Knowledge Authoring for Rule-based Reasoning

Tiantian Gao, Paul Fodor, Michael Kifer

Stony Brook University, Stony Brook, NY, USA

{tiagao,pfodor,kifer}@cs.stonybrook.edu

https://github.com/tiantiangao7/kalm

Knowledge authoring using CNL

- ***Knowledge authoring*** with Controlled Natural Languages (CNLs) enable domain experts to manually create formalized knowledge
- CNLs in a nutshell:
  - Emerged as a technology that bridges the gap between knowledge acquisition from text and logical reasoning.
  - Designed as a subset of English with restricted grammar and a set of interpretation rules that determine the unique meaning of each sentence.
  - Quite general and expressive, and requires little training to learn how to paraphrase natural language sentences into CNL.
  - Enable domain experts who lack the field experience in logic to write logical statements via CNL.
  - Representative systems include Attempto Controlled English (ACE), Processable English (PENG), BioQuery-CNL.
- Allan Institute for AI's project on knowledge extraction from science text book and supporting question answering

Issues with CNL Systems

They routinely fail to recognize when sentences have the same meaning but are expressed in different syntactical forms or using different language idioms.

- Example:
  1. Mary buys a car.
  2. Mary is the purchaser of a car.
  3. Mary makes a purchase of a car.

  Who purchases a car?

  Who is the buyer of a car?

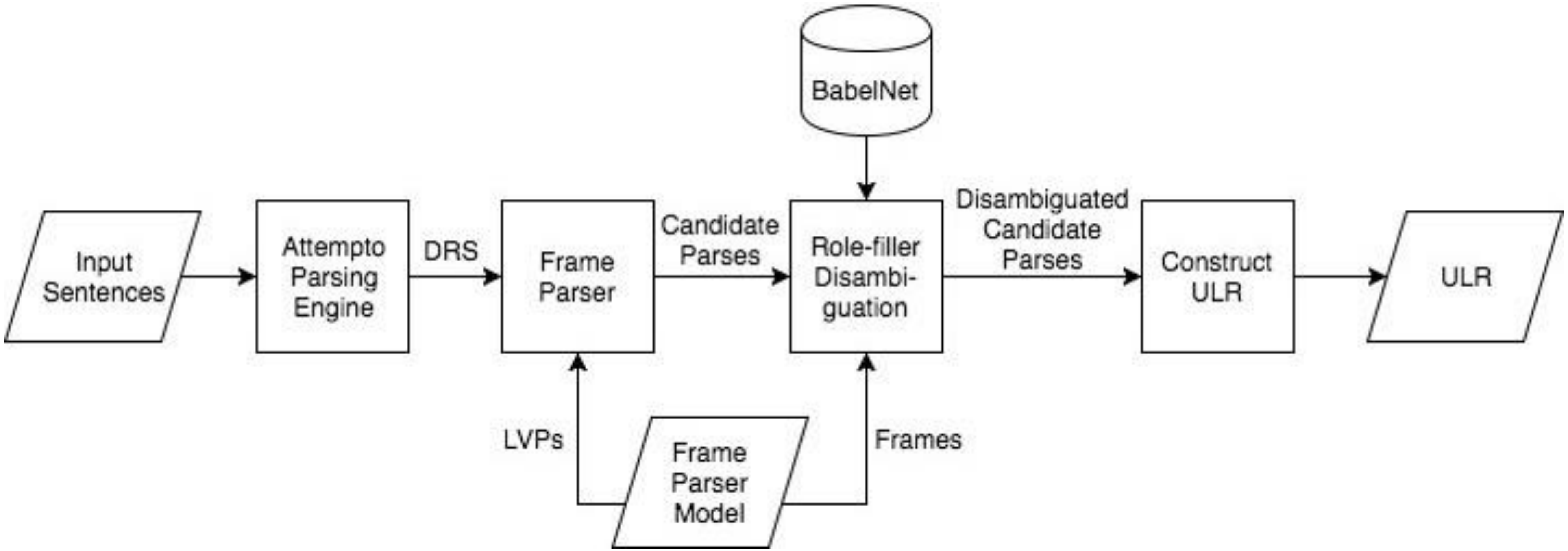
Goal of The Work

- Instead of tackling the problem of general text understanding, we developed Knowledge Authoring Logic Machine (KALM) that enables domain experts who may not be proficient in knowledge representation, to formulate actionable logic via CNL.
- Based on the acquired knowledge base, KALM provides an interface to run queries and perform complex reasoning tasks.

