language
evaluation
NLP
learning
systems
text
words
tasks
data
rules
speech
system
languages
translation
recognition
often
human
processing
input
task
machine
algorithms
information
natural
research
models
statistical
e.g
sentence
set
based
one
especially
standard
segmentation
possible
also
many
Given
edit
example
difficult
determine
chunk
different
problem
corpora
complexity
English
word
sentences
typically
named
considered
real-world
make
Note
written
developed
semantics
vs
project
result
mainly
specific
people
tagger
extraction
morphology
disambiguation
resolution
etc
gold
parsers
compared
meaning
type
Automatic
generally
However
section
sources
ambiguity
generation
criteria
intelligence
general
large
names
common
door
separate
may
procedures
including
compare
French
number
within
machine-learning
quality
referred
output
automatic
semantic
automatically
grammar
however
corresponding
Many
understanding
related
use
results
i.e
world
POS
using
much
identifying
complex
although
hand-written
devoted
termed
reliable
given
probabilistic
usually
decisions
problems
entities
campaign
Natural
larger
analysis
identify
knowledge
made
multiple
refer
serve
sometimes
focused
concerned
1980s
IR
conducted
process
entity

LEXICAL SEMANTICS & WORD SENSE DISAMBIGUATION

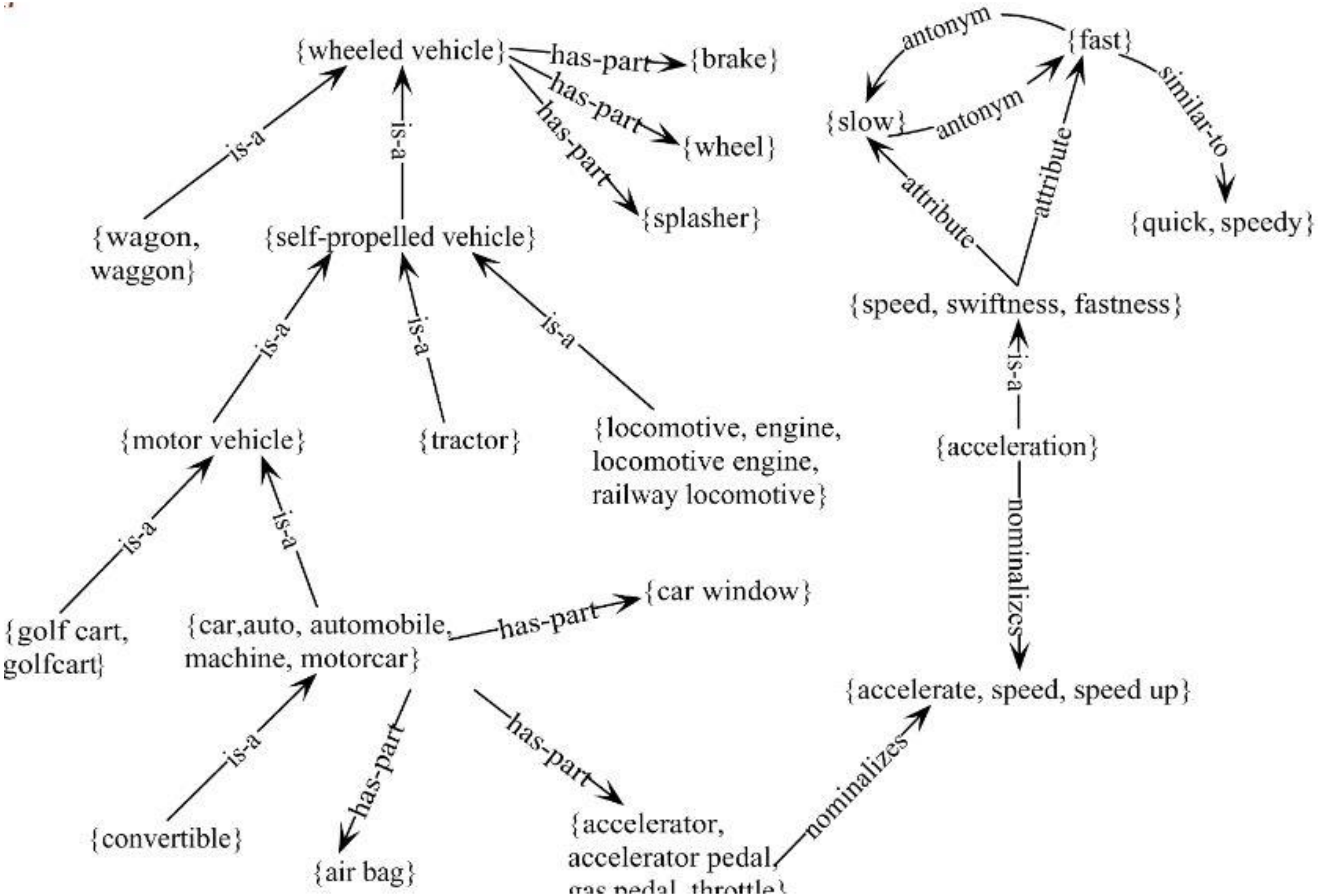
Semantic

Different sentences (wordings) have the same meaning:

```
Does this restaurant serve vegetarian dishes?
Do they have vegetarian stuff?
Are there any vegetarian options?
```

Aim: Derive the same meaning representation for all three sentences.

WordNet



{wheeled vehicle}
has-part
{brake}
has-part
{wheel}
has-part
{splasher}
is-a
is-a
{wagon, waggon}
{self-propelled vehicle}
is-a
is-a
is-a
{motor vehicle}
{tractor}
{locomotive, engine, locomotive engine, railway locomotive}
is-a
is-a
