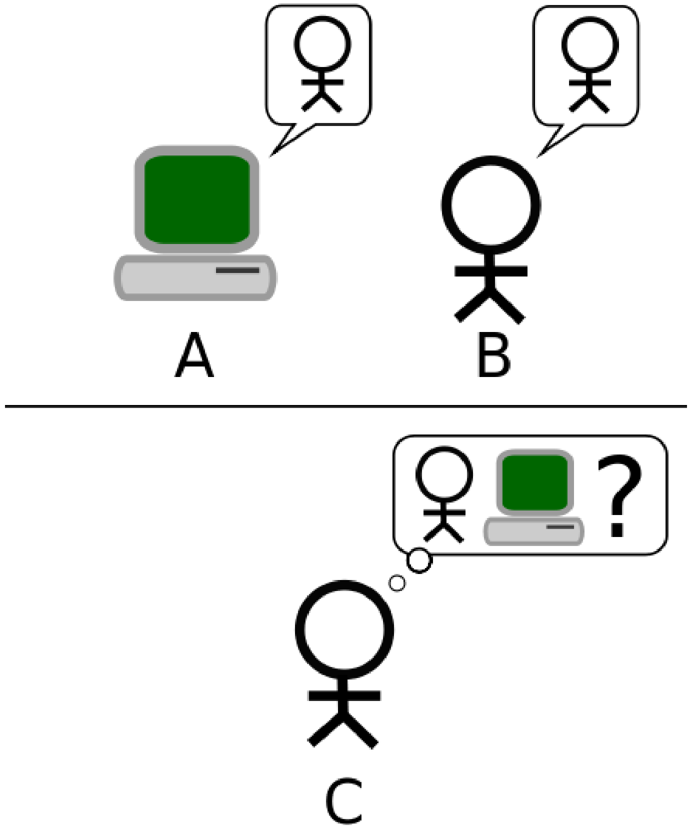
Amazon Alexa

Microsof Cortana



Why QA

- QA application:
  - One of the most natural human-computer interaction
  - Key components of Chatbot, which attracts wide research interests from industries
- QA for AI:
  - One of most important tasks to evaluate the machine intelligence: Turing test
  - Important testbed of many AI techniques, such as machine learning, natural language processing, machine cognition

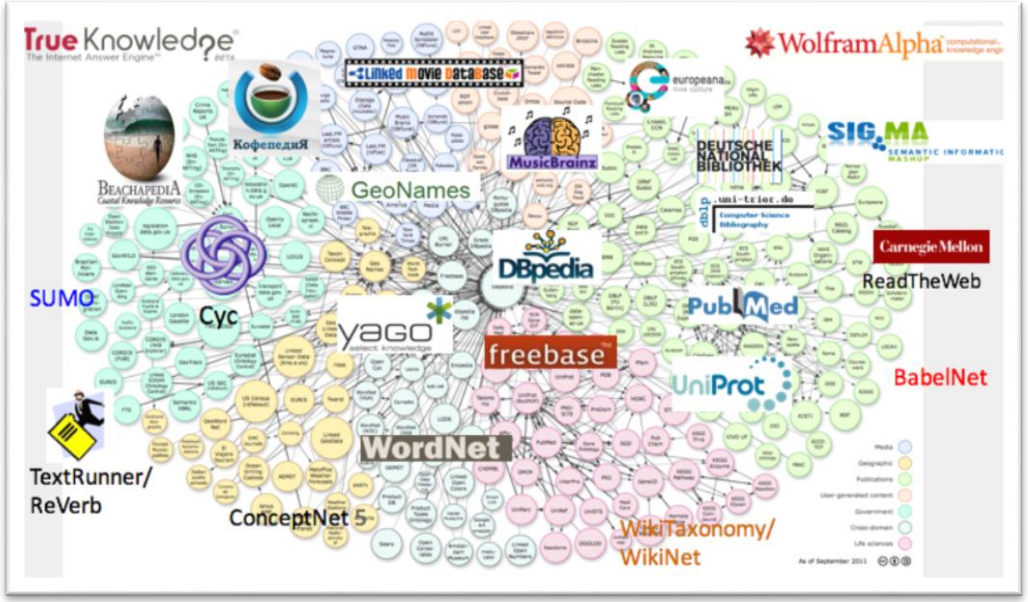


Turing test

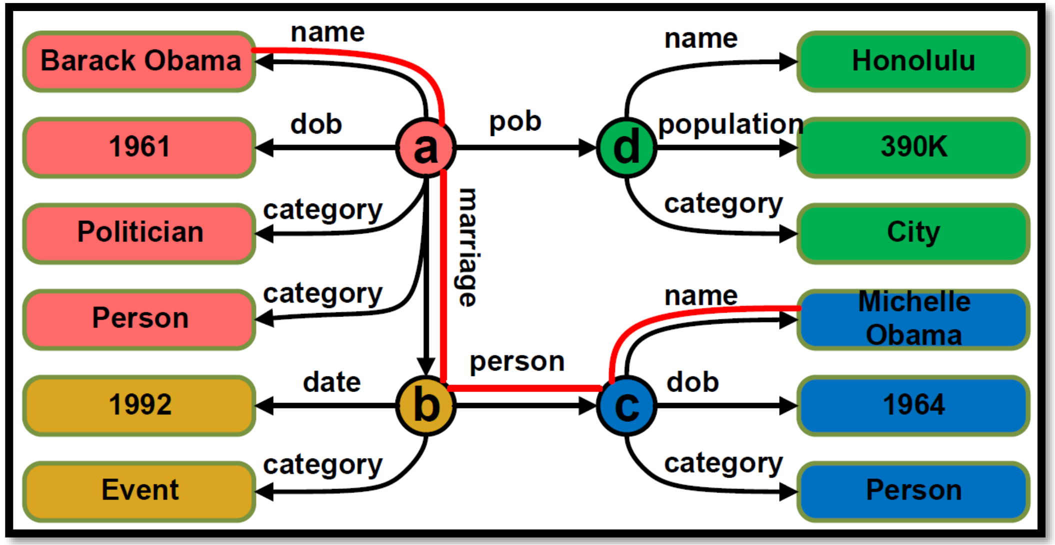
Why KBQA?