Contributions

- A formal, FrameNet-inspired ontology FrameOnt that formalizes FrameNet frames and integrates linguistic resources from BabelNet to represent the meaning of English sentences.
- An incrementally-learned semantic parser that disambiguates CNL sentences by mapping semantically equivalent sentences into the same FrameOnt frames and assigns them unique logical representation (ULR).
- The approach makes it possible to explain both why particular meanings are assigned and also why mistakes were made (so they can be fixed).
- The system achieves an unprecedented accuracy of 95.6% in standardizing the semantic parses extracted from CNL sentences and a superior accuracy of 95% in parsing queries.

Knowledge Authoring Logic Machine (KALM)



1. Syntactic Parsing
2. Frame-based Parsing
3. Role-Filler Disambiguation
4. Translating into ULR

Syntactic Parsing

- Attempto Parsing Engine (APE) translates CNL sentences into *Discourse Representation Structure* (DRS).
- DRS uses logical facts to represent the grammatical structure of the sentence.
- Example: A customer buys a watch.

```
object(A,customer,countable,na,eq,1)-1/2.     // customer entity
object(B,watch,countable,na,eq,1)-1/5.        // watch entity
predicate(D,buy,A,B)-1/3.                     // buy-event where A is the subject,
                                                 B is the object
```

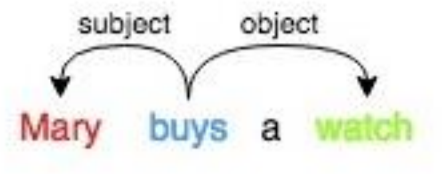
FrameNet

- FrameNet uses *frames*, described by frame *roles*, to capture the meaning of English sentences. Each frame is defined with a set of **lexical units (LUs)** and **valence patterns** that provide the guideline on how to extract an instance of a frame from a sentence.
- Example: **Commerce_Buy** frame describes purchases involving frame roles like buyers, sellers, goods, recipient, money, etc.

Commerce_Buy Frame

LU: buy.v
Buyer: subject
Goods: object

Valence Pattern



Valence Pattern

- FrameNet is not formal enough and contains only textual description of frames and valence patterns.

FrameOnt—The Logical Model of Frames

- We represent the frames and valence patterns in logic:
- Example of **Commerce_Buy** frame:

```
fp(Commerce_Buy,[
   role(Buyer,[bn:00014332n],[]),
   role(Seller,[bn:00053479n],[]),
   role(Goods,[bn:00006126n,
               bn:00021045n],[]),
   role(Recipient,[bn:00066495n],[]),
   role(Money,[bn:00017803n],[Currency])]).
```

1. We disambiguate each role with a set of BabelNet synsets that capture the meaning.
2. Constraints may be imposed on roles.

```
lvp(buy,v,Commerce_Buy, [
   pattern(Buyer,verb->subject,required),
   pattern(Goods,verb->object,required),
   pattern(Recipient,verb->dep[for],optnl),
   pattern(Money,verb->dep[for],optnl),
   pattern(Seller,verb->dep[from],optnl)]).
```

Grammatical pattern specifies the grammatical context that relates the lexical unit, a role, and a role-filler word.

Frame Construction

- For each frame, the System Engineer must provide the semantics for the roles by choosing the most appropriate synsets for the roles in question.
- The frames may not be restricted to FrameNet frames. The System Engineer can invent new frames based on specific requirements.
- The logical valence patterns are constructed in an automatic way by learning linguistic structures from annotated training sentences.
- Importantly, the learning process is INCREMENTAL and requires no retraining as more frames are added. KALM gets better with time & its maintenance is inexpensive.

Example of Frame-based Parsing

- Sentence: a customer buys a watch.

DRS

```
object(A,customer,countable,na,eq,1)-1/2.    // customer entity
object(B,watch,countable,na,eq,1)-1/5.       // watch entity
predicate(D,buy,A,B)-1/3.                    // buy-event where A is the subject,
                                                B is the object
```

LVP

```
lvp(buy,v,Commerce_Buy, [
   pattern(Buyer,verb->subject,required),
   pattern(Goods,verb->object,required),
   pattern(Recipient,verb->dep[for],optnl),
   pattern(Money,verb->dep[for],optnl),
   pattern(Seller,verb->dep[from],optnl)]).
```

Each grammatical pattern corresponds to a separate parsing rule (a prolog rule).