{golf cart, golfcart}
{car,auto, automobile, machine, motorcar}
has-part
{car window}
is-a
{convertible}
has-part
{air bag}
has-part
{accelerator, accelerator pedal, gas pedal, throttle}
nominalizes
{accelerate, speed, speed up}
antonym
{fast}
{slow}
antonym
similar-to
{quick, speedy}
attribute
attribute
{speed, swiftness, fastness}
is-a
{acceleration}
nominalizes

```python
from nltk.corpus import wordnet as wn

"""Synset—"synonym set"—a collection of synonymous words"""
print(wn.synsets('motorcar'))

print(wn.synset('car.n.01').lemma_names())

"""a word might be ambiguous, for example, a printer:"""
print(wn.synsets('printer'))

for synset in wn.synsets('printer'):
    print("\tLemma: {}".format(synset.name()))
    print("\tDefinition: {}".format(synset.definition()))
    print("\tExample: {}".format(synset.examples()))


"""Hyponym—a more specific concept"""

machine_that_prints = wn.synset('printer.n.03')
print(sorted([lemma.name() for synset in machine_that_prints.hyponyms() for lemma in synset.lemmas()]))

"""Hypernym—a more general concept."""

print([lemma.name() for synset in machine_that_prints.hypernyms() for lemma in synset.lemmas()])


"""Similarity"""

truck = wn.synset('truck.n.01')
limousine = wn.synset('limousine.n.01')

print(truck.lowest_common_hypernyms(limousine))

"""Example"""
train = wn.synset('train.n.01')
horse = wn.synset('horse.n.01')
animal = wn.synset('animal.n.01')
atom = wn.synset('atom.n.01')

print("Train => Horse: {}".format(train.path_similarity(horse)))

print("Horse => Train: {}".format(horse.path_similarity(train)))

print("Horse => Animal: {}".format(horse.path_similarity(animal)))
```

Two classes of similarity algorithms

Thesaurus based algorithms

- Are words “nearby” in hypernymhierarchy?
- Do words have similar glosses (definitions)?

