Relation extraction

Bill MacCartney
CS224u
Stanford University

Data resources

Data resources

- The corpus
- The knowledge base (KB)

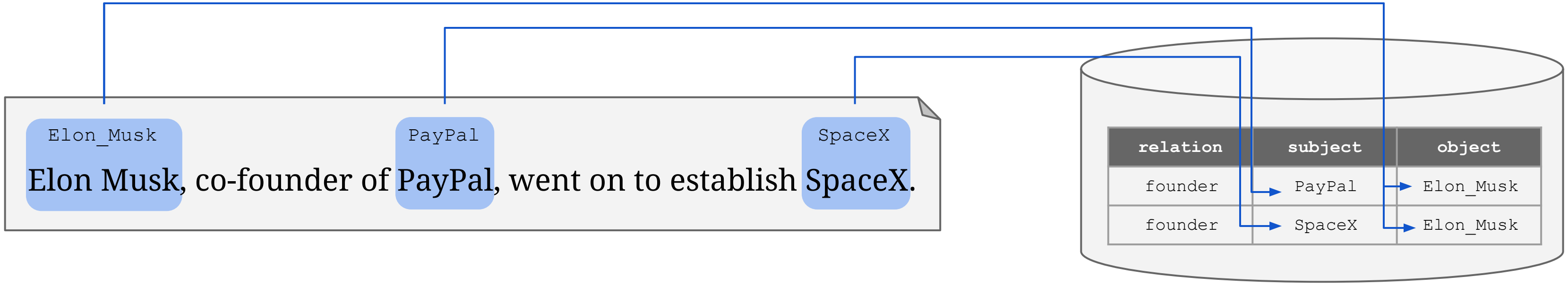
Overview

- ~~The task of relation extraction~~
- Data resources
- Problem formulation
- Evaluation
- Simple baselines
- Directions to explore

Data resources

The corpus

We need a corpus of sentences, each containing a pair of entities which have been annotated with entity resolutions so that they can be unambiguously linked to a knowledge base



Elon Musk, co-founder of PayPal, went on to establish SpaceX.

relation	subject	object
founder	PayPal	Elon_Musk
founder	SpaceX	Elon_Musk

Solution: the Wikilinks corpus (heavily adapted for our purposes)

Data resources

The corpus: the `Corpus` class

The `Corpus` class holds examples, and allows lookup by entity:

```
rel_ext_data_home = os.path.join('data', 'rel_ext_data')
corpus = rel_ext.Corpus(os.path.join(rel_ext_data_home,'corpus.tsv.gz'))
print('Read {0:,} examples'.format(len(corpus)))
```

```
Read 331,696 examples
```

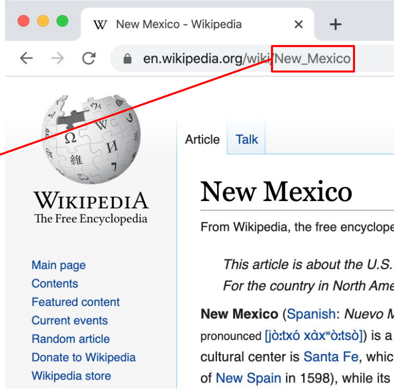
```
print(corpus.examples[1])
```

```
Example(entity_1='New_Mexico', entity_2='Arizona', left='to all Spanish-occupied lands . The horno has a
beehive shape and uses wood as the only heat source . The procedure still used in parts of', mention_1='New
Mexico', middle='and', mention_2='Arizona', right='is to build a fire inside the Horno and , when the proper
amount of time has passed , remove the embers and ashes and insert the', left_POS='to/TO all/DT
Spanish-occupied/JJ lands/NNS ./. The/DT horno/NN has/VBZ a/DT beehive/NN ... ')
```

Data resources

The corpus: the Example class

```
Example = namedtuple('Example',
    'entity_1, entity_2, left, mention_1, middle, mention_2, right, '
    'left_POS, mention_1_POS, middle_POS, mention_2_POS, right_POS')
```



New Mexico

From Wikipedia, the free encyclope

This article is about the U.S.
For the country in North Ame

New Mexico (Spanish: *Nuevo M*
pronounced [jòːtxó xàxʷòːtsò]) is a
cultural center is Santa Fe, whic
of New Spain in 1598), while its