More and More Knowledge bases are created

- Google Knowledge graph, Yago, WordNet, FreeBase, Probase, NELL, CYC, DBPedia
- Large scale, clean data



The boost of knowledge bases



A piece of knowledge base, which consist of triples such as (d, population, 390k)

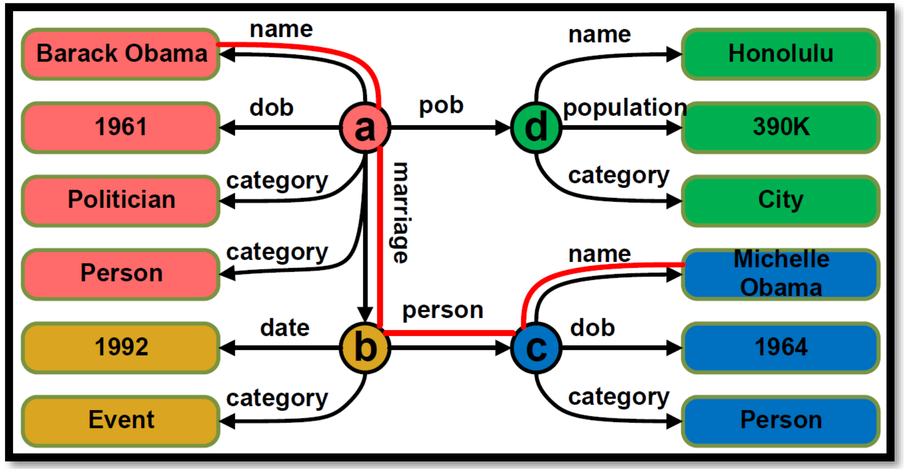
Why KBQA?

- Linked data – knowledge representation
  - Plain text: similarity between question and sentence.
  - KB: relational data provide semantics for question understanding
- Data quality – QA precision
  - Plain text: errors or contradictions in different texts
  - KB: high quality data from human labeling or table in web.
- Structured data – query efficiency
  - Plain text: inverted index
  - KB: stored in database, indexed by subject

How KB-based QA works?

- Convert natural language questions into structured queries over knowledge bases.

How many people live in Honolulu?



SPARQL

Select ?number
Where {
Res:Honolulu
dbo:population ?num
}

SQL

Select value
From KB
Where subject=‘d’ and
predicate=‘population’

- **Key: predicate inference**

Two challenges for predicate inference

- Question Representation
  - Identify questions with the same semantics
  - Distinguish questions with different intents
- Semantic matching
  - Map the question representation to the predicate in the KB
  - Vocabulary gap

Question in Natural language	Predicate in KB
ⓐ How many people are there in Honolulu?	population
ⓑ What is the population of Honolulu?	population
ⓒ What is the total number of people in Honolulu?	population
ⓓ When was Barack Obama born?	dob
ⓔ Who is the wife of Barack Obama?	marriage→person→name
ⓕ When was Barack Obama's wife born?	marriage→person→name dob

Weakness of previous solutions

- Template/rule based approaches
  - Questions are strings
  - Represent questions by string based templates, such as regular expression
  - By human labeling
  - PROs:
    - User-controllable
    - Applicable to industry use
  - CONs:
    - Costly human efforts.
    - Not good at handling the diversity of questions.

- Neural network based approaches
  - Questions are numeric
  - Represent questions by numeric embeddings
  - By learning from corpus
  - PROs:
    - Feasible to understand diverse questions
  - CONs:
    - Poor interpretability
    - Not controllable. Unfriendly to industrial application.

How to retain advantages from both approaches?