Issues in Frame-based Parsing

Example:

1. James releases a book.

2. The police releases a prisoner.

lvp(release,v,Publishing,

pattern(Author,verb->subject,required),

pattern(Work,verb->object, required)]).

lvp(release,v,Releasing_from_Custody,

pattern(Authorities,verb->subject,required),

pattern(Suspect,verb->object,required)]).

Frame(Publishing, Roles: Author = James , Work = book).

Frame(Releasing_from_Custody, Roles: Authorities = James , Prisoner = book).

Frame-based parsing is not enough

1. The parser may assign the wrong frames to the candidate parses.
2. Some candidate parses may be subsumed by others.
3. The parser may misidentify the roles for the words extracted from the CNL sentence, so wrong role-fillers may get associated with some of the frame's roles in the candidate parses.
4. A candidate parse extracts wrong role-fillers.

Role-Filler Disambiguation

The goal is to disambiguate different senses of the extracted role-fillers and find the best sense for each role-filler with respect to the frame roles in particular logical frames.

Example: Mary grows a macintosh.

Frame(Growing_Food, Roles: Grower = Mary , Food = macintosh).

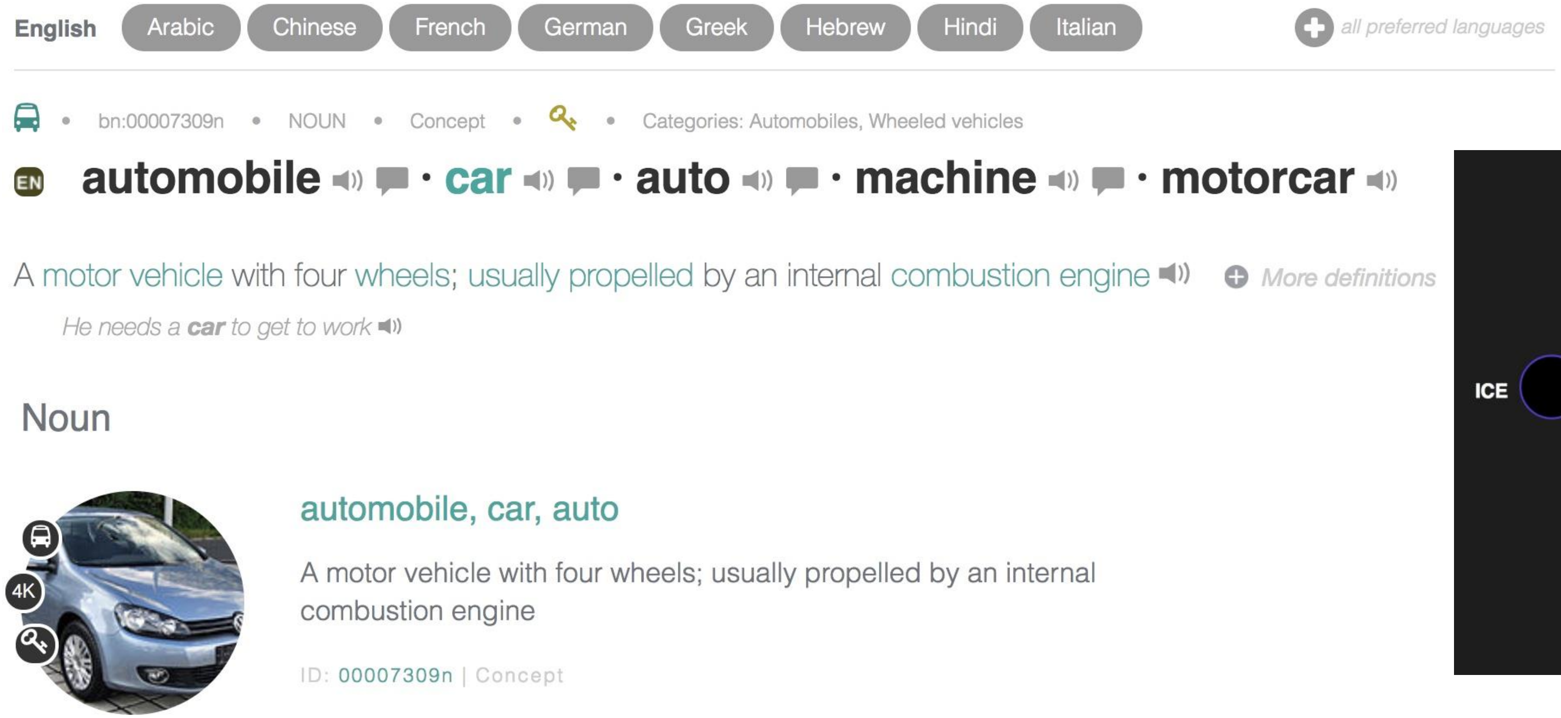
Macintosh has several meanings in BabelNet:

