Computational Semantics

PCL II, CL, UZH
May 25, 2016

Universität Zürich[UZH]

Useful list

Universität Zürich[UZH]

http://www.nltk.org/howto/:

- tag
- classify
- probability
- japanese
- wordnet
- data
- ccg
- logic
- …

Outline

1. Knowledge representation and automated reasoning
   - Logical connectives (or boolean operators)
   - Propositional logic
   - First-order logic
2. Wordnet and Word Sense Disambiguation
3. Introduction to word embeddings

Propositional logic

- Basic statements are represented with *propositional symbols*:
  - "John is chasing Mary" = P
  - "Mary is running" = Q

Propositional logic

Universität Zürich[UZH]

- Basic statements are represented with *propositional symbols*:
  - "John is chasing Mary" = P
  - "Mary is running" = Q
- They can be combined:
  - John is chasing Mary and she (Mary) is running: $P \,\&\, Q$
- Using the logical connectives (boolean operators)
  - not (-), and (&), or (|),
    therefore (→), equivalence (↔)

Logical Connectives

Universität Zürich[UZH]

- Negation (not) -
  -X = True if X is False (=the reverse value of X)
- Conjunction (and) &
  X & Y = True only if both X and Y are True
- Disjunction (or) |
  X | Y = True if at least one of X and Y is True
- Implication (if …, then ...) →
  X → Y = False only when X is True and Y is False
- Equivalence (if and only if) ↔
  X ↔ Y = True if X and Y are both True or False

Logical Connectives

Universität Zürich[UZH]

Conjunction (X & Y):

	X = False	X = True
Y = False	False	False
Y = True	False	True

Disjunction (X | Y):

	X = False	X = True
Y = False	False	True
Y = True	True	True

Implication (X → Y):

	X = False	X = True
Y = False	True	False
Y = True	True	True

Equivalence (X ↔ Y):

	X = False	X = True
Y = False	True	False
Y = True	False	True

Propositional Logic @ NLTK:

Universität Zürich[UZH]

```
# NLTK can process logical expressions into subclasses of Expression object

>>> from nltk.sem.logic import LogicParser
>>> lp = LogicParser()
>>> lp.parse('-(P & Q)')
<NegatedExpression -(P & Q)>

>>> lp.parse('P & Q')
<AndExpression (P & Q)>

>>> lp.parse('P | (R -> Q)')
<OrExpression (P | (R -> Q))>

>>> lp.parse('P <-> -- P')
<IffExpression (P <-> --P)>
```

Valuation

Universität Zürich[UZH]

- Assigning values:

```
>>> val = nltk.Valuation([('P', True), ('Q', True), ('R', False)])
>>> val['P']
True
```

- Evaluating expressions:

```
>>> dom = set([])
>>> g = nltk.Assignment(dom)
>>> m = nltk.Model(dom, val)

>>> print m.evaluate('(P & Q)', g)
True
>>> print m.evaluate('-(P & Q)', g)
False
>>> print m.evaluate('(P | R)', g)
True
```

Arguments

Universität Zürich[UZH]

- Starting with some assumptions
  - propositional symbol expressions, e.g.
    - P (fish live in water)
    - Q (fish can fly)
    - $P \rightarrow -Q$
- Can we reach a conclusion?
  - e.g. $-Q$
- Argument: $[A_1, ..., A_n] / C$, where $A_1, ..., A_n$ are **assumptions**

An argument is **valid** if there is no possible situation in which its premises are all true and its conclusion is not true.

First-order logic

- Propositions are analyzed into predicates and arguments
  - read(John)
- combined with boolean operators
  - read(John, book) & falling_asleep(John)

First-order logic @ NLTK:

Universität Zürich[UZH]

```
>>> from nltk.sem.logic import LogicParser
>>> lp = LogicParser()

>>> p = lp.parse('read(john)')

>>> p.argument
<ConstantExpression john>

>>> p.function
<ConstantExpression read>
```

Quantifiers

Universität Zürich[UZH]