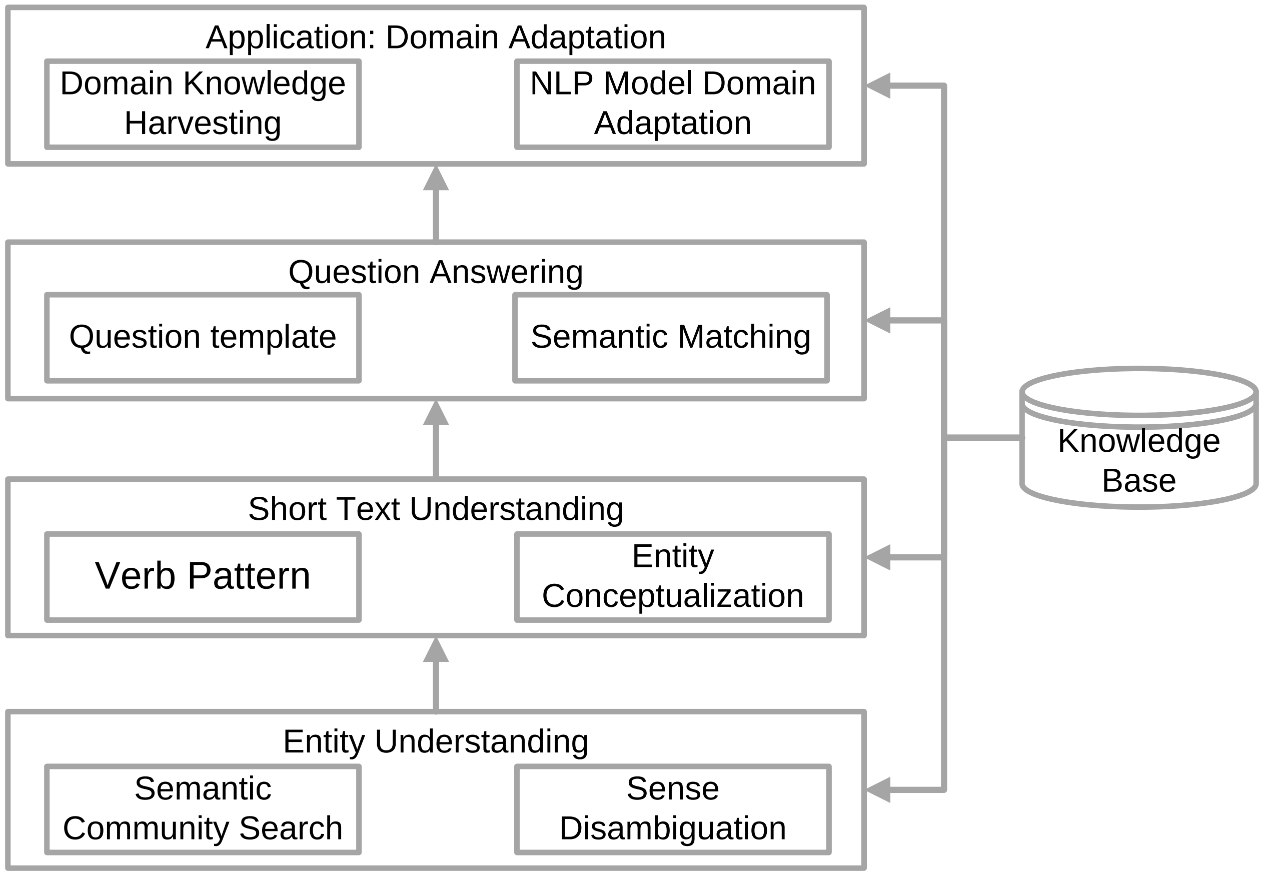
Our systematic work

Adapt QA system to specific domains

Core level: question answering

Short text connects entities/words and questions

Provides the basic semantic computing for entities in questions

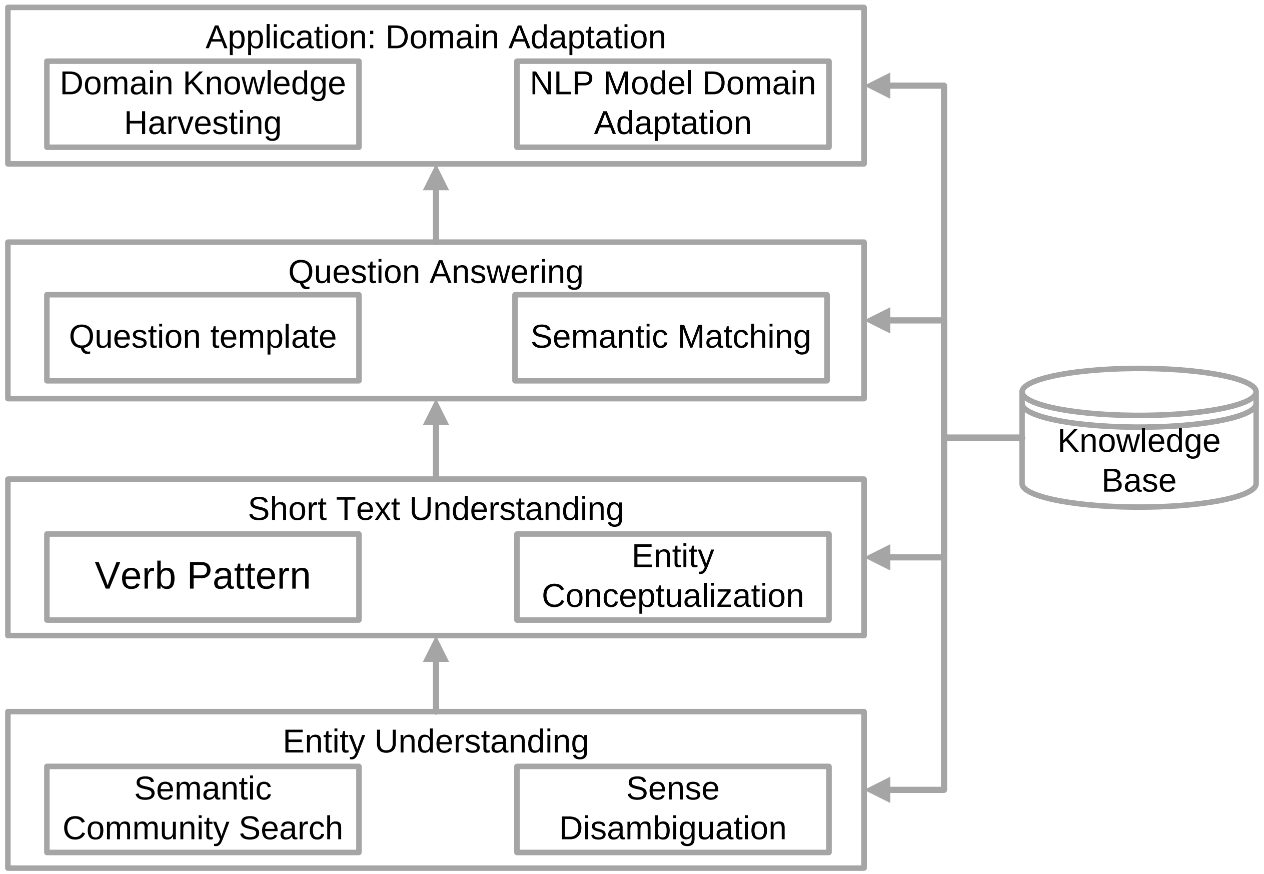


Our systematic work

IJCAI 2016， VLDB 2017

AAAI 2016

SIGMOD 2013， SIGMOD 2014



Our approach

- Representation: concept based templates.
  - Questions are asking about entities
  - Interpretable
  - User-controllable