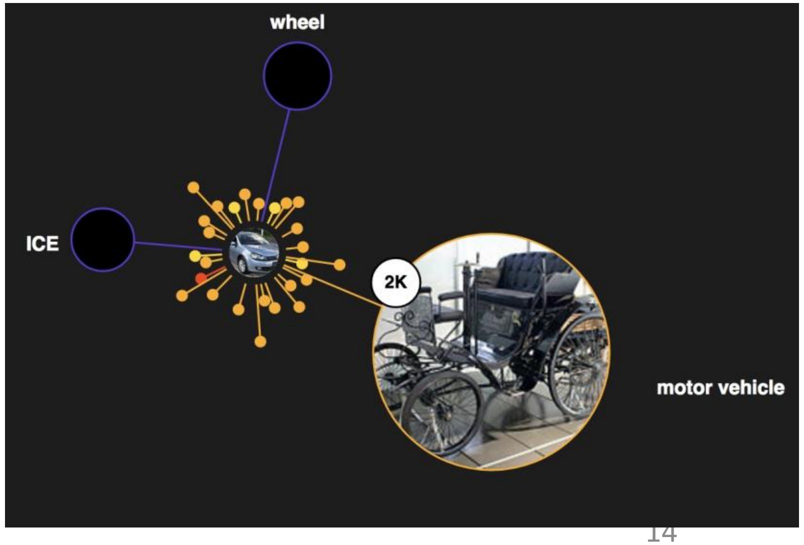
- an early-ripening apple (bn:00053981n)
- a computer sold by Apple Inc. (bn:21706136n)
- a kind of waterproof fabric (bn:00052580n).

Each disambiguated sense comes with a score indicating the semantic affinity between the role-filler synset and the role synset. Each individual score is combined into the score for the entire candidate parse. We rank the parses based on their scores, and remove the ones whose scores fall below a threshold.

BabelNet

- A multilingual knowledge base that contains a rich semantic network for words with their linguistic and semantic information.
- Constructed by integrating multiple well-known structured knowledge bases, such as WordNet, DBPedia, and Wikidata.





The Disambiguation Algorithm

1. Query BabelNet for each role-filler and gets a list of *candidate role-filler synsets* , which are BabelNet synsets for the role-filler words.
2. Performs a heuristic breadth-first search to find all semantic paths that start at each candidate role-filler synset and end at the role synset, or vice versa.
3. A heuristic scoring function assigns a score to each path, prunes the unpromising paths, and selects the path with the highest score.
4. Compute the geometric mean of all role-filler scores to score each disambiguated candidate parse and prune the ones with lower scores.

Scoring Function

- To score the semantic path between the candidate role-filler synset and the role synset, we consider three factors:
  1. The number of semantic links for each synset node
  2. The edge type and weight value
  3. The path length
- The scoring function encourages the paths with higher semantic connection numbers and edge weights, and to penalize the longer paths.
- Let n_1 be a role-filler synset node, n_l be a role synset node and $L = \{n_1, e_{12}, n_2, \ldots, n_l\}$ be a semantic path from n_1 to n_l.
- The semantic score of a path is computed as

$$score = \frac{\sum_{i=1}^{n-1} \sqrt{f_n(n_i)} \times f_w(e_{i,i+1})}{5^{\sum_{i=1}^{n-1} f_p(e_{i,i+1})}}$$

Explaining of Correct Parses

Example: Robin Li is a founder of Baidu.

Frame parses: Create_Organization(Creator = Robin Li, Org = Baidu)

People_by_Origin(Person = Robin Li, Origin = founder)

Being_Employed(Person = Robin Li, Position = founder)

Explanations:

Creator = Robin Li: Robin Li (bn:03307893n) $\xrightarrow{\text{hypernym}}$ a person who founds or establishes some institution (bn:00009631n)

Org = Baidu: Baidu (bn:00914124n) $\xrightarrow{\text{hypernym}}$ an institution created to conduct business (bn:00009631n) $\xrightarrow{\text{hypernym}}$ an organization (bn:00059480n)

Explaining of Erroneous Parses

Example: Mary works in Rockefeller Center.

