Natural Language to First Order Logic NatLog Meeting 9/29/2022

NatLog Group Meeting status

[The Google slides link to Natural Language to First Order logic-Sep-29-2022]

Changes to dataset for NL to FOL translation

For each hypothesis, FOLIO has multiple premises (1 to 3 or more)

So I selected all premises and corresponding FOL statements and used that as input for testing Encoder decoder model.

Results from Stanford Logic API for FOL

Using First Order Logic parser from StanfordNLP

Logic

In [ ]:
```
prem_fol = "Czech(miroslav) ∧ ChoralConductor(miroslav) ∧ Specialize(miroslav, renaissance) ∧ Specialize(miroslav, baroque)"
pred = read_expr(r'\N F x.(N(\G H.H(G(F)))(\u.x)(\u.u))')

miroslav = read_expr(r'miroslav')
baroque = read_expr(r'baroque')
renaissance = read_expr(r'renaissance')
Czech = read_expr(r'Czech(miroslav)')
ChoralConductor = read_expr(r'ChoralConductor(miroslav)')
Specialize1 = read_expr(r'Specialize(miroslav, renaissance)')
Specialize2 = read_expr(r'Specialize(miroslav, baroque)')
Czech(miroslav).simplify()
```

Out[ ]:
```
<ApplicationExpression Czech(miroslav,miroslav)>
```

In [ ]:
```
print((Czech(miroslav) & ChoralConductor(miroslav)).simplify())
```

```
(Czech(miroslav,miroslav) & ChoralConductor(miroslav,miroslav))
```

In [ ]:
```
print((Czech(miroslav) & ChoralConductor(miroslav) & Specialize1(miroslav, renaissance) & Specialize2(miroslav, baroque)).simplify())
```

```
(Czech(miroslav,miroslav) & ChoralConductor(miroslav,miroslav) & Specialize(miroslav,renaissance,miroslav,renaissance) & Specialize(mir
oslav,baroque,miroslav,baroque))
```

Results from Z3 Solver

Z3 solver with axioms and functions

Boolean Logic

First we define BoolSort functions. \ We define an Object of type DeclareSort. \ Then we define constants - which could remain constant in this world. \

Proof explanation

Miroslav is from Czech republic and is a ChoralConductor who specializes in renaissance genre and in baroque music. By Classic proof by refutation: We prove that if x is not a ChoralConductor will make this entire logical And operation invalid. \ So each of the Czech(x), ChoralConductor(x), Specialize(x, renaissance) , Specialize(x, baroque) will have to be true for the "Czech(miroslav) ∧ ChoralConductor(miroslav) ∧ Specialize(miroslav, renaissance) ∧ Specialize(miroslav, baroque)" to be true.

So z3 proves this by proof by refutation.

In [ ]:

```
prem_fol = "Czech(miroslav) ∧ ChoralConductor(miroslav) ∧ Specialize(miroslav, renaissance) ∧ Specialize(miroslav, baroque)"
Object = DeclareSort('Object')

Czech = Function('Czech', Object, BoolSort())
ChoralConductor = Function('ChoralConductor', Object, BoolSort())
Specialize = Function('Specialize', Object, Object, BoolSort())
miroslav = Const('miroslav', Object)
renaissance = Const('renaissance', Object)
baroque = Const('baroque', Object)
axioms1 = And(Czech(miroslav) , ChoralConductor(miroslav))
axioms2 = And(Czech(miroslav) , ChoralConductor(miroslav), Specialize(miroslav, renaissance), Specialize(miroslav, baroque))

s = Solver()
s.add(axioms1)
s.add(axioms2)
print(s.check()) # prints sat so axioms are coherent

print(s.model())

print(s.check()) # prints sat so this conjunction is satisfied
```

```
sat
[miroslav = Object!val!0,
 baroque = Object!val!2,
 renaissance = Object!val!1,
 ChoralConductor = [else -> True],
 Czech = [else -> True],
 Specialize = [else -> True]]
sat
```

Proof explanation

For All x if x is ChoralConductor then this implies x is musician. By Classic proof by refutation: We prove that if x is not a ChoralConductor then x is not a musician. \ The else (the negation) should have to be false. \ So we know true case valid.

In [ ]:

```
premise_fol = "∀x (ChoralConductor(x) → Musician(x))"
Object = DeclareSort('Object')

ChoralConductor = Function('ChoralConductor', Object, BoolSort())
Musician = Function('Musician', Object, BoolSort())
x = Const('x', Object)

axioms1 = ForAll(x, Implies( ChoralConductor(x), Musician(x)))

s = Solver()
s.add(axioms1)
print(s.check()) # prints sat so axioms are coherent

print(s.model())

print(s.check()) # prints sat so this conjunction is satisfied
```

```
sat
[ChoralConductor = [else -> False],
 Musician = [else -> False]]
sat
```

Using Encoder Decoder T5 model without Fine tuning

Test BLEU score : 0.00064

# of Test samples: 513

...	...	...
1021	BornIn(ailtonsilva, y1995) ∧ CommonlyKnownAs(a...	<pad> Natürliche Sprache bis zu First Order Lo...
1022	FootballPlayer(ailton) ∧ LoanedTo(ailton, braga)	<pad> Natürliche Sprache zu der Logik der Erst...
1023	Brazilian(ailtonsilva) ∧ Footballplayer(ailton...	<pad> Natürliche Sprache zu der Logik der erst...
1024	FootballClub(nautico) ∧ FootballClub(braga)	<pad> Natürliche Sprache zu First Order Logik:...
1025	FootballClub(fluminense)	<pad> Natürliche Sprache zu erster Ordnung Log...