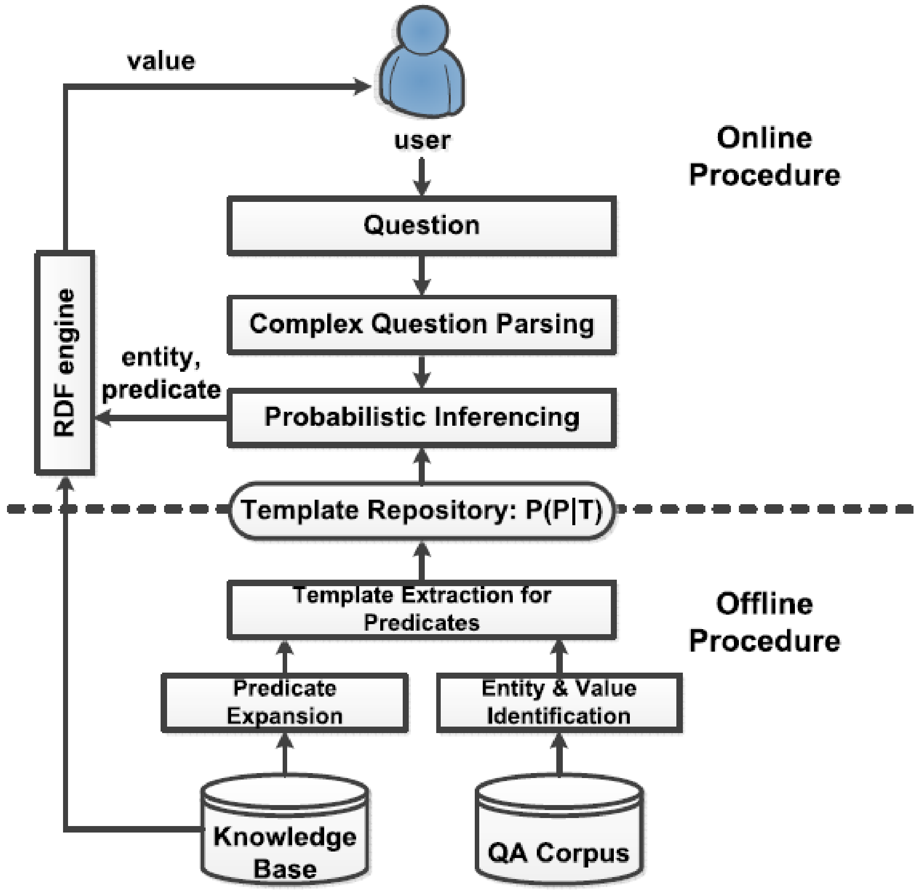
Question	What is the population of Honolulu?	How many people are there in Honolulu?	What is the number of people in Honolulu?
Template	What is the population of $City?	How many people are there in $City?	What is the total number of people in $City?

- Learn templates from QA corpus, instead of manfully construction.
  - 27 million templates, 2782 intents
  - Understand diverse questions

System Architecture

- Offline procedure
  - Learn the mapping from templates to predicates: P (p|t),
  - Input: qa corpora, large scale taxonomy, KB
  - Output: P(P|T)

- Online procedure
  - Parsing, predicate inference and answer retrieval
  - Input: binary factoid questions (BFQs)
  - Output: answers in KG



Problem Model

- Given a question q, our goal is to find an answer v with maximal probability (v is a simple value)

$$\arg\max_{v} P(V = v | Q = q) \longrightarrow \arg\max_{v} \sum_{e,t,p} P(v | q, e, t, p)$$

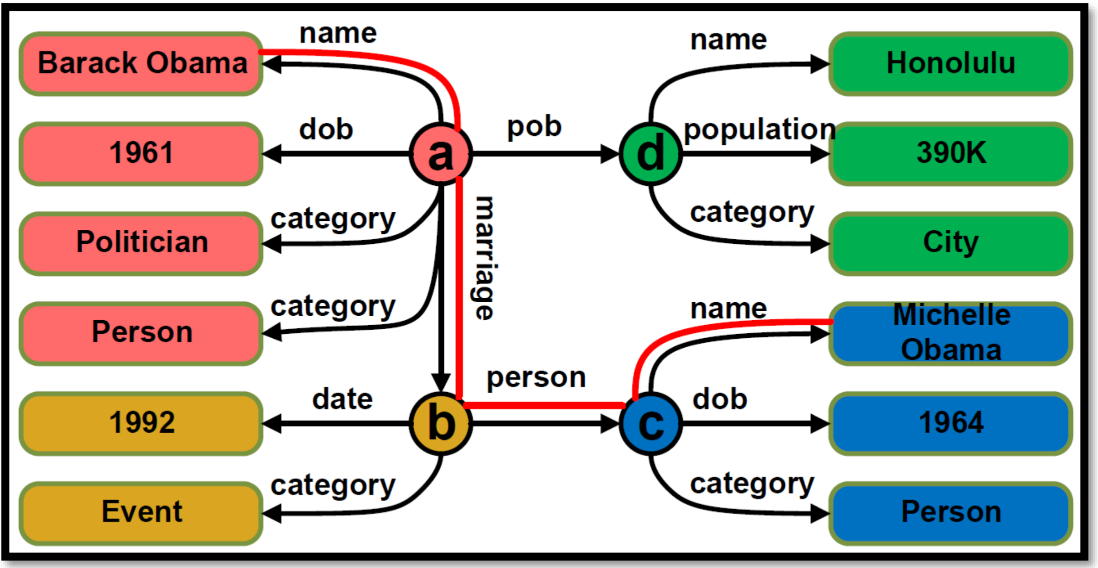
e: entity; t: template; p: predicate

- Basic idea：We proposed a generative model to explain how a value is found for a given question,