	New_Mexico		Arizona	
	entity_1		entity_2	
The procedure still used in parts of	New Mexico	and	Arizona	is to build a fire inside the Horno ...
left	mention_1	middle	mention_2	right
The/DT procedure/NN still/RB used/VBN in/IN parts/NNS of/IN	New/NNP Mexico/NNP	and/CC	Arizona/NNP	is/VBZ to/TO build/VB a/DT fire/NN inside/IN the/DT Horno/NNP ...
left_POS	mention_1_POS	middle_POS	mention_2_POS	right_POS

Data resources

The corpus: most common entities

```
counter = Counter()
for example in corpus.examples:
    counter[example.entity_1] += 1
    counter[example.entity_2] += 1
print('The corpus contains {} entities'.format(len(counter)))
counts = sorted([(count, key) for key, count in counter.items()], reverse=True)
print('The most common entities are:')
for count, key in counts[:10]:
    print('{:10d} {}'.format(count, key))
```

```
The corpus contains 95909 entities
The most common entities are:
      8137 India
      5240 England
      4121 France
      4040 Germany
      3937 Australia
      3779 Canada
      3633 Italy
      3138 California
      2894 New_York_City
      2745 Pakistan
```

Data resources

The corpus: finding examples by entities

```
corpus.show_examples_for_pair('Elon_Musk', 'Tesla_Motors')
```

```
The first of 5 examples for Elon_Musk and Tesla_Motors is:
Example(entity_1='Elon_Musk', entity_2='Tesla_Motors', left='space for a while , here ’ s what might be
launching Americans into space in the next decade . Falcon 9 From sometimes Canadian , South African &
American', mention_1='Elon Musk', middle='‘ s company Space X . Musk is a PayPal alumni and',
mention_2='Tesla Motors', right='co-founder - remember that latter company name for future trivia questions
and/or a remake of Back to the Future . After several successful launches on their Falcon', ...)
```

```
corpus.show_examples_for_pair('Tesla_Motors', 'Elon_Musk')
```

```
The first of 2 examples for Tesla_Motors and Elon_Musk is:
Example(entity_1='Tesla_Motors', entity_2='Elon_Musk', left='their factory in Hethel . If you want to see
one in action , Robert Scoble got a ride in the first production model , driven by', mention_1='Tesla
Motors', middle='chairman', mention_2='Elon Musk', right='. Needless to say he got the whole thing on video
, and covers a lot of technical details about the car – this is the', ...)
```

The corpus: final observations

The Wikilinks corpus has some flaws. For example, it contains many near-dupes — an artefact of the document sampling methodology used to construct it.

One thing this corpus does *not* include is any annotation about relations. So, can’t be used for the fully-supervised approach.

To make headway, we need to connect the corpus to a KB!

Data resources

The knowledge base (KB)

Our KB is derived from Freebase (which shut down in 2016 😞).

It contains relational triples of the form (relation, subject, object).

```
(place_of_birth, Barack_Obama, Honolulu)
(has_spouse, Barack_Obama, Michelle_Obama)
(author, The_Audacity_of_Hope, Barack_Obama)
```

The relation is one of a handful of predefined constants.

The subject and object are entities identified by Wiki IDs.

Data resources

The knowledge base: the KB class

The KB class holds KBTriples, and allows lookup by entity:

```
kb = rel_ext.KB(os.path.join(rel_ext_data_home,'kb.tsv.gz'))

print('Read {0:,} KB triples'.format(len(kb)))
```

```
Read 45,884 KB triples
```

```
print(kb.kb_triples[0])
```

```
KBTriple(rel='contains', sbj='Brickfields', obj='Kuala_Lumpur_Sentral_railway_station')
```

The knowledge base: data exploration

```
len(kb.all_relations)
```

```
16
```

Data resources

The knowledge base: data exploration

```
for rel in kb.all_relations:
    print('{:12d} {}'.format(len(kb.get_triples_for_relation(rel)), rel))
```

```
        1702 adjoins
        2671 author
         522 capital
       18681 contains
        3947 film_performance
        1960 founders
         824 genre
        2563 has_sibling
        2994 has_spouse
        2542 is_a
        1598 nationality
        1586 parents
        1097 place_of_birth
         831 place_of_death
        1216 profession
        1150 worked_at
```

