Grounded Semantic Parsing of Claims and Questions

Pascual Martínez-Gómez Artificial Intelligence Research Center, AIST

Tokyo, Japan

January 22, 2018

Objective

Convert a claim/question into a SPARQL query.

Angelina Jolie's net worth is above 1.5 million USD.

$$\downarrow$$

```
ASK WHERE {
  dbr : AgenlinaJolie  dbp : NetWorth    ?x    .
  FILTER(?x > 1500000). }
```

Requirements:

1. Able to process claims and questions.
   - Possibly extend to paragraphs.
2. Must be independent to the language.
   - Most pressing: English, Japanese (FIJ) and French (?).
3. Easily extensible to different KBs and data stores.
4. Interpretable: Journalists may interact or inspect the process.

Approach

Modular pipeline (interpretable).

1. Identify **mentions** (e.g. *Agenlina Jolie*, *net worth*).
2. **Map** mentions **to KB** nodes and relations.
3. **Induce** a grammar that describes the **space** of SPARQL queries.
4. **Generate** SPARQL queries **in order** of plausibility (with scores).
5. **Execute** (and evaluate).

Identify mentions

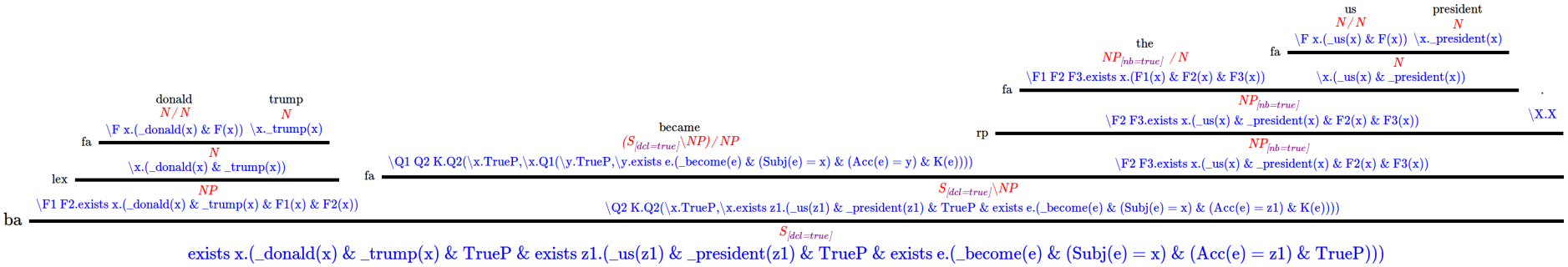
Depending on the depth of linguistic annotation:

1. Character sequence (e.g. *A, n, g, e, l, i, n, a, , J, o,...*).
2. Part of Speech tags (e.g. *Angelina:NNP*, *net:ADJ*, ...).
3. Syntax (e.g. *"Angelina Jolie":NP*, ...)
4. Semantics (e.g. $\exists x : \texttt{Person}, \texttt{angelina}(x) \wedge \texttt{jolie}(x)$).

Identify mentions

- Semantics.

Conclusion: donald trump became the us president .



Identify mentions

- Semantics.

Conclusion: emmanuel 's wife is brigitte

emmanuel
N
\x._emmanuel(x)
lex
NP
\F1 F2.exists x.(_emmanuel(x) & F1(x) & F2(x))

's
$(NP_{[nb=true]} / N)\backslash NP$
\Q F1 F2 F3.exists x.(Q(\w.TrueP,\y.Rel(x,y)) & F1(x) & F2(x) & F3(x))

ba
$NP_{[nb=true]} / N$
\F1 F2 F3.exists x.(exists z1.(_emmanuel(z1) & TrueP & Rel(x,z1)) & F1(x) & F2(x) & F3(x))

wife
N
\x._wife(x)

fa
$NP_{[nb=true]}$
\F2 F3.exists x.(exists z1.(_emmanuel(z1) & TrueP & Rel(x,z1)) & _wife(x) & F2(x) & F3(x))

is
$(S_{[dcl=true]}\backslash NP) / NP$
\Q1 Q2 K.Q2(\x.TrueP,\x.Q1(\y.TrueP,\y.(x = y)))

brigitte
N
\x._brigitte(x)
lex
NP
\F1 F2.exists x.(_brigitte(x) & F1(x) & F2(x))

