Introduction to Natural Language Processing

Announcements

- HW1 posted on Thursday
  - Any questions?

- Are you on eCommons and Piazza?
  - If not, email me to get access (elahe@soe.ucsc.edu)

Main Goals for Week 2

Moving Beyond Words

WORD CLASSES, LEXICAL INFORMATION

TAGGING WORD SEQUENCES

LANGUAGE MODELS (N-GRAMS)

PROBABILITY AND CORPUS-BASED NLP

REGULAR EXPRESSIONS: WRITING PATTERNS FOR PHRASES

Goals for Today

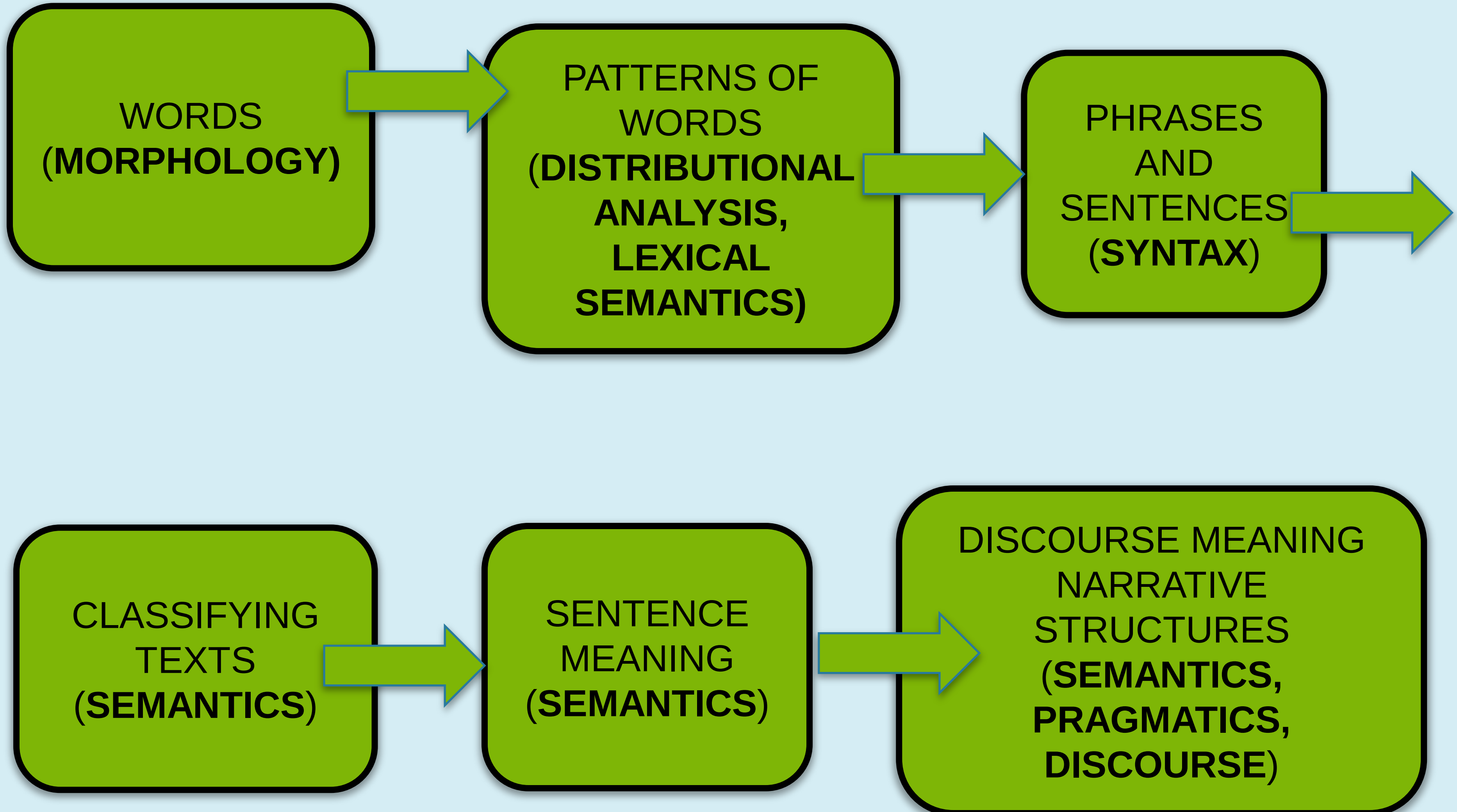
More WORDNET

REVIEW: PROBABILITY

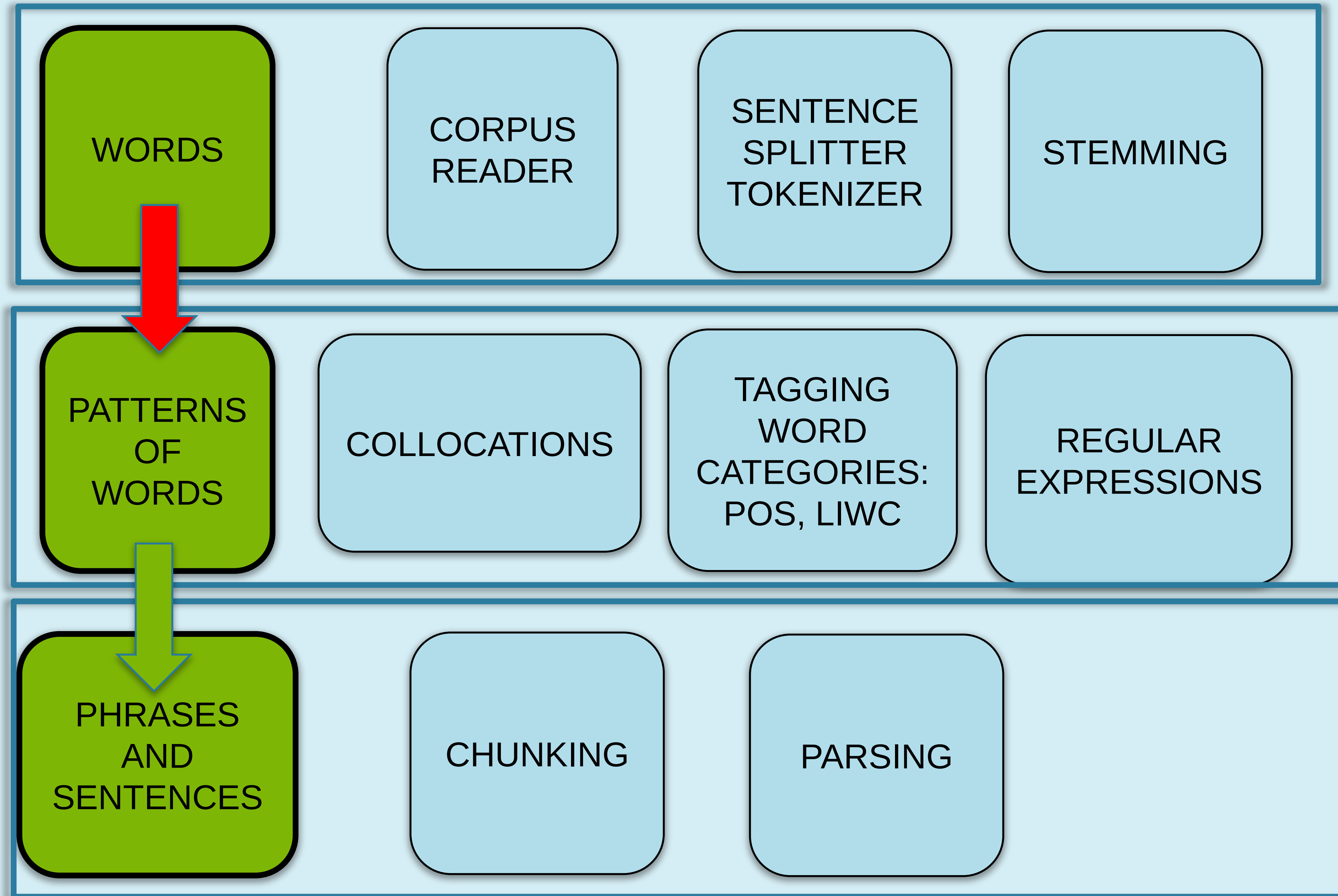
STATISTICAL NLP

CORPUS-BASED NLP

NLP PIPELINE



NLP Architecture



The Bigger Picture

- Our first application: Classifying Text using features of the texts
- The features should give us ‘abstractions’ over the basic words in the text
- How to get there
  - Learn how to produce featural descriptions of text