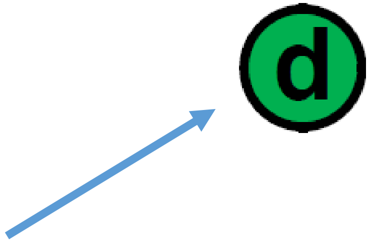
- Rationality of probabilistic inference
  - *uncertainty* (e.g. some questions' intents are vague)
  - *Incompleteness* (e.g. the knowledge base is almost always incomplete),
  - *noisy* (e.g. answers in the QA corpus could be wrong)

question2answer: a generative process

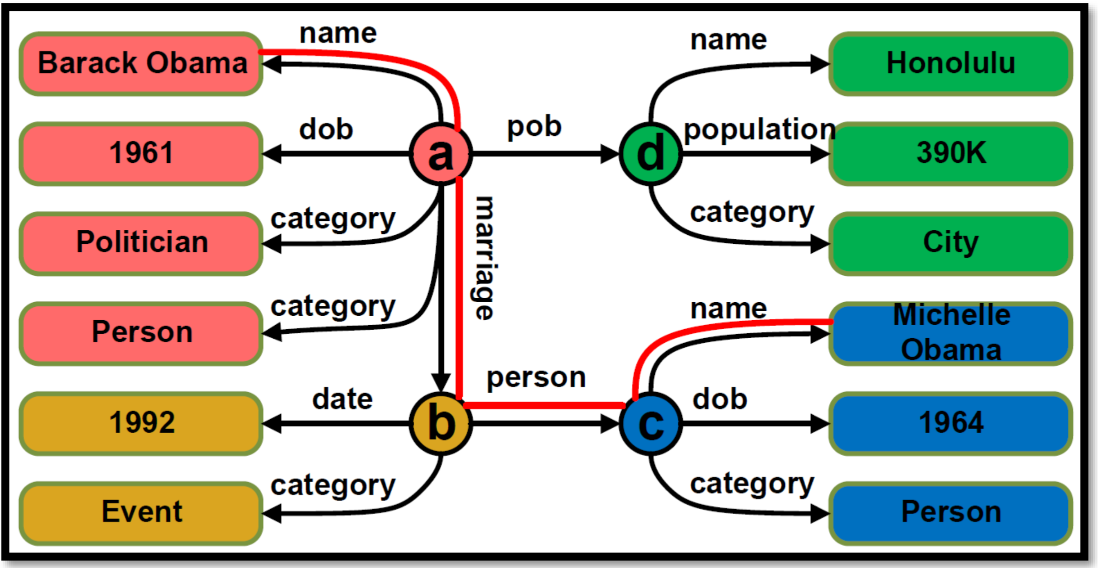
- A qa pair
  - Q: How many people live in Honolulu?
  - A: It’s 390K.



question2answer: entity linking

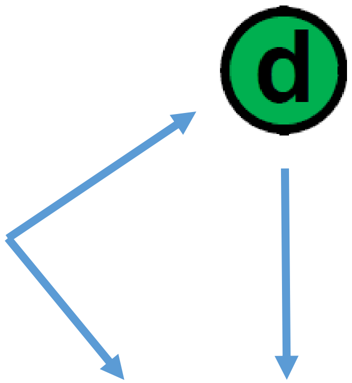


How many people live in Honolulu?

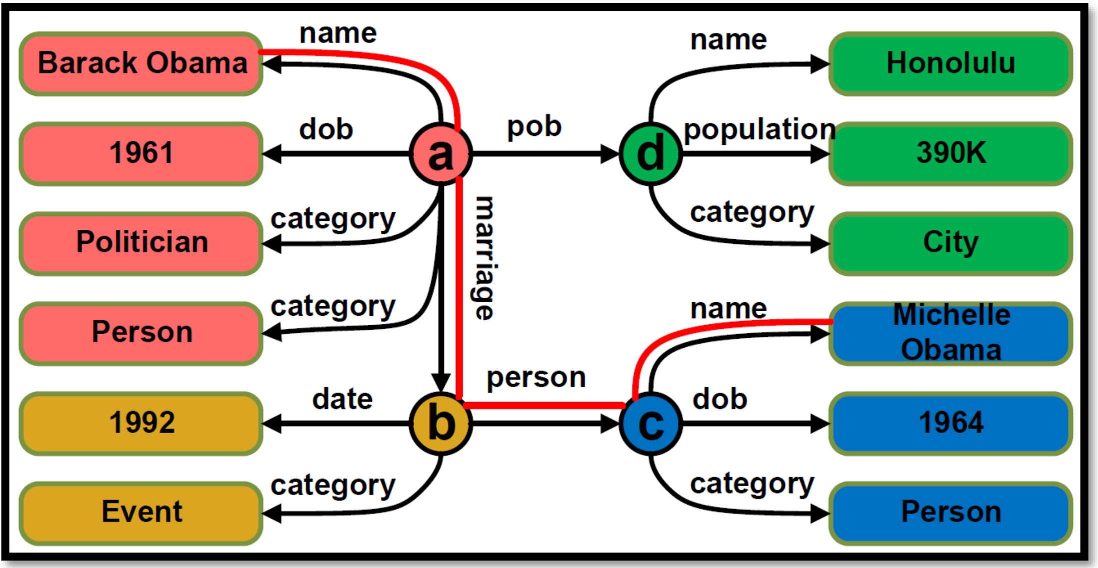


question2answer: conceptualization

How many people live in Honolulu?



How many people live in $city?