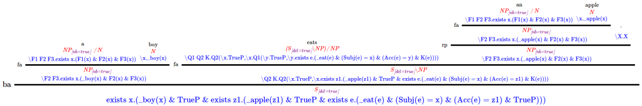
fa
$S_{[dcl=true]}\backslash NP$
\Q2 K.Q2(\x.TrueP,\x.exists z2.(_brigitte(z2) & TrueP & (x = z2)))

ba
$S_{[dcl=true]}$
exists x.(exists z1.(_emmanuel(z1) & TrueP & Rel(x,z1)) & _wife(x) & TrueP & exists z2.(_brigitte(z2) & TrueP & (x = z2)))

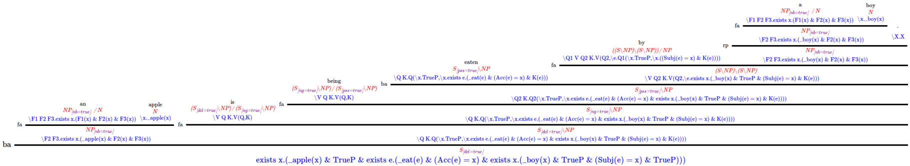
Identify mentions

- Semantics.

Premise 1: a boy eats an apple .

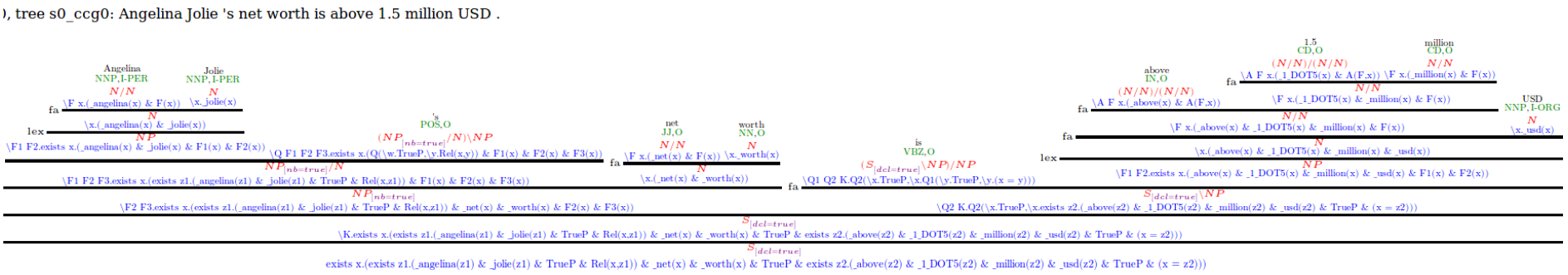


Conclusion: an apple is being eaten by a boy .



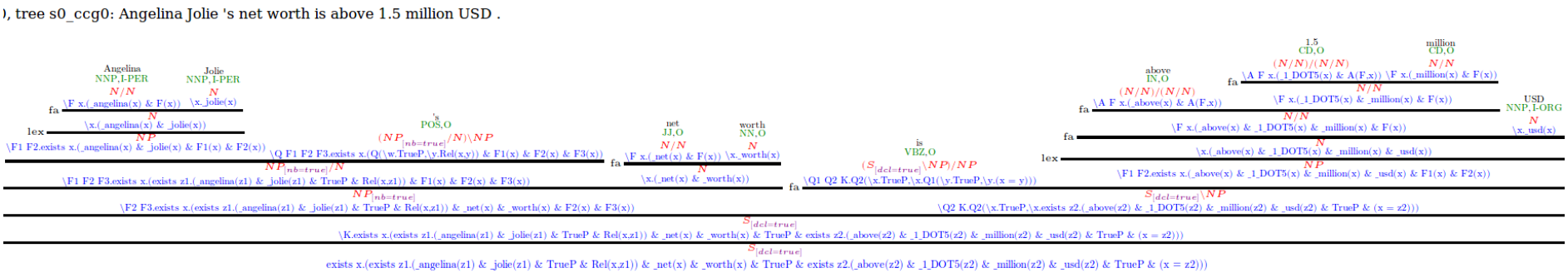
Identify mentions

- Semantics (e.g. $\exists x : \texttt{Person}, \texttt{angelina}(x) \wedge \texttt{jolie}(x) \ldots$).