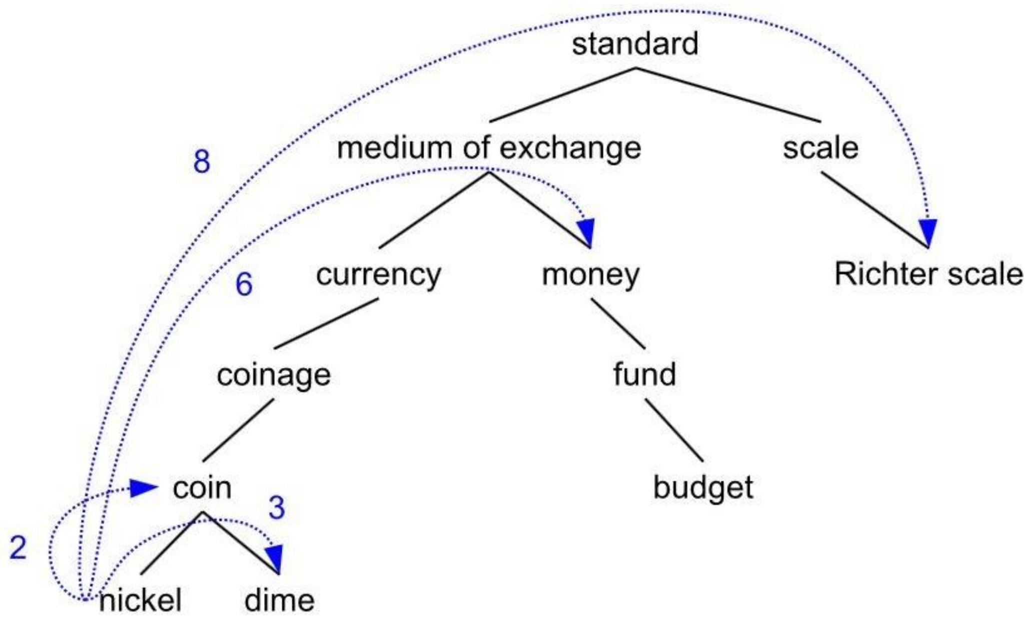
Distributional algorithms

- Do words have similar distributional contexts?
- Distributional (Vector) semantics.

Thesaurus based algorithm

Two concepts (senses/synsets) are similar if they are near each other in the thesaurus hierarchy

```
- have a short path between them
- concepts have path 1 to themselves
```



$\text{pathlen}(c_1,c_2)$ = 1 + number of edges in the shortest path in the hypernym graph between sense nodes c_1 and c_2

ranges from 0 to 1 (identity)

$$\text{simpath}(c_1,c_2) = \frac{1}{\text{pathlen}(c_1,c_2)}$$

$$\text{wordsim}(w_1,w_2) = \max_{c_1 \in \text{senses}(w_1),\, c_2 \in \text{senses}(w_2)} \text{sim}(c_1,c_2)$$

Distributional Similarity

"Differences of meaning correlates with differences of distribution" (Harris, 1970)

Idea: similar linguistic objects have similar contents (for documents, sentences) or contexts(for words)

Two Kinds of Distributional Contexts

1.Documents as bags-of-words

```
Similar documents contain similar words;
similar words appear in similar documents
```

2.Words in terms of neighboring words

```
"You shall know a word by the company it keeps!" (Firth, 1957)
Similar words occur near similar sets of other words (e.g., in a 5-word window)
```

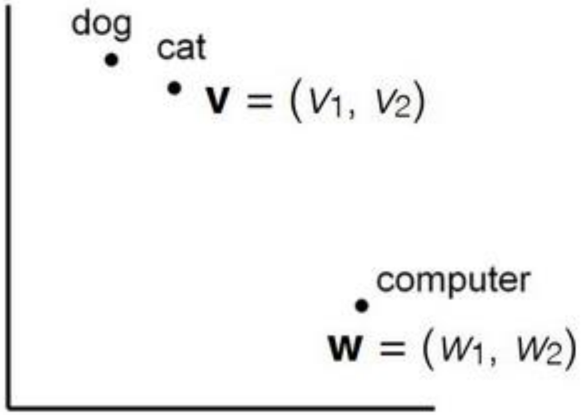
Word Vectors

A word type can be represented as a vector of features indicating the contexts in which it occurs in a corpus.

$$vec(v) = (f_1, f_2, f_3, \cdots, f_N)$$

Words in a Vector Space

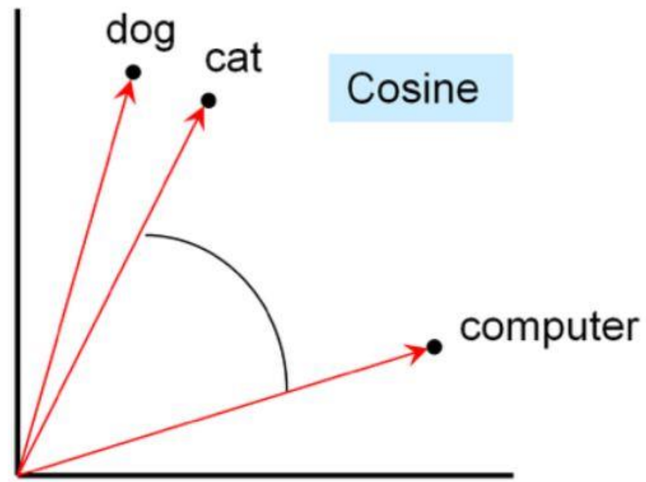
- In 2 dimensions:
  $\mathbf{v}$ = "cat"
  $\mathbf{w}$ = "computer"



Cosine Similarity

Cosine distance: borrowed from information retrieval

$$sim_{cosine}(\vec{v} \cdot \vec{w}) = \frac{\sum_{i=1}^{N} v_i \times w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}}$$