Frame parses: Being_Employed(Employee = Mary, Field = Rockefeller Center)

Being_Employed(Employee = Mary, Place = Rockefeller Center)

Being_Employed(Employee = Mary, Employer = Rockefeller Center)

Explanations:

Field = Rockefeller Center

Rockefeller Center (bn:00897288n) $\xrightarrow{\text{hypernym}}$ a building structure (bn:00013722n) $\xrightarrow{\text{hypernym}}$ a discipline dealing with art and science (bn:00005105n) $\xrightarrow{\text{hypernym}}$ a branch of knowledge (bn:00007985n)

BabelNet contains noisy data: wrong synset assignment, incorrect semantic links, or missing semantic links.

Translating into ULR

ULR uses the predicates frame/2 and role/2 for representing instances of the frames and the roles. The predicates synset/2 and text/2 are used for the synset and the textual information.

Example: Mary buys a Camry for 15000 dollars.

```
frame(id_1, Commerce_buy).
role(id_1, Buyer, id_2).
role(id_1, Goods, id_3).
role(id_1, Money, id_4).
text(id_2, Mary).    synset(id_2, bn:00046516n). % Person synset
text(id_3, Camry).    synset(id_3, bn:03606178n). % Camry synset
text(id_4, '15000 dollars’). synset(id_4, bn:00024507n). % Currency synset
```

Evaluation

Dataset

1. Encoded a total of 50 logical frames.
2. Learned 213 logical valence patterns from 213 training sentences.
3. Used 28 additional tuning sentences to adjust the parameters of the scoring function used for role-filler disambiguation and to deal with noise in BabelNet.
4. Evaluated on 250 sentences (distinct from the training sentences) and verified whether the system returns the expected frames and disambiguates each role-filler correctly.

Sample Test Sentences

Commerce_Buy Frame:

- Mary buys a laptop.

Education Frame:

- John receives a bachelor degree from Cambridge University.

Travel Frame:

- Shinzo Abe visits China.

Being_Employed Frame:

- Kate is a teacher.

Performing Frame:

- Kate Winslet co-stars with Leonardo DiCaprio in Titanic.

Residence Frame:

- Susan is a resident of New York City.

Renting Frame:

- A student borrows a textbook from a library.

The test sentences we used are simple and common.

Results

FrSynC All frames & roles (semantic relations) identified correctly and all role-fillers disambiguated

FrC All frames and roles identified correctly

PFrC Some frames/roles identified, but some not

Wrong Some frames or roles are misidentified

KALM: 239 (95.6%) FrSynC, 248 (> 99%) FrC, and 2 (< 1%) Wrong.
SEMAFOR identified 236 sentences: 59 (25%) FrC, 44 (18.6%) PFrC, 133 (56.4%) Wrong.
SLING identified 233 sentences: 98 (42.1%) FrC, 63 (27%) PFrC, 72 (30.9%) Wrong.
Stanford CoreNLP identified 26 sentences: 14 (53.8%) FrC, 10 (38.5%) PrC, 2 (7.7%) Wrong.

None of the comparable systems disambiguated the extracted entities. Nor did they provide any explanations to the parsing results.

Question Answering

- Interrogative queries are sentences like:
  - Does Mary buy a car? // true/false query
  - Mary buys which car? // contain an output variable ***which*** – a placeholder for entities to be shown in the result
- Output variables: who, where, when, which, whose, what, how many/how much.
- DRS:

  query(A,which)-1/3 // use **query-predicate** to represent an output variable

  object(A,car,countable,na,eq,1)-1/4

  predicate(B,buy,named(Mary),A)-1/2

Issues with Question Answering

1. Query parse standardization
   - ➢Who is a buyer of a car?
   - ➢Who buys a car?
   - ➢Who makes a purchase of a car?
   - ➢A car is purchased by who?
2. Identify the type of entities each output variable holds and this type information will be used to justify the answers to the query.

   KB: Mary buys a Camry, a pen, and a book.

   Question: Mary buys ***which*** car?

   Answer: Camry.

Issues with Question Answering

3. The ACE query language is suffice to express the aforementioned type information in questions, the questions may get convoluted from the point of English grammar when the sentence contains more than two query words.

Example: Which person buys which car in which place at which price?