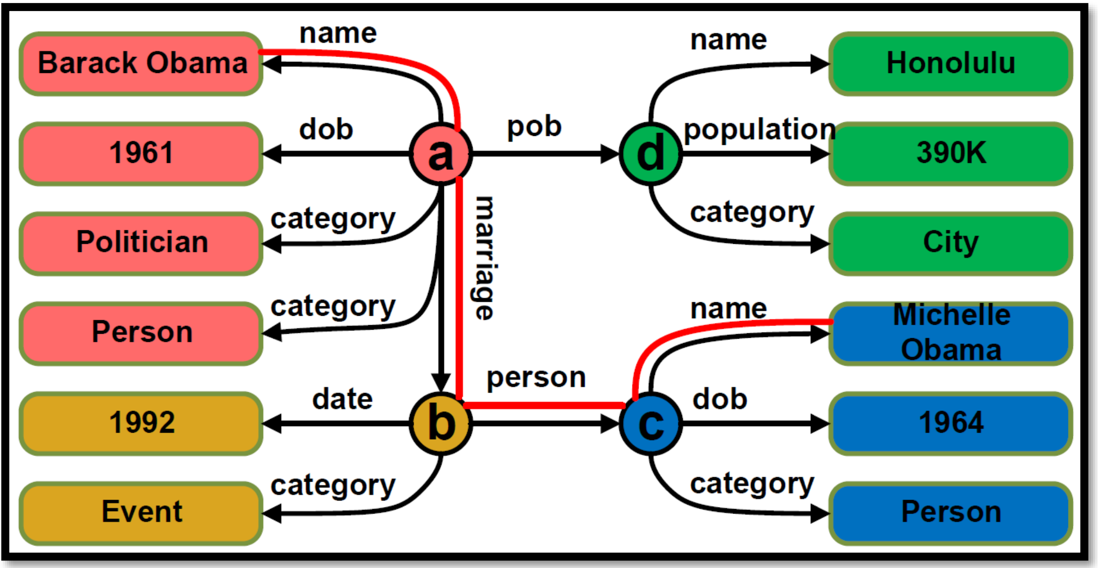


question2answer: predicate inference

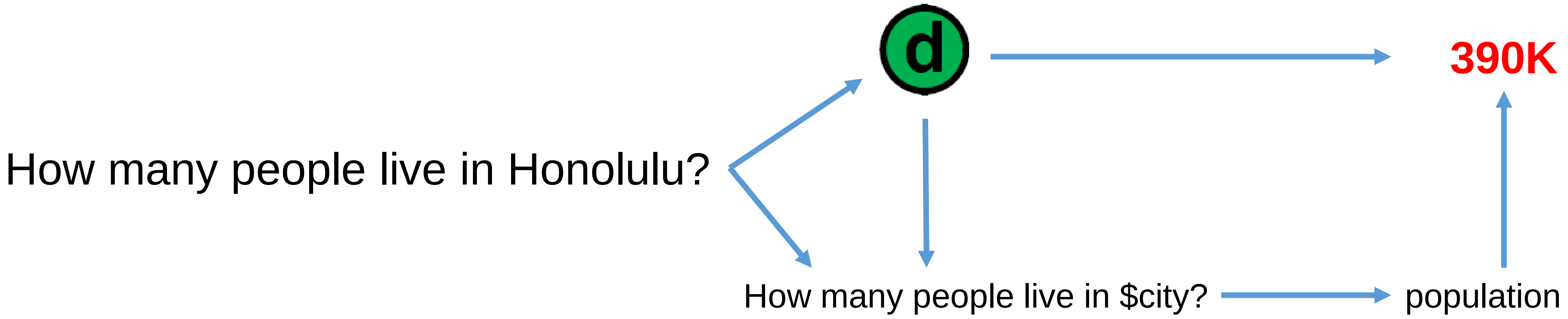
How many people live in Honolulu?