- Used with variables
- Existence quantifier: ∃
  - ∃*x* means: for some *x* it is true that...
  - e.g. ∃*x*.(person(x) ^ like(*x*, waffles))
  - meaning: there are some that like waffles
  - NLTK:

```
'exists x.(person(x) & like(x, waffles))'
```

Quantifiers

Universität Zürich[UZH]

- Used with variables
- Universal quantifier: $\forall$
  - $\forall x$ means: for all x…
  - e.g. $\forall x.(person(x) \rightarrow \text{lie}(x))$
  - meaning: everybody lies
  - NLTK: `'all x.(person(x) → lie(x))'`

Outline

Universität Zürich[UZH]

1. knowledge representation and automated reasoning
   - Logical connectives (or boolean operators)
   - Propositional logic
   - First-order logic
2. Wordnet and Word Sense Disambiguation
3. Introduction to word embeddings

Lexical semantics

Universität Zürich[UZH]

Words meanings and their relations

- word meanings/senses
  - bank → financial institution, river shore,…
  - crane → construction machine, bird,…
  - …
- word relations
  - synonyms/antonyms, hyponyms/hyperonyms, …
  - e.g. WordNet
    - http://www.nltk.org/howto/wordnet.html

WordNet @ NLTK

Universität Zürich[UZH]

- Lemma → senses:

```
from nltk.corpus import wordnet as wn

wn.synsets('bank')
[Synset('bank.n.01'), Synset('depository_financial_institution.n.01'),
Synset('bank.v.01'),...]
```

- Senses for a given PoS-tag only:

```
wn.synsets('bank', pos=wn.VERB)
[Synset('bank.v.01'), Synset('bank.v.02'),...]
```

- Sense definitions:

```
print wn.synsets('bank')[1].definition()
Synset('depository_financial_institution.n.01') a financial institution
that accepts deposits and channels the money into lending activities
```

WordNet @ NLTK

Universität Zürich[UZH]

- Sense → lemmas:

```
wn.synsets('bank')[1].lemmas()
[Lemma('depository_financial_institution.n.01.
depository_financial_institution'), Lemma
('depository_financial_institution.n.01.bank'), Lemma
('depository_financial_institution.n.01.banking_concern'), Lemma
('depository_financial_institution.n.01.banking_company')]

wn.synsets('bank')[1].lemma_names()
['depository_financial_institution'), 'bank', 'banking_concern'),
'banking_company']
```

WordNet @ NLTK

Universität Zürich[UZH]

- Other languages:

```
wn.langs()
[u'als', u'arb', u'cmn', u'dan', u'eng', u'fas', u'fin', u'fra', u'fre',
u'heb', u'ita', u'jpn', u'cat', u'eus', u'glg', u'spa', u'ind', u'zsm',
u'nno', u'nob', u'pol', u'por', u'tha']

wn.synsets('pane')
[Synset('pane.n.01'), Synset('paneling.n.01'), Synset('acid.n.02')]

wn.synsets('pane', lang='ita')
[Synset('support.n.06'), Synset('cake.n.01'), Synset('bread.n.01'), Synset
('loaf_of_bread.n.01')]

wn.synsets('fromage')
[]

wn.synsets('fromage', lang='fra')
[Synset('cheese.v.02'), Synset('cheese.v.01'), Synset('cheese.n.01'),
Synset('tall_mallow.n.01')]
```

WordNet @ NLTK

Universität Zürich[UZH]

- Other relations:

```
dog = wn.synset('dog.n.01')
dog.hypernyms()
[Synset('canine.n.02'), Synset('domestic_animal.n.01')]

dog.hyponyms()
[Synset('basenji.n.01'), Synset('corgi.n.01'), Synset('cur.n.01'), Synset
('dalmatian.n.02'), ...]

dog.member_holonyms()
[Synset('canis.n.01'), Synset('pack.n.06')]

dog.root_hypernyms()
[Synset('entity.n.01')]
```

WordNet @ NLTK

Universität Zürich[UZH]

- Other relations:

```
good = wn.synset('good.a.01')

good.antonyms()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'Synset' object has no attribute 'antonyms'

good.lemmas()
[Lemma('good.a.01.good')]

good.lemmas()[0].antonyms()
[Lemma('bad.a.01.bad')]
```