  - Words, stemmed words, bigrams, POS, sentiment dictionaries, dictionaries, word sequences, POS sequences, other useful patterns
  - These are abstractions of the basic words in the text that should support **generalization** to unseen cases

Example of featural description of text: Restaurant Review

The staff of the ***restaurant*** *is* ***nice*** *and the* ***eggplant*** *is not bad. Apart from that, very* ***uninspired food****, lack of atmosphere and too* ***expensive****. I am a staunch vegetarian and was sorely* ***disappointed*** *with the* ***veggie*** *options on the menu.*

Category: Location

Sentiment polarity (+/-)

Category: Food

Wordnet API
and Definitions

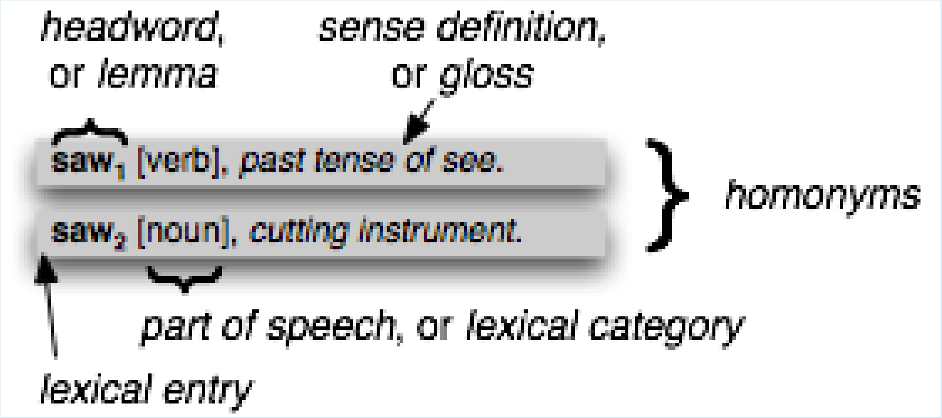
Lexical Resources

- A lexicon, or lexical resource, is a collection of words and/or phrases along with some associated information such as part of speech and sense definitions.