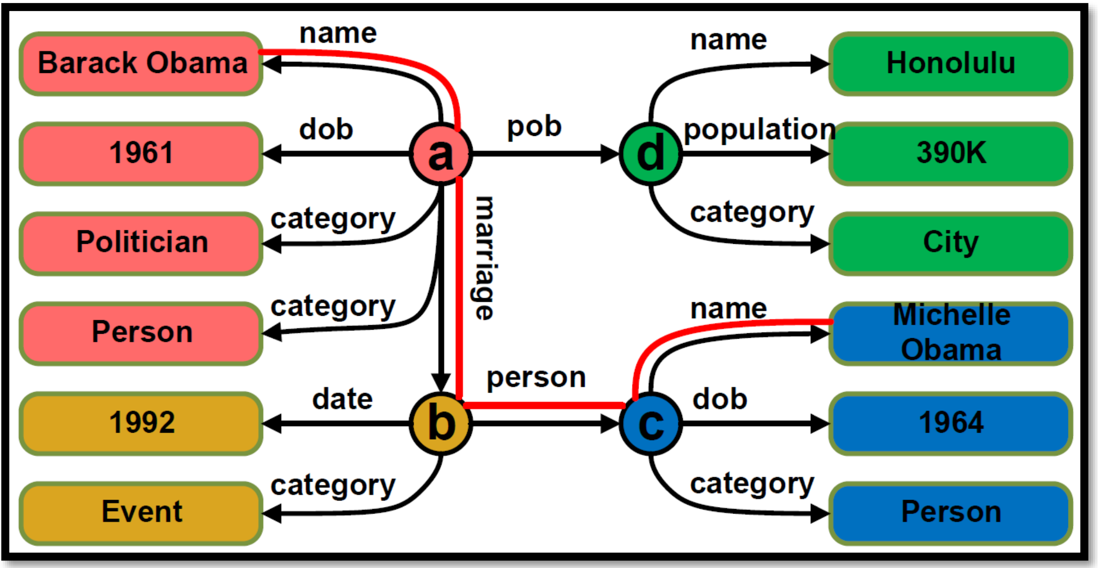
d

How many people live in $city? → population



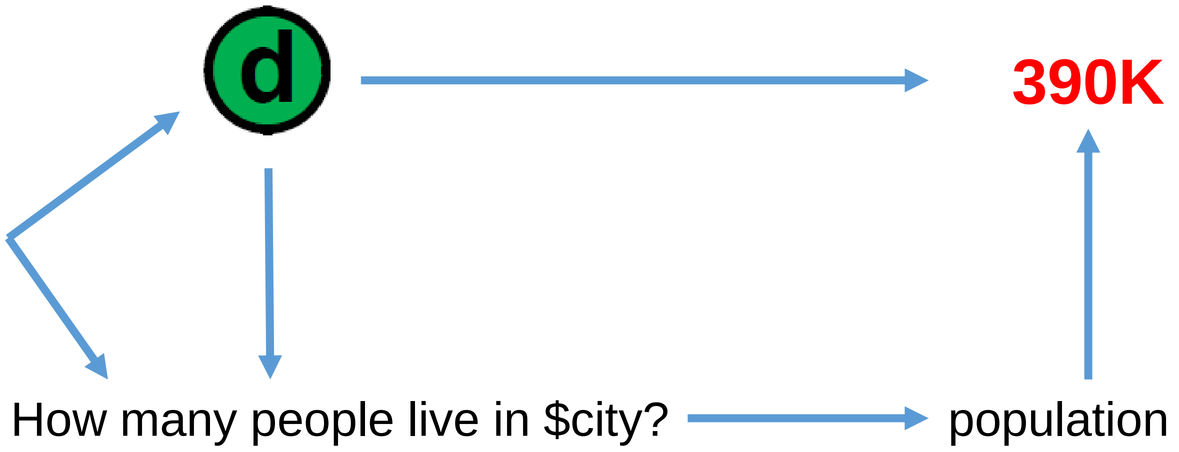
question2answer: value lookup





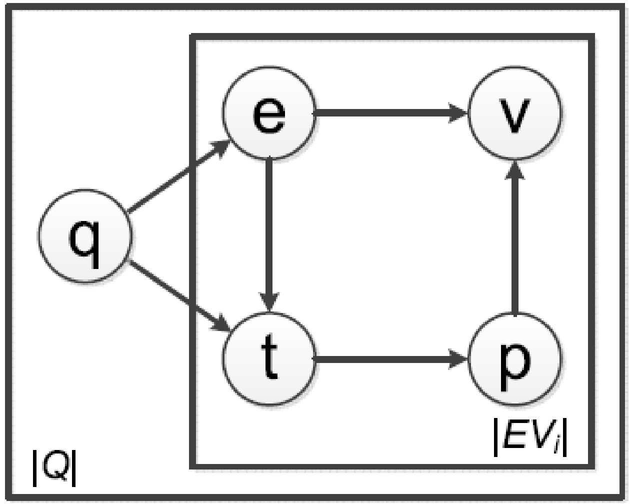
Probabilistic graph model

How many people live in Honolulu?



$$P(q,e,t,p,v) = P(q)P(e|q)P(t|e,q)P(p|t)p(v|e,p)$$

$$\arg\max_{v} \sum_{e,t,p} P(v|q,e,t,p)$$



Probability Inference

- Source
  - QA corpora (42M Yahoo! Answers)
  - Knowledge base such as Freebase
  - Probase(a large scale taxonomy)

- Directly estimated from data
  - Entity distribution P (e|q)
  - Template distribution P(t|q,e)
  - Value (answer) distribution P(v|e,p)

Question	Answer
When was Barack Obama born?	The politician was born in 1961.
When was Barack Obama born?	He was born in 1961.
How many people are there in Honolulu?	It’s 390K.

Yahoo! Answers QA pairs

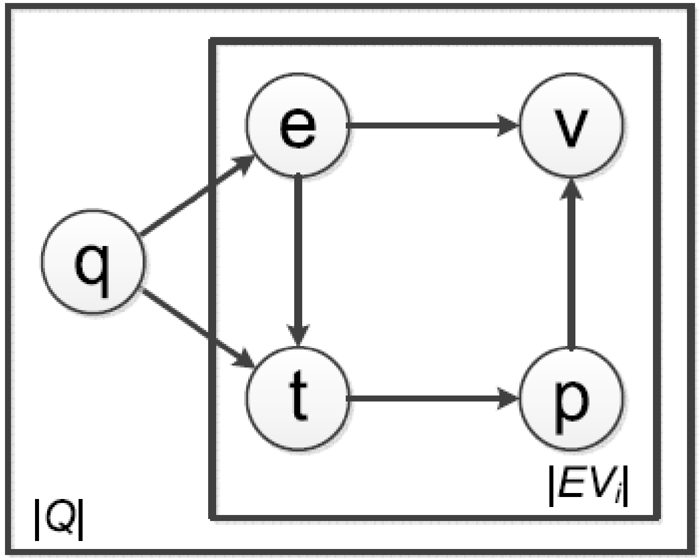
P(P|T) estimation

- We treat P(P|T) as parameters, and learn the parameter using maximum likelihood estimator, maximizing the **likelihood** of observing QA corpora
- An EM algorithm is used for parameter estimation

$$\hat{\theta} = \arg\max L(\theta)$$

$$L(\theta) = \sum_{i=1}^{m} \log P(x_i) = \sum_{i=1}^{m} \log P(q_i, e_i, v_i)$$

$$= \sum_{i=1}^{m} \log[\sum_{p \in P, t \in T} P(q_i)P(e_i|q_i)P(t|e_i, q_i)\boxed{\theta_{pt}}P(v_i|e_i, p)]$$



Answering complex questions

- When was Barack Obama's wife born?
  - (Who is) Barack Obama's wife?
  - When was Michelle Obama born?
- How to decompose the question into a series of binary questions?

$$\arg\max_{\mathcal{A}\in\mathbb{A}(q)} P(\mathcal{A})$$

- A binary question sequence is meaningful, only if each of the binary question is meaningful.

$$P(\mathcal{A}) = \prod_{\check{q}\in\mathcal{A}} P(\check{q})$$

- A dynamic programming (DP) algorithm is employed to find the optimal decomposition.

Experiments

	KBQA	Bootstrapping
Corpus	41M QA pairs	256M sentences
Templates	27,126,355	471,920
Predicates	2782	283
Templates per predicate	9751	4639

KBQA finds significantly more templates and predicates than its competitors despite that the corpus size of bootstrapping is larger.

$marriage \rightarrow person \rightarrow name$
Who is \$person marry to?
Who is \$person's husband?
What is \$person's wife's name?
Who is the husband of \$person?
Who is marry to \$person?

Concept based templates are meaningful

Experiments

	#pro	#ri	#par	R		R*		P	P*
Xser	42	26	7	0.52		0.66		0.62	0.79
APEQ	26	8	5	0.16		0.26		0.31	0.50
QAnswer	37	9	4	0.18		0.26		0.24	0.35
SemGraphQA	31	7	3	0.14		0.20		0.23	0.32
YodaQA	33	8	2	0.16		0.20		0.24	0.30
				R	R_{BFQ}	R*	R^*_{BFQ}		
KBQA+KBA	7	5	1	0.10	0.42	0.12	0.50	0.71	0.86
KBQA+Freebase	6	5	1	0.10	0.42	0.12	0.50	0.83	1.00
KBQA+DBpedia	8	8	0	0.16	0.67	0.16	0.67	**1.00**	**1.00**

Results over QALD-5. The results verify the effectiveness of KBQA over BFQs.