Data resources

The knowledge base: data exploration

```
for rel in kb.all_relations:
    print(tuple(kb.get_triples_for_relation(rel)[0]))
```

```
('adjoins', 'France', 'Spain')
('author', 'Uncle_Silas', 'Sheridan_Le_Fanu')
('capital', 'Panama', 'Panama_City')
('contains', 'Brickfields', 'Kuala_Lumpur_Sentral_railway_station')
('film_performance', 'Colin_Hanks', 'The_Great_Buck_Howard')
('founders', 'Lashkar-e-Taiba', 'Hafiz_Muhammad_Saeed')
('genre', '8_Simple_Rules', 'Sitcom')
('has_sibling', 'Ari_Emanuel', 'Rahm_Emanuel')
('has_spouse', 'Percy_Bysshe_Shelley', 'Mary_Shelley')
('is_a', 'Bhanu_Athaiya', 'Costume_designer')
('nationality', 'Ruben_Rausing', 'Sweden')
('parents', 'Rosanna_Davison', 'Chris_de_Burgh')
('place_of_birth', 'William_Penny_Brookes', 'Much_Wenlock')
('place_of_death', 'Jean_Drapeau', 'Montreal')
('profession', 'Rufus_Wainwright', 'Actor')
('worked_at', 'Brian_Greene', 'Columbia_University')
```

The knowledge base: data exploration

The `get_triples_for_entities()` method allows easy lookup:

```
kb.get_triples_for_entities('France', 'Germany')
```

```
[KBTriple(rel='adjoins', sbj='France', obj='Germany')]
```

```
kb.get_triples_for_entities('Germany', 'France')
```

```
[KBTriple(rel='adjoins', sbj='Germany', obj='France')]
```

Relations like `adjoins` are intuitively symmetric — but there's no guarantee that such inverse triples actually appear in the KB!

Data resources

The knowledge base: data exploration

Most relations are intuitively asymmetric:

```
kb.get_triples_for_entities('Tesla_Motors', 'Elon_Musk')
```

```
[KBTriple(rel='founders', sbj='Tesla_Motors', obj='Elon_Musk')]
```

```
kb.get_triples_for_entities('Elon_Musk', 'Tesla_Motors')
```

```
[KBTriple(rel='worked_at', sbj='Elon_Musk', obj='Tesla_Motors')]
```

So it can be the case that one relation holds between *X* and *Y*, and a different relation holds between *Y* and *X*.

The knowledge base: data exploration

An entity pair can belong to multiple relations.

```
kb.get_triples_for_entities('Cleopatra', 'Ptolemy_XIII_Theos_Philopator')
```

```
[KBTriple(rel='has_sibling', sbj='Cleopatra', obj='Ptolemy_XIII_Theos_Philopator'),
 KBTriple(rel='has_spouse', sbj='Cleopatra', obj='Ptolemy_XIII_Theos_Philopator')]
```

🙊

Data resources

The knowledge base: data exploration

```
counter = Counter()
for kbt in kb.kb_triples:
    counter[kbt.sbj] += 1
    counter[kbt.obj] += 1
print('The KB contains {:,} entities'.format(len(counter)))
counts = sorted([(count, key) for key, count in counter.items()], reverse=True)
print('The most common entities are:')
for count, key in counts[:10]:
    print('{:10d} {}'.format(count, key))
```

```
The KB contains 40,141 entities
The most common entities are:
       945 England
       786 India
       438 Italy
       414 France
       412 California
       400 Germany
       372 United_Kingdom
       366 Canada
       302 New_York_City
       247 New_York
```

Data resources

The knowledge base: data exploration

Note, no promise or expectation that the KB is *complete!*

In the KB:

```
(founders, Tesla_Motors, Elon_Musk)
(worked_at, Elon_Musk, Tesla_Motors)
(founders, SpaceX, Elon_Musk)
```

Not in the KB:

```
(worked_at, Elon_Musk, SpaceX)
```