- A vocabulary (list of words in a text) is the simplest lexical resource

- Lexical entry

A **lexical entry** typically consists of a **headword** (also known as a **lemma**) along with additional information such as the part of speech and the sense definition.



WordNet in NLTK

- Unlike some lexical resources (like regular dictionaries) that have a flat semantic structure, WordNet is organized into an *ONTOLOGY.*
- Ontologies link concepts by *LEXICAL RELATIONS*
- IS-A relation creates a hierarchy of concepts

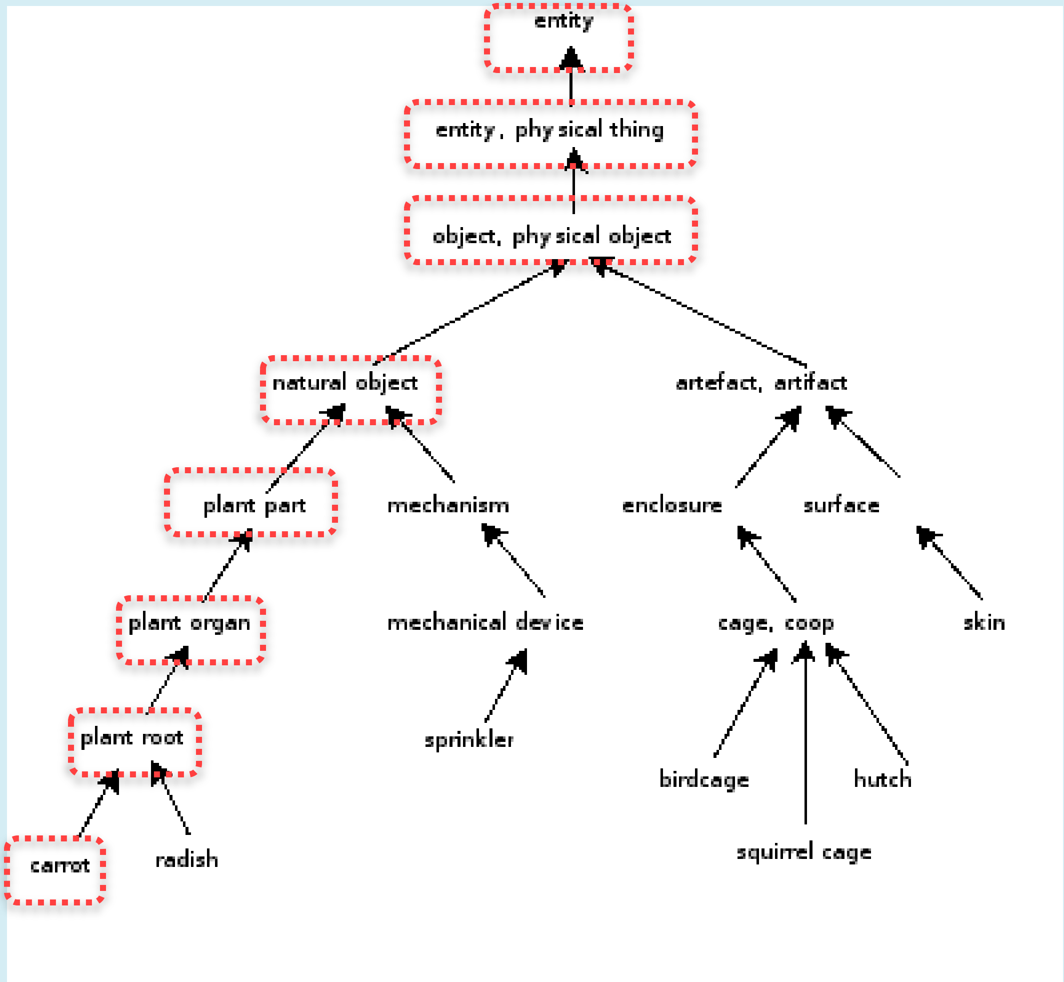
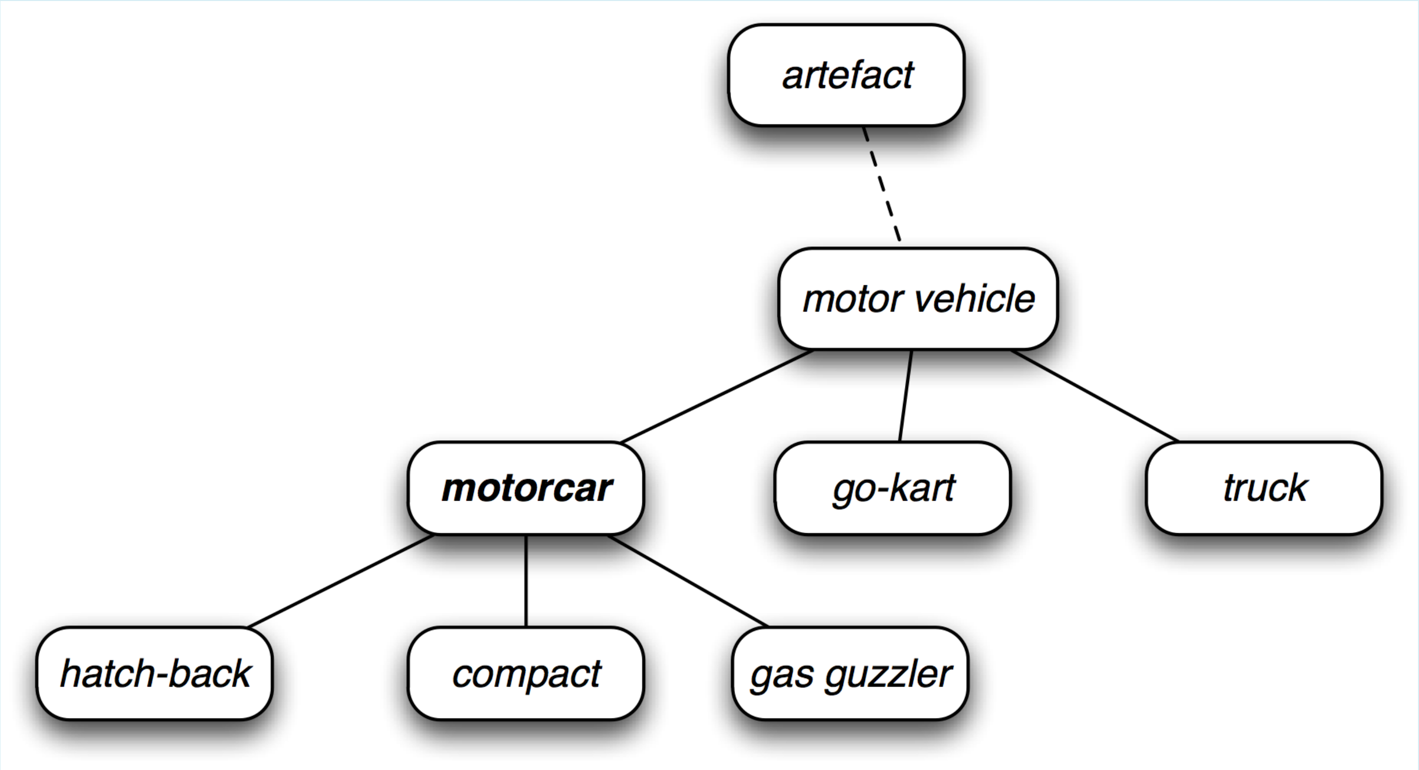