1026 rows × 2 columns

```
df = results.to_pandas()
df.loc[0,'prediction']
```

```
'<pad> Natürliche Sprache zu erster Ordnung Logik: Wenn Menschen häufig in schulischen Tale
<pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><
```

```
test.column_names
```

```
['nl', 'fol']
```

Now evaluate the quality of translations using the BLEU metric:

```
from datasets import load_metric

metric = load_metric('sacrebleu')

for r in results:
    prediction = r['prediction']
    reference = [r['reference']]
    metric.add(prediction=prediction, reference=reference)

metric.compute()
```

```
{'score': 0.0006430186886559712,
 'counts': [297, 0, 0, 0],
 'totals': [229797, 228771, 227745, 226719],
 'precisions': [0.1292445071084479,
  0.00021855917052423601,
  0.00010977189400425915,
  5.513432927985745e-05],
 'bp': 1.0,
 'sys_len': 229797,
 'ref_len': 12947}
```

Baseline : Finetuned Encoder Decoder T5 model

Evaluation BLEU score : 46.69

# of Evaluation samples: 513

Test BLEU score: 0.011

```
metrics = train_result.metrics
metrics['train_samples'] = len(train_dataset)

trainer.log_metrics('train', metrics)
trainer.save_metrics('train', metrics)
trainer.save_state()
```

```
***** train metrics *****
  epoch                    =        3.0
  total_flos               =   117836GF
  train_loss               =      0.124
  train_runtime            = 0:04:52.00
  train_samples            =       5227
  train_samples_per_second =     53.702
  train_steps_per_second   =     13.428
```

Now evaluate:

```
# https://discuss.huggingface.co/t/evaluation-results-metric-during-training-is-different-from-the-evaluation-results-at-the-end/15401

metrics = trainer.evaluate(
    max_length=max_target_length,
    num_beams=num_beams,
    metric_key_prefix='eval',
)

metrics['eval_samples'] = len(eval_dataset)

trainer.log_metrics('eval', metrics)
trainer.save_metrics('eval', metrics)
```

```
***** Running Evaluation *****
  Num examples = 513
  Batch size = 4
[ 67/129 00:19 < 00:18, 3.32 it/s]
[129/129 00:35]
***** eval metrics *****
  epoch                   =        3.0
  eval_bleu               =    46.6949
  eval_loss               =     0.2484
  eval_runtime            = 0:00:35.96
  eval_samples            =        513
  eval_samples_per_second =     14.264
  eval_steps_per_second   =      3.587
```

Observations

Removes first letter of the function in FOL statement:

NL statement: *People either perform in school* talent shows often or are inactive and disinterested members of their community.

Original FOL statements:

∀x (Combine(Talent, Shows)(x) → *Not(disinterested*(x))) "Engaged"

∀x (TalentShows(x) ∨ Inactive(x))

Predicted FOL statement:

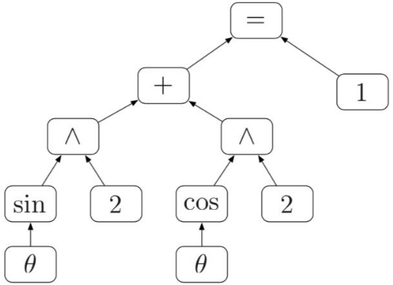
∀ x (AlentShows(x)→ ingrijireInactive(x))

∀ x (AlentShows(x)→ ingrijireInactive(x))

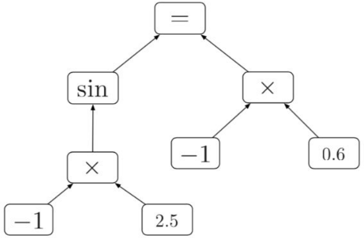
Next steps to explore

Neural Math

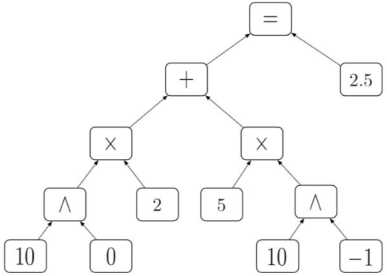
1. Weighted tree LSTMs for Math equations and evaluation for formal evaluation https://openreview.net/forum?id=Hksj2WWAW&noteId=Hksj2WWAW
2. Terminal symbols use one-hot encoding
3. Each equation LHS (Left Hand Side) and RHS (Right Hand Side) is represented as an LSTM. And weighted LSTM training and evaluation for values.



$\sin^2(\theta) + \cos^2(\theta) = 1$



$\sin(-2.5) = -0.6$



decimal tree for 2.5

Dependency Parsing

Dependency parsing for FOL and NL using Hierarchical Tree LSTMs:

https://aclanthology.org/Q16-1032/

Exploring Neural Models for Parsing Natural Language into First-Order Logic

1. https://arxiv.org/pdf/2002.06544.pdf
2. *"...Encoder decoder model by introducing a variable alignment mechanism that enables it to align variables across predicates in the predicted FOL. We further show the effectiveness of predicting the category of FOL entity - Unary, Binary, Variables and Scoped Entities, at each decoder step as an auxiliary task on improving the consistency of generated FOL. We perform rigorous evaluations and extensive ablations."*
3. Lambda Dependency-based Compositional Semantics
4. They used sequence to sequence transduction: https://www.cs.toronto.edu/~graves/seq_trans_slides.pdf
5.

Siamese Recurrent networks

Siamese recurrent networks (using LSTM + GRUs) :
https://arxiv.org/abs/1906.00180

Approach to explore similar to converting NL to SQL statements or SPARQL

Alignment between NL and FOL - approach to explore (similar to converting NL to SQL statements or SPARQL which have different symbols/commands)