WordNet @ NLTK

Universität Zürich[UZH]

- Similarity
  - based on path between two senses

```
dog = wn.synset('dog.n.01')
cat = wn.synset('cat.n.01')

print dog.path_similarity(cat)
0.2
print wn.path_similarity(dog, cat)
0.2

for s in wn.synsets('wolf'):
    print dog.path_similarity(s)
0.333333333333
0.1
0.1
0.111111111111
...
```

WordNet @ NLTK

Universität Zürich[UZH]

- Lowest common hypernym

```
print dog.lowest_common_hypernyms(cat)
[Synset('carnivore.n.01')]

print dog.lowest_common_hypernyms(wn.synset('wolf.n.01'))
[Synset('canine.n.02')]

print dog.lowest_common_hypernyms(wn.synset('wolf.n.03'))
[Synset('organism.n.01')]
```

WordNet @ NLTK

Universität Zürich[UZH]

- Lowest common hypernym

```
print dog.lowest_common_hypernyms(cat)
[Synset('carnivore.n.01')]

print dog.lowest_common_hypernyms(wn.synset('wolf.n.01'))
[Synset('canine.n.02')]

print dog.lowest_common_hypernyms(wn.synset('wolf.n.03'))
[Synset('organism.n.01')]

print wn.synset('wolf.n.03').definition()
German classical scholar who claimed that the Iliad and Odyssey were
composed by several authors (1759-1824)
```

WordNet @ NLTK

Universität Zürich[UZH]

- Lowest common hypernym

```
print dog.lowest_common_hypernyms(cat)
[Synset('carnivore.n.01')]

print dog.lowest_common_hypernyms(wn.synset('wolf.n.01'))
[Synset('canine.n.02')]

print dog.lowest_common_hypernyms(wn.synset('wolf.n.03'))
[Synset('organism.n.01')]

print wn.synset('wolf.n.03').definition()
German classical scholar who claimed that the Iliad and Odyssey were
composed by several authors (1759-1824)

print dog.lowest_common_hypernyms(wn.synset('love.n.01'))
[Synset('entity.n.01')]

wn.synset('dog.n.01').path_similarity(wn.synset('love.n.01'))
0.06666666666666667
```

Word sense disambiguation

Universität Zürich[UZH]

- another computational semantics task
- given a sentence containing words with multiple meanings, determine which meanings are used
  - e.g. “Today I am going to rob a bank”
  - bank → bank.n.01 / depository_financial_institution.n.01 / etc.?
- different from tagging: how?

Word sense disambiguation

Universität Zürich[UZH]

- another computational semantics task
- given a sentence containing words with multiple meanings, determine which meanings are used
  - e.g. “Today I am going to rob a bank”
  - bank → bank.n.01 / depository_financial_institution.n.01 / etc.?
- different from tagging: how?

  different number of senses per word

Word sense disambiguation

Can the word senses be labeled independently?

Word sense disambiguation

Universität Zürich[UZH]

Can the word senses be labeled independently?

- words in sentence and their sentence are not independent
- choice of one sense can influence another:

```
time
[Synset('time.n.01'), Synset('fourth_dimension.n.01'), Synset('clock.v.
01'),...]

flies
[Synset('flies.n.01'), Synset('fly.n.01'), Synset('fly.v.01'),...]

like
[Synset('like.n.01'), Synset('like.n.02'), Synset('wish.v.02'), Synset
('like.v.02'), Synset('comparable.s.02'),...]
```

Word sense disambiguation

Universität Zürich[UZH]

Can the word senses be labeled independently?