Identify mentions

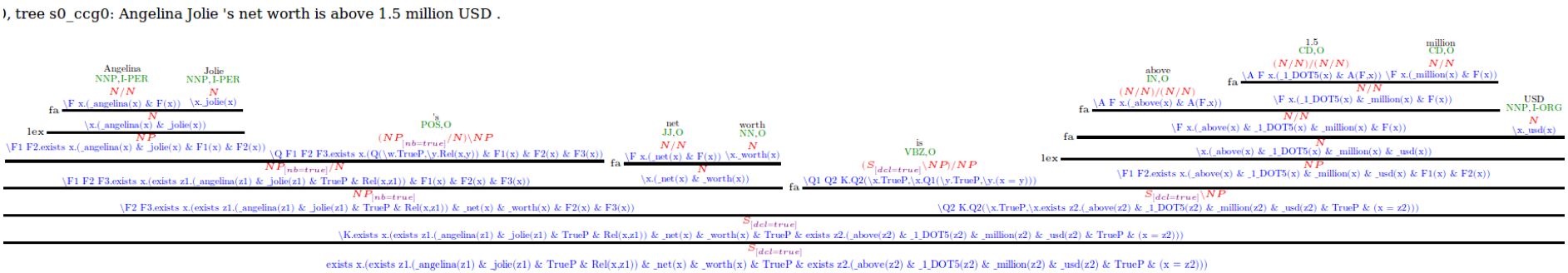
- Semantics (e.g. $\exists x : \texttt{Person}, \texttt{angelina}(x) \wedge \texttt{jolie}(x) \ldots$).



- Semantics for *net worth is above 1.5M USD* not accurate!
- $\exists z.\texttt{above}(z) \wedge 1.5(z) \wedge \texttt{million}(z) \wedge \texttt{usd}(z)$.

Identify mentions

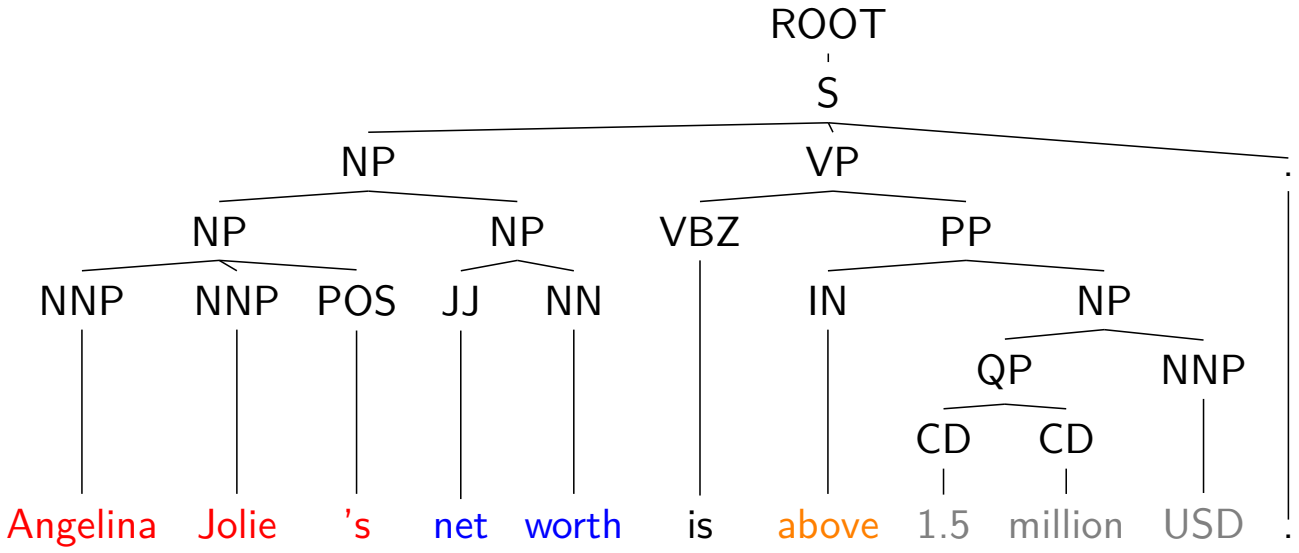
- Semantics (e.g. $\exists x : \texttt{Person}, \texttt{angelina}(x) \wedge \texttt{jolie}(x) \ldots$).



- Semantics for *net worth is above 1.5M USD* not accurate!
- $\exists z.\texttt{above}(z) \wedge \texttt{1.5}(z) \wedge \texttt{million}(z) \wedge \texttt{usd}(z)$.
- Errors tend to accumulate. Explore the use of less annotations.

Identify mentions

- Syntax



- Mentions: *Angelina Jolie's*, *net worth*, *Angelina Jolie's net worth*, *above*, *1.5 million USD*.
- It overgenerates mentions.
- But it is simple and may have good coverage.

Map mentions to KB nodes and relations.

Problems with traditional IR approaches:

- Symbolic nature: tf-idf sensitive to lexical variations.
- KB textual information is quite short (i.e. labels, aliases, names).
- Scoring functions are adhoc.

Map mentions to KB nodes and relations.