Figure 1. "is a" relation example

The WordNet Hierarchy in NLTK

- A **hyponym** is a word or phrase whose semantic field is more specific than its **hypernym**.
- Hypernyms and hyponyms are called **lexical relations** because they relate one **synset** to another. These two relations navigate up and down the "*is-a*" hierarchy.



WordNet is in NLTK: Ch. 2, Sec. 5

- Using the WordNet API

```
>>> from nltk.corpus import wordnet as wn
>>> wn.synsets('motorcar')
[Synset('car.n.01')]
```

- Thus, *motorcar* has just one possible meaning and it is identified as car.n.01, the first **noun sense** of *car*.
- The entity car.n.01 is called a **synset**, or "synonym set", a collection of synonymous words or "lemmas”.

```
>>> wn.synset('car.n.01').lemma_names()
['car', 'auto', 'automobile', 'machine', 'motorcar']
```

- Synsets also come with a definition and some example sentences:

```
>>> wn.synset('car.n.01').definition()
'a motor vehicle with four wheels; usually propelled by an internal combustion engine'
>>> wn.synset('car.n.01').examples()
['he needs a car to get to work']
```

Wordnet API in NLTK

- Definitions help understand the intended meaning of a synset.
- The *words* of the synset are often more useful:

  car.n.01.motorcar

- This pairing of a **synset** with a word is called a **lemma**
  - get all the lemmas, look up a particular lemma , get the synset corresponding to a lemma , and get the "name" of a lemma

```
>>> wn.synset('car.n.01').lemmas() ❶
[Lemma('car.n.01.car'), Lemma('car.n.01.auto'), Lemma('car.n.01.automobile'),
Lemma('car.n.01.machine'), Lemma('car.n.01.motorcar')]
>>> wn.lemma('car.n.01.automobile') ❷
Lemma('car.n.01.automobile')
>>> wn.lemma('car.n.01.automobile').synset() ❸
Synset('car.n.01')
>>> wn.lemma('car.n.01.automobile').name() ❹
'automobile'
```