- words in sentence and their sentence are not independent
- choice of one sense can influence another:

```
time
[Synset('time.n.01'), Synset('fourth_dimension.n.01'), Synset('clock.v.
01'),...]

flies
[Synset('flies.n.01'), Synset('fly.n.01'), Synset('fly.v.01'),...]

like
[Synset('like.n.01'), Synset('like.n.02'), Synset('wish.v.02'), Synset
('like.v.02'), Synset('comparable.s.02'),...]
```

- in practice this can be ignored

Word sense disambiguation

Universität Zürich[UZH]

Approaches:

- **knowledge-based**: using a dictionary, lexicon...
  - e.g. the Lesk algorithm
- **corpus-based**: using a sense-disambiguated corpus
  - e.g. transformation-based learning

Word Sense Disambiguation
Lesk Algorithm

1. go through the definitions of all senses of the word
2. check the overlap between the words in the definition and in the context
3. select the sense with the highest overlap
    - in case none of the senses have an overlap, select the most frequent sense of the word (first in WordNet)

- simplified version: perform the algorithm independently for each word in sentence

Word Sense Disambiguation
Lesk Algorithm

Universität Zürich[UZH]

```
from nltk.wsd import lesk

from nltk import word_tokenize

sent = word_tokenize("I went to the bank to deposit money." )
word = "bank"
pos = "n"

print(lesk(sent, word, pos))
Synset('depository_financial_institution.n.01')
```

Word Sense Disambiguation
Transformation-based Learning

Universität Zürich[UZH]

1. learn to disambiguate from a sense-annotated corpus
2. tag the corpus with a baseline tagger (e.g. most frequent sense)
3. learn transformations

   e.g.

   if current_sense = 'bank.n.01' (river bank)

   and sentence includes 'money'

   then change sense to 'depository_financial_institution.n.01'
4. when doing WSD, apply the baseline and then the learned transformations

Outline

Universität Zürich[UZH]

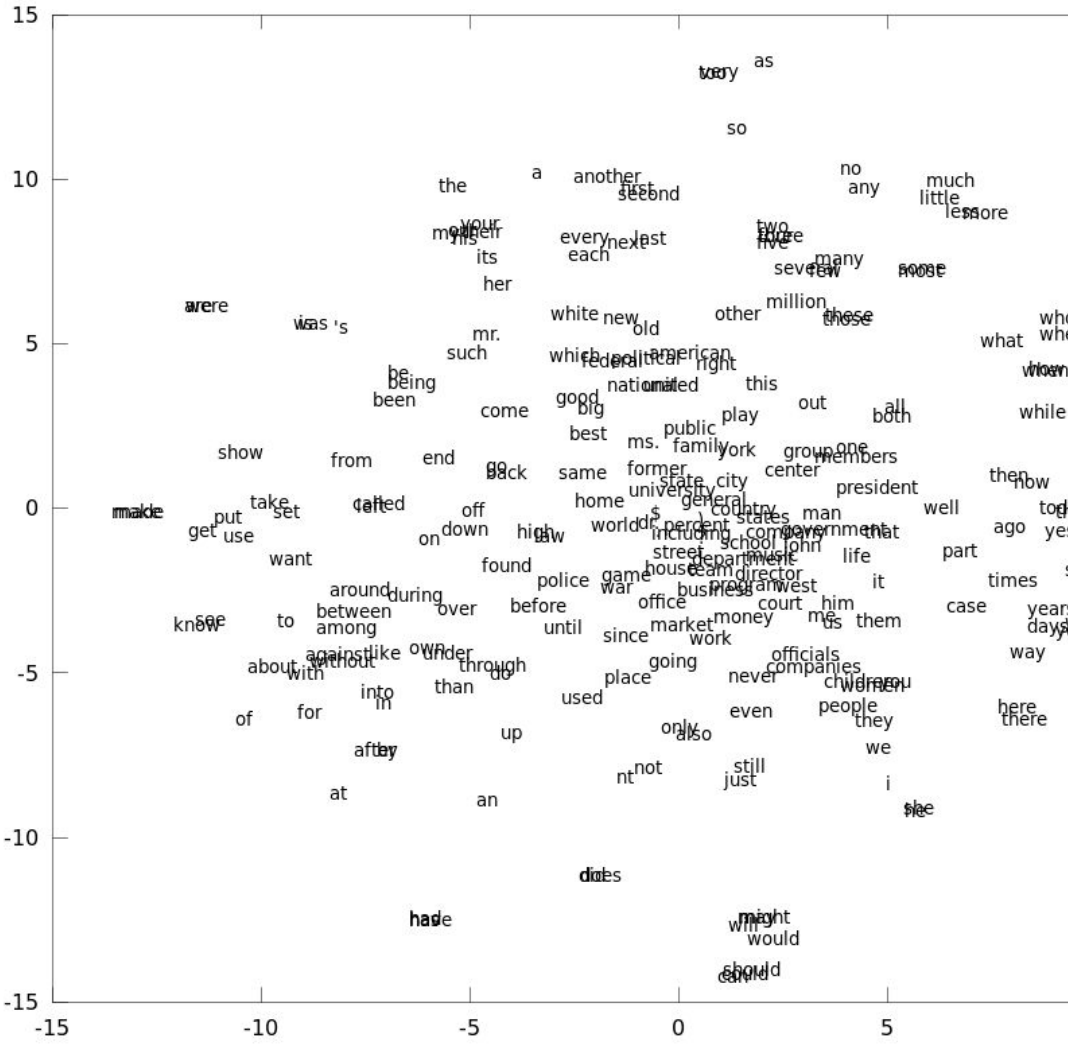
1. knowledge representation and automated reasoning
   - Logical connectives (or boolean operators)
   - Propositional logic
   - First-order logic
2. Wordnet and Word Sense Disambiguation
3. Introduction to word embeddings