Problems with traditional IR approaches:

- Symbolic nature: tf-idf sensitive to lexical variations.
- KB textual information is quite short (i.e. labels, aliases, names).
- Scoring functions are adhoc.

Proposed solution:

- Learn a regressor $f_\theta : \mathcal{L} \times \mathcal{M} \rightarrow \mathcal{R}$.
  - $\mathcal{L}$ are labels of entities, relations or types.
  - $\mathcal{M}$ are mentions (as identified earlier).
- It generalizes easily to other KBs ("simply" retrain!).
- Make f_θ robust against spelling variations.
- Flexibility and adaptability.

Map mentions to KB nodes and relations.

Approach: metric learning.

1. Encode a label $l \in \mathcal{L}$ into a vector $\vec{l} \in \mathcal{R}^D$ with $\texttt{Enc}_{\theta'} : \mathcal{L} \rightarrow \mathcal{R}^D$.

Map mentions to KB nodes and relations.

Approach: metric learning.

1. Encode a label $l \in \mathcal{L}$ into a vector $\vec{l} \in \mathcal{R}^D$ with $\texttt{Enc}_{\theta'} : \mathcal{L} \rightarrow \mathcal{R}^D$.
2. Enc. mention $m \in \mathcal{M}$ into a vector $\vec{m} \in \mathcal{R}^D$ with $\texttt{Enc}_{\theta''} : \mathcal{M} \rightarrow \mathcal{R}^D$.
   - Encoding parameters θ' and θ'' might be equal.

Map mentions to KB nodes and relations.

Approach: metric learning.

1. Encode a label $l \in \mathcal{L}$ into a vector $\vec{l} \in \mathcal{R}^D$ with $\texttt{Enc}_{\theta'} : \mathcal{L} \rightarrow \mathcal{R}^D$.
2. Enc. mention $m \in \mathcal{M}$ into a vector $\vec{m} \in \mathcal{R}^D$ with $\texttt{Enc}_{\theta''} : \mathcal{M} \rightarrow \mathcal{R}^D$.
   - Encoding parameters θ' and θ'' might be equal.
3. Use vector similarity function between $\vec{l}$ and $\vec{m}$.
   - At the moment, I use only $\texttt{cosine}(\vec{l}, \vec{m}) = \frac{\vec{l}^\intercal \vec{m}}{||\vec{l}||_2 \cdot ||\vec{m}||_2}$.

Map mentions to KB nodes and relations.

Approach: metric learning.

1. Encode a label $l \in \mathcal{L}$ into a vector $\vec{l} \in \mathcal{R}^D$ with $\texttt{Enc}_{\theta'} : \mathcal{L} \rightarrow \mathcal{R}^D$.
2. Enc. mention $m \in \mathcal{M}$ into a vector $\vec{m} \in \mathcal{R}^D$ with $\texttt{Enc}_{\theta''} : \mathcal{M} \rightarrow \mathcal{R}^D$.
   - Encoding parameters θ' and θ'' might be equal.
3. Use vector similarity function between $\vec{l}$ and $\vec{m}$.
   - At the moment, I use only $\texttt{cosine}(\vec{l}, \vec{m}) = \frac{\vec{l}^\intercal \vec{m}}{||\vec{l}||_2 \cdot ||\vec{m}||_2}$.
4. Linking results are: $\arg\max_{l}^{k} \texttt{cosine}(\texttt{Enc}_{\theta'}(l), \texttt{Enc}_{\theta''}(m))$.

Map mentions to KB nodes and relations.

Approach: metric learning.

1. Encode a label $l \in \mathcal{L}$ into a vector $\vec{l} \in \mathcal{R}^D$ with $\texttt{Enc}_{\theta'} : \mathcal{L} \rightarrow \mathcal{R}^D$.
2. Enc. mention $m \in \mathcal{M}$ into a vector $\vec{m} \in \mathcal{R}^D$ with $\texttt{Enc}_{\theta''} : \mathcal{M} \rightarrow \mathcal{R}^D$.
   - Encoding parameters θ' and θ'' might be equal.
3. Use vector similarity function between $\vec{l}$ and $\vec{m}$.
   - At the moment, I use only $\texttt{cosine}(\vec{l}, \vec{m}) = \frac{\vec{l}^\intercal \vec{m}}{||\vec{l}||_2 \cdot ||\vec{m}||_2}$.
4. Linking results are: $\arg\max_{l}^{k} \texttt{cosine}(\texttt{Enc}_{\theta'}(l), \texttt{Enc}_{\theta''}(m))$.
5. Estimation:

$$\arg\max_{\theta', \theta''} \texttt{cosine}(\texttt{Enc}_{\theta'}(l_1), \texttt{Enc}_{\theta''}(l_1)) - \texttt{cosine}(\texttt{Enc}_{\theta'}(l_1), \texttt{Enc}_{\theta''}(l_2))$$

   - This is an autoencoder with noise contrastive estimation.
   - Uses positive and negative examples.