Experiments

Hybrid systems

- First KBQA
- If KBQA gives no reply, then baseline systems.

System	R	R*	P	P*
SWIP	0.15	0.17	0.71	0.81
KBQA+SWIP	0.33(+0.18)	0.35(+0.18)	0.87(+0.16)	0.92(+0.11)
CASIA	0.29	0.37	0.56	0.71
KBQA+CASIA	0.38(+0.09)	0.44(+0.07)	0.66(+0.10)	0.76(+0.05)
RTV	0.3	0.34	0.34	0.62
KBQA+RTV	0.39(+0.09)	0.42(+0.08)	0.66(+0.32)	0.71(+0.09)
gAnswer	0.32	0.43	0.42	0.57
KBQA+gAnswer	0.39(+0.07)	-	-	-
Intui2	0.28	0.32	0.28	0.32
KBQA+Intui2	0.39(+0.11)	0.41(+0.09)	0.39(+0.11)	0.41(+0.09)
Scalewelis	0.32	0.33	0.46	0.47
KBQA+Scalewelis	0.44(+0.12)	0.45(+0.12)	0.60(+0.14)	0.62(+0.15)

Results of hybrid systems on QALD-3 over DBpedia. The results verify the effectiveness of KBQA for a dataset that the BFQ is not a majority.

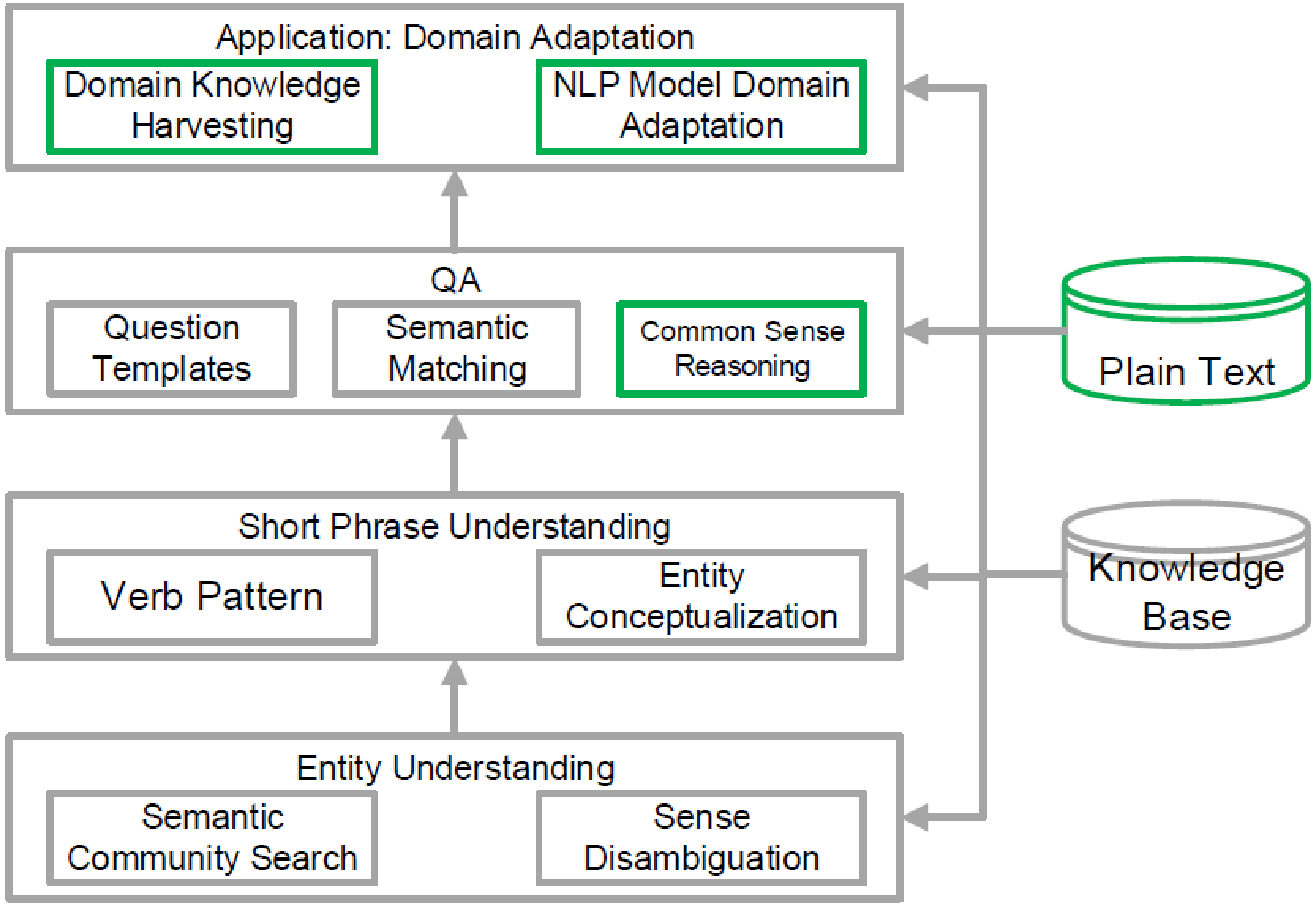
Conclusion

- Concept based templates are effective in representing questions’ semantic
- Template-predicate mapping is the key in building a QA system over KB
- Big QA corpora and KBs are good sources to learn the QA inference procedure
- A generative inference model is effective in modelling the question answering procedure

- We still have a long way to go in building a good QA system over knowledge bases in open domain.

Future works

- Domain adaptation
  - More applicable
- Common sense reasoning
  - Deeply understand the question
- Hybrid QA
  - Plain text + KB
  - Improve the effectiveness



Thank you!

Wechat QR code for our Chinese version system.

上财智能金融课题组招聘链接