Word Embeddings

Universität Zürich[UZH]

Word embeddings: vector representation of words

Vector space models: represent words in a continuous vector space



Word Embeddings Approaches

Universität Zürich[UZH]

1. Count-based methods

   How often some word co-occur with its neighbor words

   e.g. Latent Semantic Analysis

2. Predictive Methods

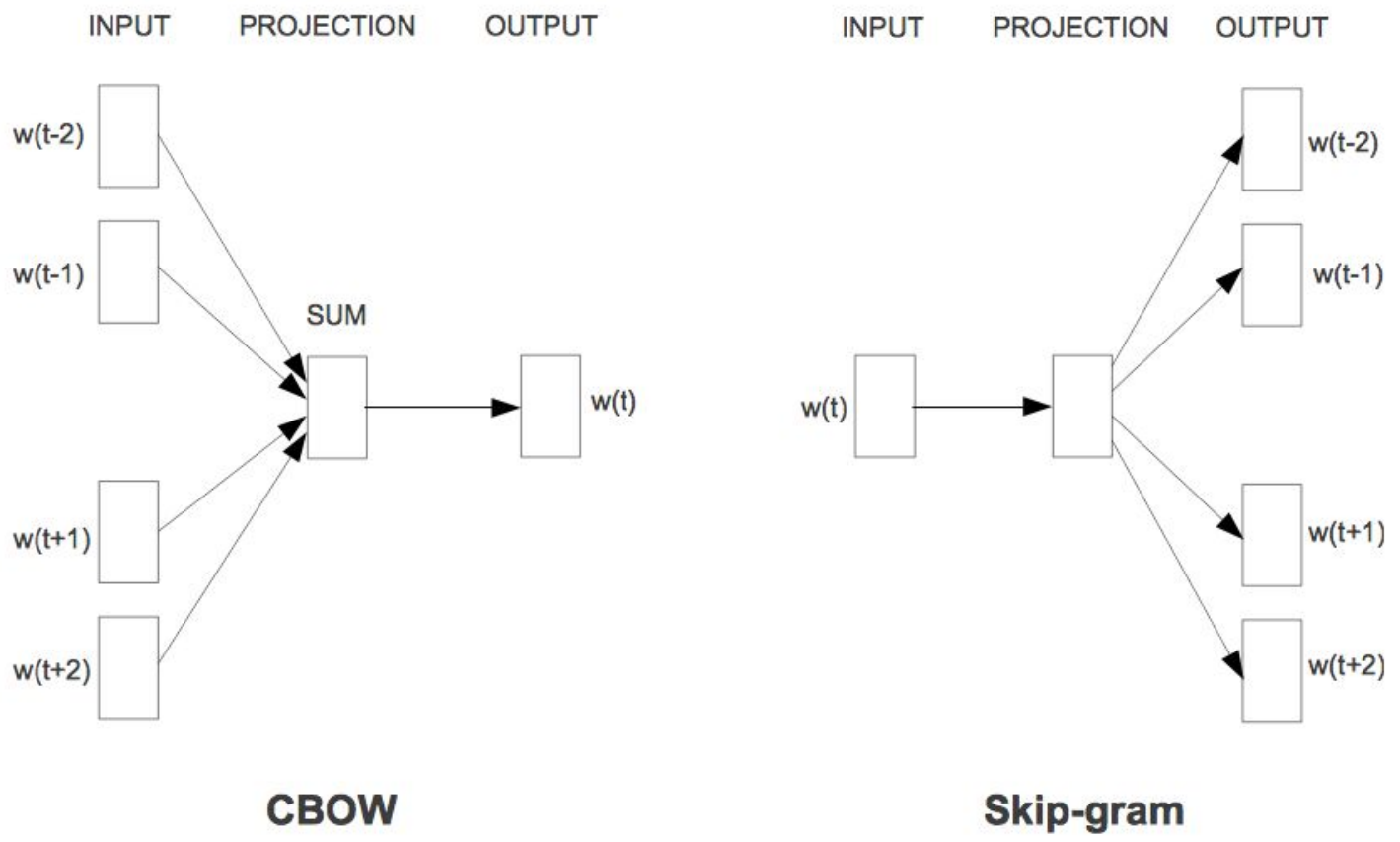