Example Calculate cosine similarity for words I and like.

$$X = \begin{array}{c} \\ I \\ like \\ enjoy \\ deep \\ learning \\ NLP \\ flying \\ . \end{array} \begin{array}{c} \begin{array}{cccccccc} I & like & enjoy & deep & learning & NLP & flying & . \end{array} \\ \begin{bmatrix} 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

Word Sense Disambiguation

- This is a task where you use a corpus to learn how to disambiguate a small set of target words using supervised learning. The aim is to build a classifier that maps each occurrence of a target word in a corpus to its sense.
- We will use a Naive Bayes classifier. In other words, where the context of an occurrence of a target word in the corpus is represented as a feature vector, the classifier estimates the word senses on the basis of its context as shown below.

Naive Bayes for Word Sense Disambiguation

We assume that each feature is conditionally independent of all the other features:

$$
\begin{aligned}
\hat{s} &= \arg\max_{s \in S} P(s|\vec{f}) && \\
&= \arg\max_{s \in S} \frac{P(\vec{f}|s)P(s)}{P(\vec{f})} && \text{Bayes} \\
&= \arg\max_{s \in S} P(\vec{f}|s)P(s) && P(\vec{f}) \text{ is fixed} \\
&\approx \arg\max_{s \in S} P(s) \prod_{j=1}^{n} P(f_j|s) && \text{cond. independence}
\end{aligned}
$$

Word Vectors

Collocational features and bag-of-words features

Collocational

Features about words at specific positions near target word – Often limited to just word identity and POS

Bag-of-words

Features about words that occur anywhere in the window (regardless of position) – Typically limited to frequ
ency counts

- **Example** L = About three years ago, he nearly gave up because he had nothing to sell; now his shelves are full, and towels and clothes hang from a **line** overhead.
- Give a collocational feature vector for the word **line** in **L**, given a window size of 3 words to the left and 3 words to the right.(±3)
- Give a bag-of-words feature vector for the word **line** in **L**, given the following word feature list: [written, school, speech, row, major, hang, sell, nothing, rope, words]

Example:

```
L = results <p="NNS"/> are <p="VBP"/> due <p="JJ"/> on <p="IN"/> march
<p="NNP"/> 13 <p="CD"/> . <p="."/> at <p="IN"/> the <p="DT"/> same
<p="JJ"/> time <p="NN"/> , <p=","/> local <p="JJ"/> talks <p="NNS"/> to
<p="TO"/> restart <p="VB"/> some <p="DT"/> sort <p="NN"/> of <p="IN"/>
<head> accident <p="NN"/></head> and <p="CC"/> emergency <p="NN"/> service
<p="NN"/> appear <p="VB"/> to <p="TO"/> be <p="VB"/> getting <p="VBG"/>
nowhere <p="RB"/> . <p="."/>
```

- Give a collocational feature vector for the word **accident** in **L**, given a window size of 5 words to the left and 5 words to the right.(±5)

Example: Bag-of-words vector for "watch":

[likes, movies, time, escape, football, wrist, prison, night]

What is the correct binary vector representation for sentences(±10)

- Mary also likes to watch football games
- John’s new watch shows the time in five locations.
- The investigation clearly shows that no-one has escaped the prison during my watch!
- Duels usually have one scene where the actors go out of frame and you watch their shadows fighting, at least in cliche movies.

Example

Cosine Similarity

cat:

- The **cat** was playing in the garden.
- The owner feed her **cat** every morning.
- You can find **cat** food in the markets.
- The **cat** often eats in the morning.
- They were fighting like a **cat** and a dog.
- How much should I feed my **cat**?
- Her **cat** was always sleeping.

dog:

- The family's cat and **dog** are playing in the garden.
- Encourage your **dog** to play in the garden.
- **Dog** food is not sold here.
- His **dog** does not eat meat.
- The **dog** was hit by a car.
- I never feed my **dog** raw meat.

Given two words and a few sentences containing them, compute their cosine similarity.
Use bag-of-words method with a window size of ±3 by assigning frequency values in the feature vector.

Naive Bayes

- **Example** Let's walk through an example of training and testing naive Bayes with add-one smoothing. We'll use a sentiment analysis domain with the two classes positive(+) and negative(-), and take the following miniature training and test documents simplified from actual movie reviews.

	Cat	Documents
Training	-	just plain boring
	-	entirely predictable and lacks energy
	-	no surprises and very few laughs
	+	very powerful
	+	the most fun film of the summer
Test	?	predictable with no fun