Map mentions to KB nodes and relations.

Examples (see gsemparse github repo):

Example 1: no misspellings:

1. Mention: *angelina jolie*.
2. Labels: Angelina jolie, Angelina Jolie Trapdoor Spider, Angelina Jolie Voight, Angelina Jolie cancer treatment, Angelina Jolie Pitt, Angelina Joli, Angelina Jolie Filmography, Anjelina Jolie.

Map mentions to KB nodes and relations.

Examples (see gsemparse github repo):

Example 1: no misspellings:

1. Mention: *angelina jolie*.
2. Labels: Angelina jolie, Angelina Jolie Trapdoor Spider, Angelina Jolie Voight, Angelina Jolie cancer treatment, Angelina Jolie Pitt, Angelina Joli, Angelina Jolie Filmography, Anjelina Jolie.

Example 2: misspellings (2 char substitutions, 1 char deletion):

1. Mention: *angeline yoli*
2. Labels: Angeline Jolie, Uncle Willie, Parmelia (lichen), Uriele Vitolo, Ding Lieyun, Earl of Loudon, Angeline Myra Keen, Angel Negro, Angeline Malik, Angeline of Marsciano.

Map mentions to KB nodes and relations.

Show the CNN over characters (on a whiteboard).

Map mentions to KB nodes and relations.

Resources for KB: `http://wiki.dbpedia.org/downloads-2016-10`

1. Infobox relations.
2. Infobox property definitions.
3. Ontology (types/classes and relations).
4. Labels (NIF)
5. Contexts (NIF)

Map mentions to KB nodes and relations.

Resources for KB: `http://wiki.dbpedia.org/downloads-2016-10`

1. Infobox relations.
2. Infobox property definitions.
3. Ontology (types/classes and relations).
4. Labels (NIF)
5. Contexts (NIF)

Resources for questions: Hobbit data - Scalable QA challenge.
`http://hobbitdata.informatik.uni-leipzig.de/SQAOC`

1. Many questions with annotations on mentions and SPARQL queries.
2. Hopefully easy to evaluate.

Induce a grammar that describes the space of SPARQL queries.

1. A Regular Tree Grammar (RTG) describes a tree language.
2. A big fragment of SPARQL can be represented with RTG.
3. A RTG is a compact representation of the language.

Example of RTG:

```
IDO -> (ID count IDN)
IDO -> (ID max IDN)
IDO -> IDN
IDN -> (ID PRED ENT)
IDN -> (ID !PRED ENT)
PRED -> pred1 | pred2
ENT -> ent1 | ent2
```

- There are only 7 productions.
- It generates 24 SPARQL queries.

Induce a grammar that describes the space of SPARQL queries.

- Each production may have a score (wRTG).
- Then, we can generate SPARQL queries in order of plausibility.

Example of RTG:

```
IDO -> (ID count IDN) # 0.1
IDO -> (ID max IDN) # 0.2
IDO -> IDN # 0.7
IDN -> (ID PRED ENT) # 0.8
IDN -> (ID !PRED ENT) # 0.2
PRED -> pred1 # 0.9 | pred2 # 0.1
ENT -> ent1 # 0.2 | ent2 # 0.8
```

- The highest scoring tree would be: `(ID pred1 ent2)`.
- It is important to estimate good parameters for these productions.
- Luckily, these methods are well studied and we *only* need to implement them.

Possible NLP objectives in this workshop

- Make a working end-to-end pipeline "claim $\rightarrow$ SPARQL query(ies)".
  - Pascual.
  - Share the code with WebClaimExplain team.

Possible NLP objectives in this workshop

- Make a working end-to-end pipeline "claim $\rightarrow$ SPARQL query(ies)".
  - Pascual.
  - Share the code with WebClaimExplain team.
- Implement routine to estimate weights of wRTG productions.
  - Pascual, Bevan (?).

Possible NLP objectives in this workshop

- Make a working end-to-end pipeline "claim $\rightarrow$ SPARQL query(ies)".
  - Pascual.
  - Share the code with WebClaimExplain team.
- Implement routine to estimate weights of wRTG productions.
  - Pascual, Bevan (?).
- Constrain the RTG using ontological information.
  - Pascual, Michaël (?) and Bevan (?).
- Other suggestions? (I am open to work on other necessary issues).