   Predict a word from its neighbors

   e.g. Neural Probabilistic Language Models

   e.g. Word2Vec [Mikolov et al., 2013]

Predictive Method: Word2Vec Model Architectures

Universität Zürich[UZH]

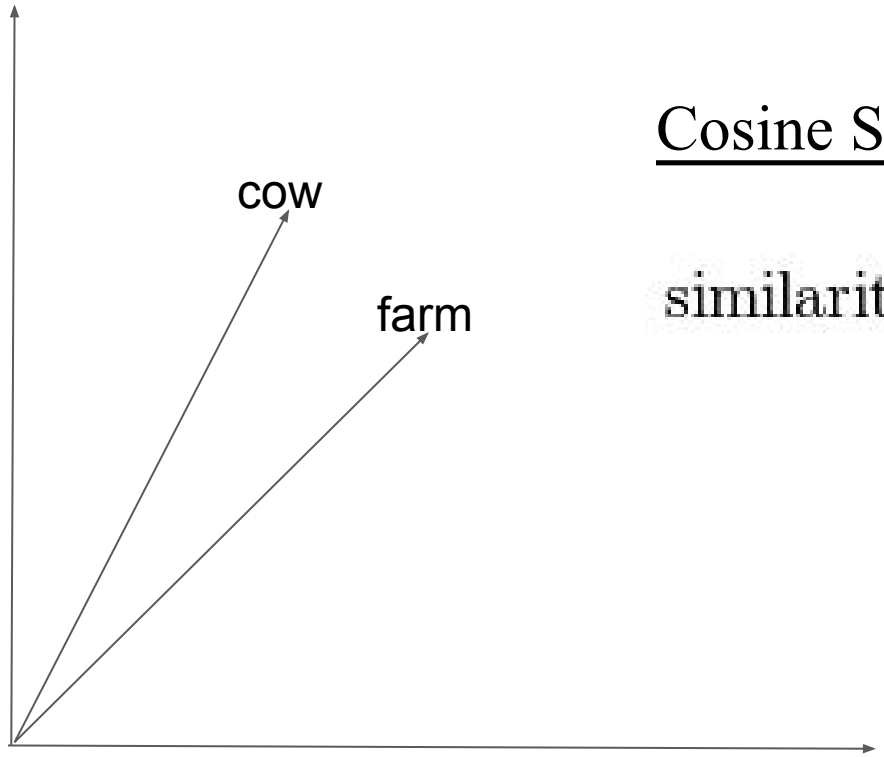


[Mikolov et al., 2013]

Predictive Method: Word2Vec
Vector Similarity

Universität Zürich[UZH]

How to compute similarity between words?