Wordnet API in NLTK

- Some words are ambiguous and have more than one synset
  - Car → 5 synsets

```
>>> wn.synsets('car')
[Synset('car.n.01'), Synset('car.n.02'), Synset('car.n.03'), Synset('car.n.04'),
Synset('cable_car.n.01')]
>>> for synset in wn.synsets('car'):
...     print(synset.lemma_names())
...
['car', 'auto', 'automobile', 'machine', 'motorcar']
['car', 'railcar', 'railway_car', 'railroad_car']
['car', 'gondola']
['car', 'elevator_car']
['cable_car', 'car']
```

  - Access all the lemmas involving the word *car* from all its synsets :

```
>>> wn.lemmas('car')
[Lemma('car.n.01.car'), Lemma('car.n.02.car'), Lemma('car.n.03.car'),
Lemma('car.n.04.car'), Lemma('cable_car.n.01.car')]
```

Wordnet API in NLTK

- It's very easy to navigate between concepts. For example, given a concept like *motorcar*, we can look at the concepts that are more specific; the (immediate) **hyponyms**.

```
>>> motorcar = wn.synset('car.n.01')
>>> types_of_motorcar = motorcar.hyponyms()
>>> types_of_motorcar[26]
Synset('ambulance.n.01')
>>> sorted([lemma.name for synset in types_of_motorcar for lemma in synset.lemmas])
['Model_T', 'S.U.V.', 'SUV', 'Stanley_Steamer', 'ambulance', 'beach_waggon',
'beach_wagon', 'bus', 'cab', 'compact', 'compact_car', 'convertible',
'coupe', 'cruiser', 'electric', 'electric_automobile', 'electric_car',
'estate_car', 'gas_guzzler', 'hack', 'hardtop', 'hatchback', 'heap',
'horseless_carriage', 'hot-rod', 'hot_rod', 'jalopy', 'jeep', 'landrover',
'limo', 'limousine', 'loaner', 'minicar', 'minivan', 'pace_car', 'patrol_car',
'phaeton', 'police_car', 'police_cruiser', 'prowl_car', 'race_car', 'racer',
'racing_car', 'roadster', 'runabout', 'saloon', 'secondhand_car', 'sedan',
'sport_car', 'sport_utility', 'sport_utility_vehicle', 'sports_car', 'squad_car',
'station_waggon', 'station_wagon', 'stock_car', 'subcompact', 'subcompact_car',
'taxi', 'taxicab', 'tourer', 'touring_car', 'two-seater', 'used-car', 'waggon',
'wagon']
```

WordNet Search - 3.1
- WordNet home page - Glossary - Help

Word to search for: motorcar [Search WordNet]

Display Options: (Select option to change) [Change]

Key: "S:" = Show Synset (semantic) relations, "W:" = Show Word (lexical) relations

Noun

- S: (n) car, auto, automobile, machine, **motorcar**
  - ***direct hyponym*** / *full hyponym*
    - S: (n) ambulance
    - S: (n) beach wagon, station wagon, wagon, estate car, beach waggon, station waggon, waggon
    - S: (n) bus, jalopy, heap
    - S: (n) cab, hack, taxi, taxicab
    - S: (n) compact, compact car
    - S: (n) convertible
    - S: (n) coupe
    - S: (n) cruiser, police cruiser, patrol car, police car, prowl car, squad car
    - S: (n) electric, electric automobile, electric car
    - S: (n) gas guzzler
    - S: (n) hardtop
    - S: (n) hatchback
    - S: (n) horseless carriage

WordNet: More Lexical Relations

- Another important way to navigate the WordNet network is from items to their components (**meronyms**) or to the things they are contained in (**holonyms**).
  - For example, the parts of a *tree* are its *trunk*, *crown*, …:

    `part_meronyms()`
  - The *substance* a tree is made of includes *heartwood* and *sapwood* :

    `substance_meronyms()`
  - A collection of trees forms a *forest*: `member_holonyms()`

```
>>> wn.synset('tree.n.01').part_meronyms()
[Synset('burl.n.02'), Synset('crown.n.07'), Synset('stump.n.01'),
Synset('trunk.n.01'), Synset('limb.n.02')]
>>> wn.synset('tree.n.01').substance_meronyms()
[Synset('heartwood.n.01'), Synset('sapwood.n.01')]
>>> wn.synset('tree.n.01').member_holonyms()
[Synset('forest.n.01')]
```

WordNet: Semantic Similarity

- Synsets are linked by a complex network of lexical relations
  - Traverse the WordNet network to find synsets with related meanings
- Knowing which words are semantically related is useful
  - Grouping words into larger sets (generalization for sparse data in classification)
  - Indexing a collection of texts
    - search for a general term like *vehicle* will match documents containing specific terms like *limousine*
- Two synsets linked to the same root may have several hypernyms in common.
  - **Idea:** If two synsets share a very specific hypernym (low down in the hierarchy) they must be closely related