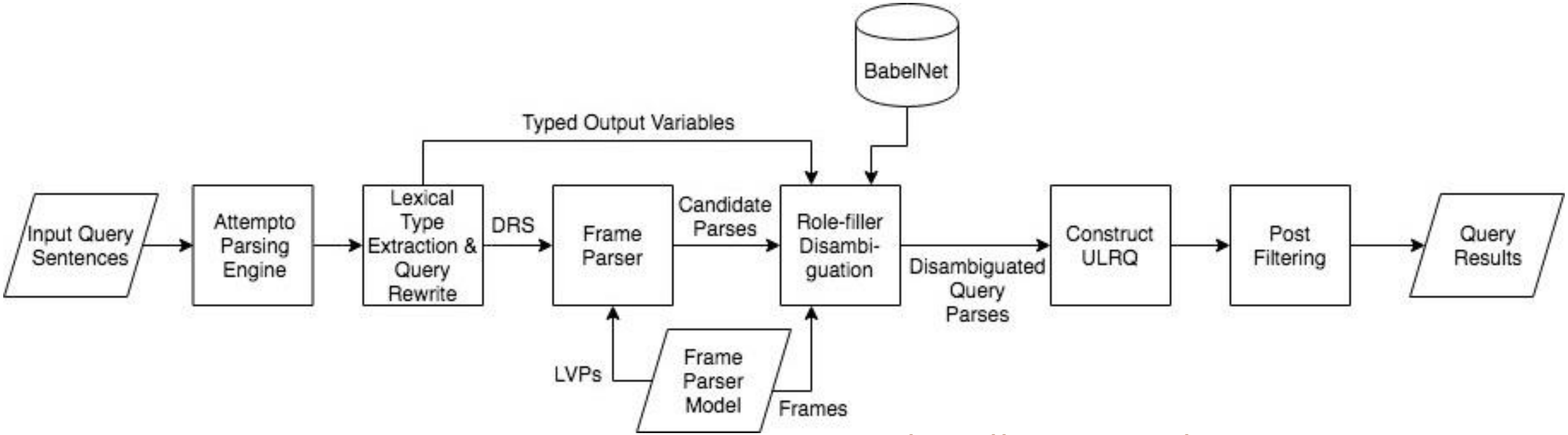
Examples of affirmative queries:

1. Mary buys a $car. //$-signed typed variables
2. A $person buys a $car in a $place at a $price.

4. The DRS for a query is not exactly the same as the DRS for a definite sentence, the frame-based parser cannot reuse the existing lvps to parse queries in the process of query parse standardization.

1. Invent new lvps for queries?
2. Rewriting the APE parse (DRS) for queries such that the lvps can be reused.

The Query Aspect of KALM



1. Syntactic Parsing
2. Lexical Typing & DRS Adaptation
3. Frame-based Parsing
4. Role-Filler Disambiguation
5. Translating into ULRQ
6. Type Filtering of Query Results

Translating into ULRQ

Suppose the KB contains :

- Mary buys a Camry for 15000 dollars.
- Mary pays 10000 dollars for a Jetta.
- Mary makes a purchase of a pen at a price of 2 dollars.
- Mary purchases a diamond with 30000 dollars.

One can query the KB as

- Who is a buyer of a $car?
- Who makes a purchase of which car?
- A $person purchases a $car.
- Who buys which car?

ULRQ:

```
?- frame(FrameV,'Commerce_Buy’),
    role(FrameV,'Buyer',BuyV),
    synset(BuyV,BuyerRoleFillerOutV),
    role(FrameV,'Goods',GoodV),
    synset(GoodV,GoodsRoleFillerOutV),
    check_type(BuyerRoleFillerOutV,bn:00046516n),
    check_type(GoodsRoleFillerOutV,bn:00007309n).
```

The query should return

Buyer = Mary;

Goods = Camry and Jetta.

How to rule out pen and diamond from the answer?

Type Filtering of Query Results

Type filtering checks the match between each candidate answer synset and the corresponding answer type synset by measuring their semantic affinity in BabelNet semantic network.

Methodology:

- Priority-based BabelNet path search
- Parallel computation for each candidate answer and answer type synset pair
- Caching BabelNet queries

Experiments

Dataset

1. Encoded a total of 50 logical frames.
2. We used 178 queries to check whether the system returns the expected frames, disambiguates role-filler words correctly, and identifies the types of the output variables .

Results

Sentences	Explanation
169 (94.94%)	all frames, roles & output variables are identified correctly; all role-filler words & variable types are disambiguated correctly
5 (2.81%)	all frames, roles and output variables are identified correctly, but some disambiguation mistakes
4 (2.25%)	some frames, roles or output variables are misidentified

Thank You!