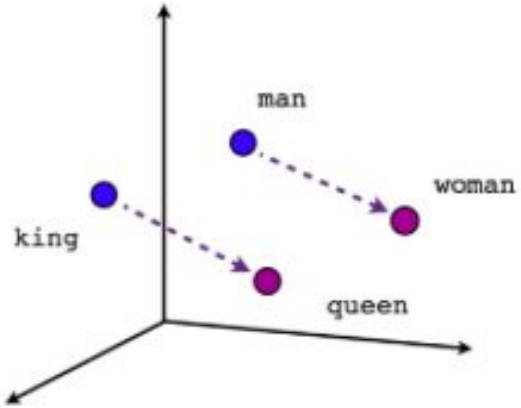
Cosine Similarity

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|}$$

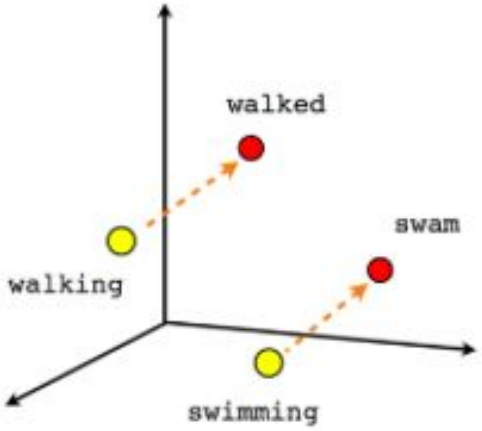
Predictive Method: Word2Vec Semantic Relationships

Universität Zürich[UZH]

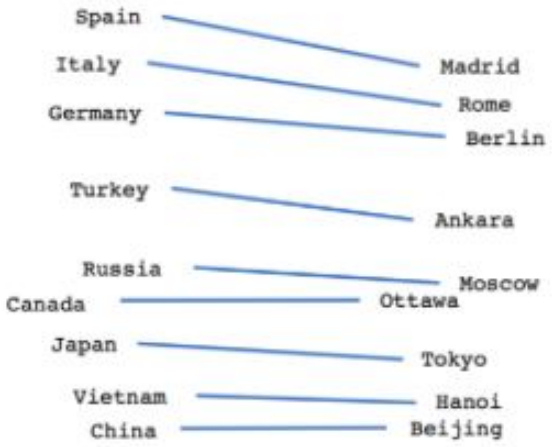
"What is the word that is similar to *king* in the same sense as *woman* is similar to *man*?"