WordNet: Semantic Similarity

- *whale* is very specific
- *baleen whale* even more specific
- *vertebrate* is more general and *entity* is completely general

```
>>> right = wn.synset('right_whale.n.01')
>>> orca = wn.synset('orca.n.01')
>>> minke = wn.synset('minke_whale.n.01')
>>> tortoise = wn.synset('tortoise.n.01')
>>> novel = wn.synset('novel.n.01')
>>> right.lowest_common_hypernyms(minke)
[Synset('baleen_whale.n.01')]
>>> right.lowest_common_hypernyms(orca)
[Synset('whale.n.02')]
>>> right.lowest_common_hypernyms(tortoise)
[Synset('vertebrate.n.01')]
>>> right.lowest_common_hypernyms(novel)
[Synset('entity.n.01')]
```

- We can quantify the concept of generality by looking up the depth of each synset

```
>>> wn.synset('baleen_whale.n.01').min_depth()
14
>>> wn.synset('whale.n.02').min_depth()
13
>>> wn.synset('vertebrate.n.01').min_depth()
8
>>> wn.synset('entity.n.01').min_depth()
0
```

WordNet: Semantic Similarity

- Similarity measures have been defined over the collection of WordNet synsets which incorporate this insight.
- For example, **path_similarity** assigns a score in the range 0–1 based on the shortest path that connects the concepts in the hypernym hierarchy
- Higher is more similar

```
>>> right.path_similarity(minke)
0.25
>>> right.path_similarity(orca)
0.16666666666666666
>>> right.path_similarity(tortoise)
0.076923076923076927
>>> right.path_similarity(novel)
0.043478260869565216
```

- The numbers don’t mean much, but they decrease as we move away from the semantic space of sea creatures to inanimate objects.

Further Reading: Learning Hyponym Relations from Text

- Automated Discovery of WordNet Relations, Marti A Hearst, 1998 (Book Chapter)
  - searching for corresponding lexico-syntactic patterns in large text collections

(2) *such NP as* $\{NP\ ,\}^*\ \{(or \mid and)\}$ *NP*

```
... works by such authors as Herrick, Goldsmith, and Shakespeare.
```

$\Longrightarrow$ *hyponym("author", "Herrick"),*
hyponym("author", "Goldsmith"),
hyponym("author", "Shakespeare")

(3) *NP* $\{,\ NP\}^*\ \{,\}$ *or other NP*

```
Bruises, ..., broken bones or other injuries ...
```

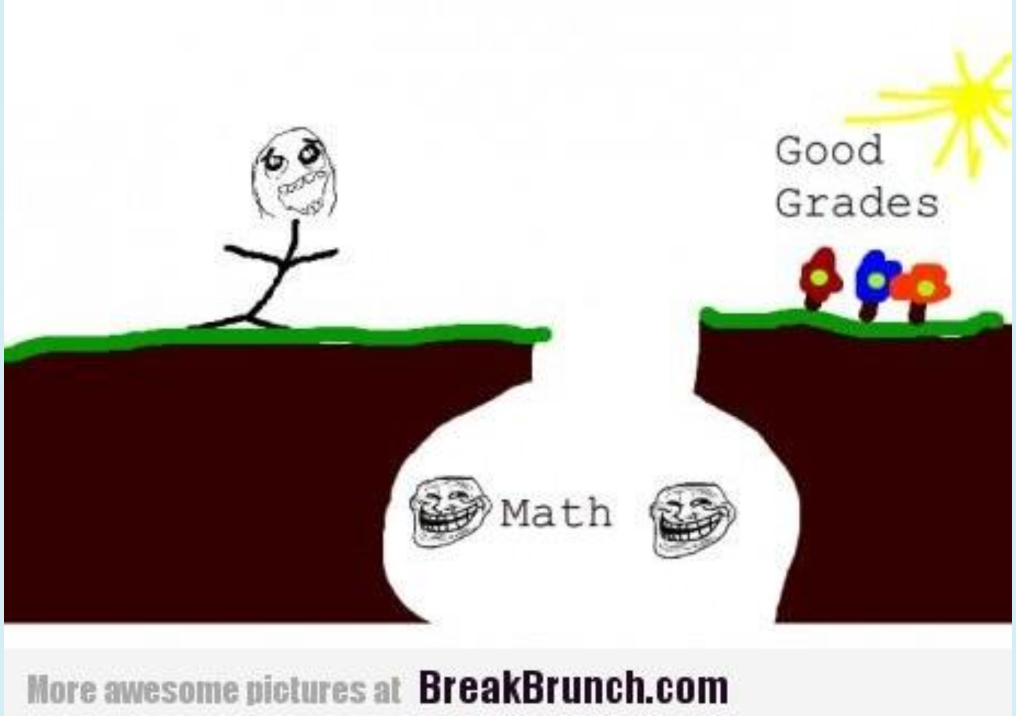
$\Longrightarrow$ *hyponym("bruise", "injury"),*
hyponym("broken bone", "injury")

Review of Probability and Conditional Probability

Should have had this in CE 16

Many of you don't remember it

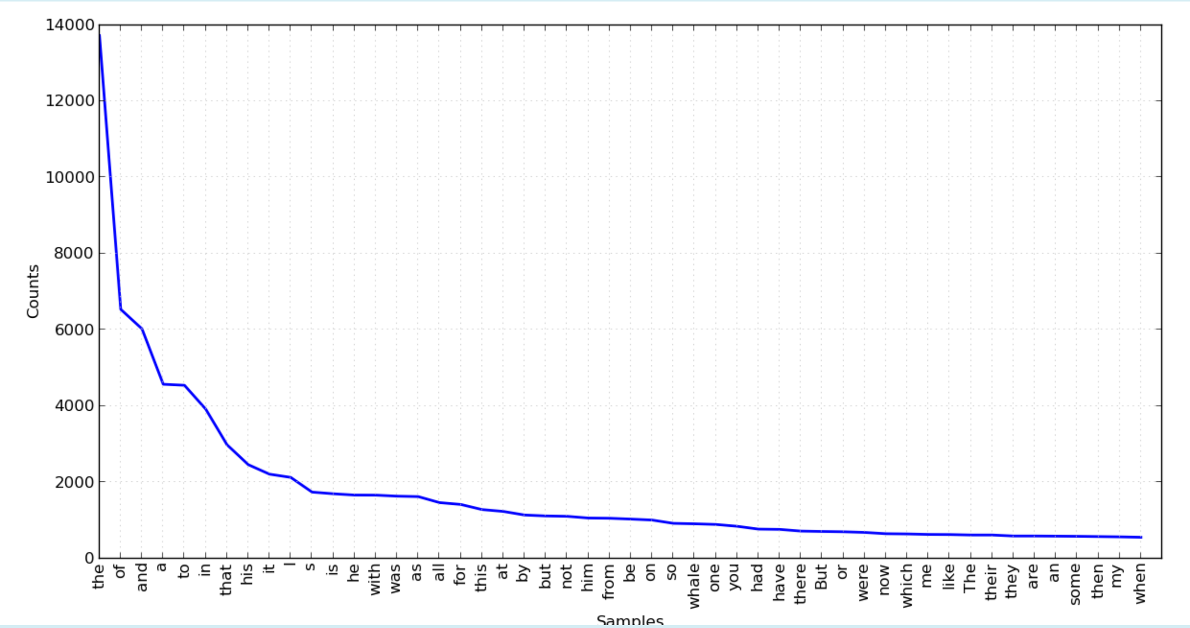
What is probability useful for?



Statistical NLP

- Statistical techniques for the automatic analysis of natural (human) language data
- Statistical NLP aims to do statistical inference for the field of NL

Statistical inference consists of taking some data and making some inference about its distribution



Language Model

- Statistical **Language Model** is a **probability distribution** over sequences of words.

- Given a sequence of words: $w_1, w_2, \ldots, w_m$
  - assigns a probability to the whole sequence:

    $P(w_1, w_2, \ldots, w_m)$
  - can be used to predict the next word

- Corpus-based (data-driven) approach
  - Compute probability of a sequence of tokens from data
  - Make inference from data
  - Probability theories will help us estimate these numbers from data

Examples of Applications

- Machine Translation:
  - P(**high** winds tonite) > P(**large** winds tonite)
- Spell Correction
  - The office is about fifteen minuets from my house
  - P(about fifteen **minutes** from) > P(about fifteen **minuets** from)
- Speech Recognition
  - P(I saw a van) >> P(eyes awe of an)
- + Summarization, Question-Answering, etc., etc.!!

Definition of Probability

- Probability theory encodes our knowledge or belief about the collective likelihood of the outcome of an event.

- We use probability theory to try to predict which outcome will occur for a given event.

Slide adapted from Dan Jurafsky's

Sample Spaces

- We think about the "sample space" as being set of all possible outcomes:
  - For tossing a coin, the possible outcomes are Heads or Tails.
  - For competing in the Olympics, the set of outcomes for a given contest are {gold, silver, bronze, no_award}.
  - For computing part-of-speech, the set of outcomes are {JJ, DT, NN, RB, … etc.}
- We use probability theory to try to predict which outcome will occur for a given event.