Male-Female



Verb tense



Country-Capital

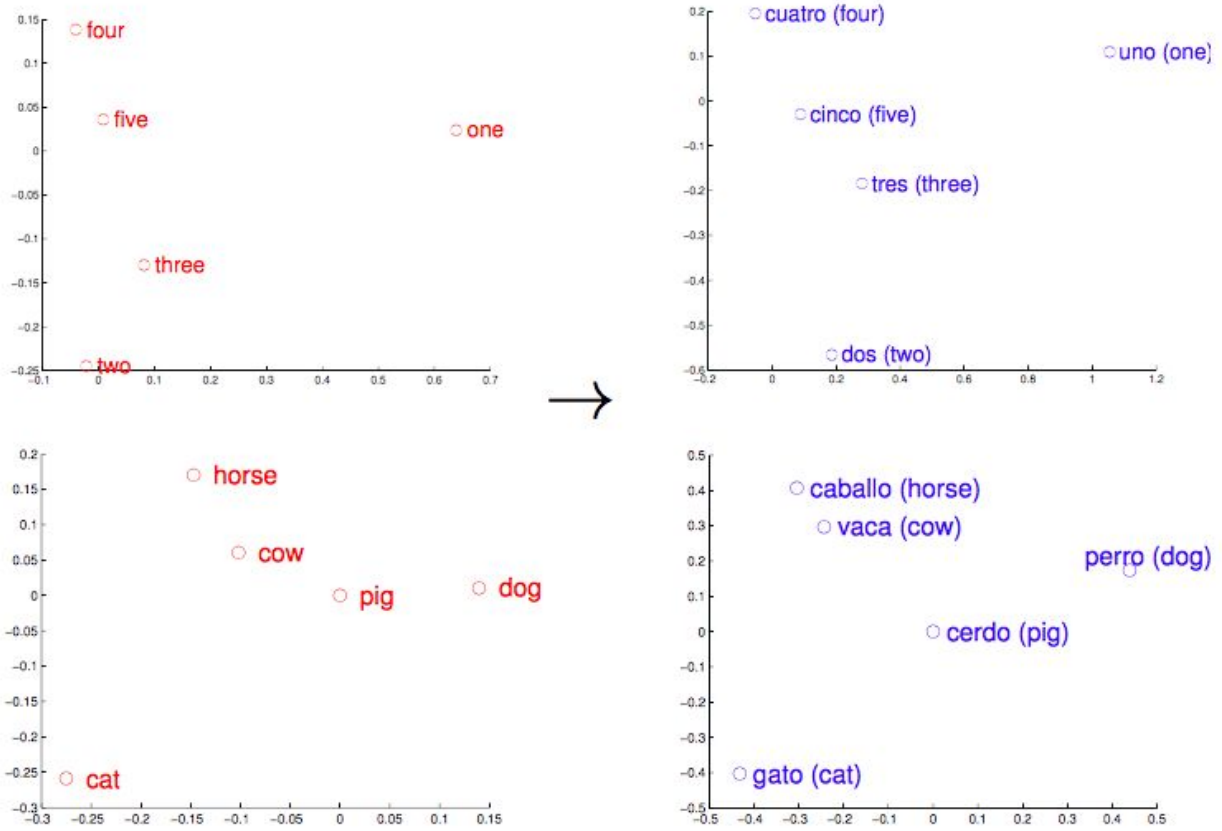
X = vector("*woman*") - vector("*man*") + vector("*king*")

Queen is the word in the vectors space closest to X

Predictive Method: Word2Vec Semantic Relationships

Universität Zürich[UZH]

Exploiting Similarities among Languages



[Mikolov, et al., 2013]

Predictive Method: Word2Vec

Universität Zürich[UZH]

Words with several meanings:

e.g. *Charge* appears in the expressions:

- *Charge* a fee
- *Charge* a battery

Predictive Method: Word2Vec

Universität Zürich[UZH]

Words with several meanings:

e.g. *Charge* appears in the expressions:

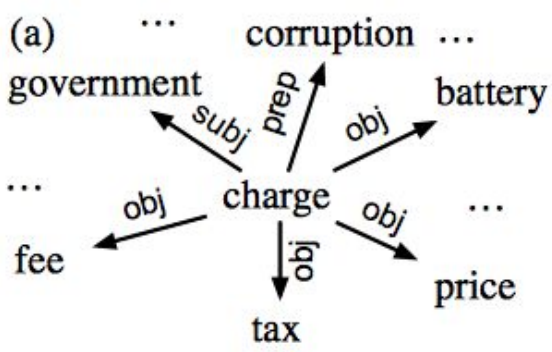
- *Charge* a fee
- *Charge* a battery

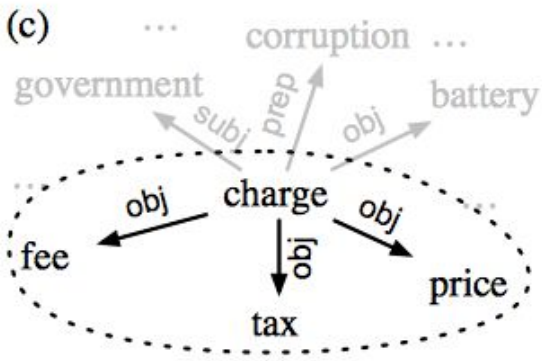
Problem: Vectors do not reflect charge/impose as near-synonyms.

Predictive Method: Word2Vec

Universität Zürich[UZH]

1. Word embeddings &
   external resources (e.g. WordNet)
   [Rothe S. and Schütze H., 2015]
   http://www.cis.lmu.de/~sascha/AutoExtend/
2. Contextualized vectors
   [Thater et al., 2011]





Computational Semantics

PCL II, CL, UZH
May 25, 2016

Universität Zürich[UZH]