Slide adapted from Dan Jurafsky's

The classic example: Flipping a coin

- We flip a fair coin
- What are the possible outcomes (sample space)?
  - Heads (H) or Tails (T)
  - Either is equally likely
- What are the chances of getting H?
  - One out of two
  - $P(H) = ½ = 0.5$

Another Example

- We roll a dice. What are the chances of getting an even number?
- There are six possible outcomes from rolling a dice, each with a 1 out of 6 chance:
  {1, 2, 3, 4, 5, 6}
- There are 3 simple outcomes of interest making up the compound event of interest:
  - even numbers: {2, 4, 6}
  - any of these qualifies as “success”
  - effectively, we can be successful 3 times out of 6

$$P(\text{Even}) = 3/6 = 0.5$$

More Example

- We write a random number generator, which generates real numbers randomly between 0 and 200.
  - Numbers can be decimals
  - Valid outcomes: 0.00002, 148.16, 4, …

- The set of possible outcomes is
  - infinite
  - uncountable (continuous)

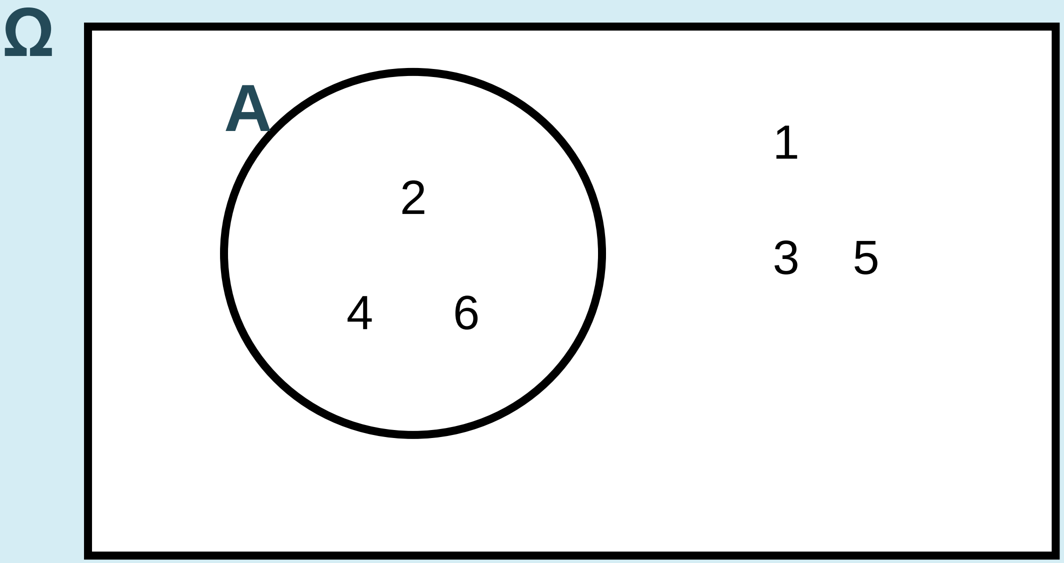
Some Notation…

- We use Ω to denote the total set of outcomes, our event space
  - Can be infinite! (cf. the random number generator)
  - Discrete event space: events can be identified individually (throw of dice)
  - Continuous event space: events fall on a continuum (number generator)

- We view events and outcomes as sets

Venn diagram representation of dice-throw example

- Possible outcomes: {1,2,3,4,5,6}
- Outcomes of interest (denoted A): {2,4,6}



Probability: classical interpretation

- Given n equally possible outcomes, and m events of interest, the probability that one of the m events occurs is m/n.
  - If we call our set of events of interest A, then:

$$P(A) = \frac{|A|}{|\Omega|}$$

Number of events of interest (A) over total number of events

- Principle of insufficient reason (Laplace):
  - We should assume that events are equally likely, unless there is good reason to believe they are not.

Compound vs. simple events

- If A is a compound event, then P(A) is the sum of the probabilities of the simple events making it up:

$$P(A) = \sum_{a \in A} P(a)$$

The sum of probabilities, for all elements a of A

- Recall, that P(Even) = 3/6 = 0.5
- In a throw of the Dice, the simple events are {1,2,3,4,5,6}, each with probability 1/6
- P(Even) = P(2) + P(4) + P(6) = 1/6 * 3 = 0.5

More rules…

- Since, for any compound event A:

$$P(A) = \sum_{a \in A} P(a)$$

the probability of all events, P(Ω) is:

$$P(\Omega) = \sum_{e \in \Omega} P(e) = 1$$

(this is the likelihood of “anything happening”, which is always 100% certain)

Yet more rules…

- If A is any event, the probability that A does not occur is the probability of the complement of A:

$$P(\overline{A}) = 1 - P(A)$$

  i.e. the likelihood that anything which is not in A happens.

- Impossible events are those which are not in Ω. They have probability of 0.
- For any event A:

$$P(A) \in [0,1]$$

Example: Throwing a dice

- A = {4}
- Probability that A does not occur
  - complement of A → B = {1, 2, 3, 5, 6}
  - P(B) = 5/6
  - P(A) = 1/6
  - P(B) = 1 – P(A)
- Probability that the outcome of rolling a dice is 8
  - Impossible!
  - P ({8}) = 0

More complicated outcomes…

- Here's an even more complicated example:
  - You flip a coin twice.
  - Possible outcomes (order irrelevant):
    - 2 heads (HH)
    - 1 head, 1 tail (HT)
    - 2 tails (TT)

Only one way to obtain this: both throws give H

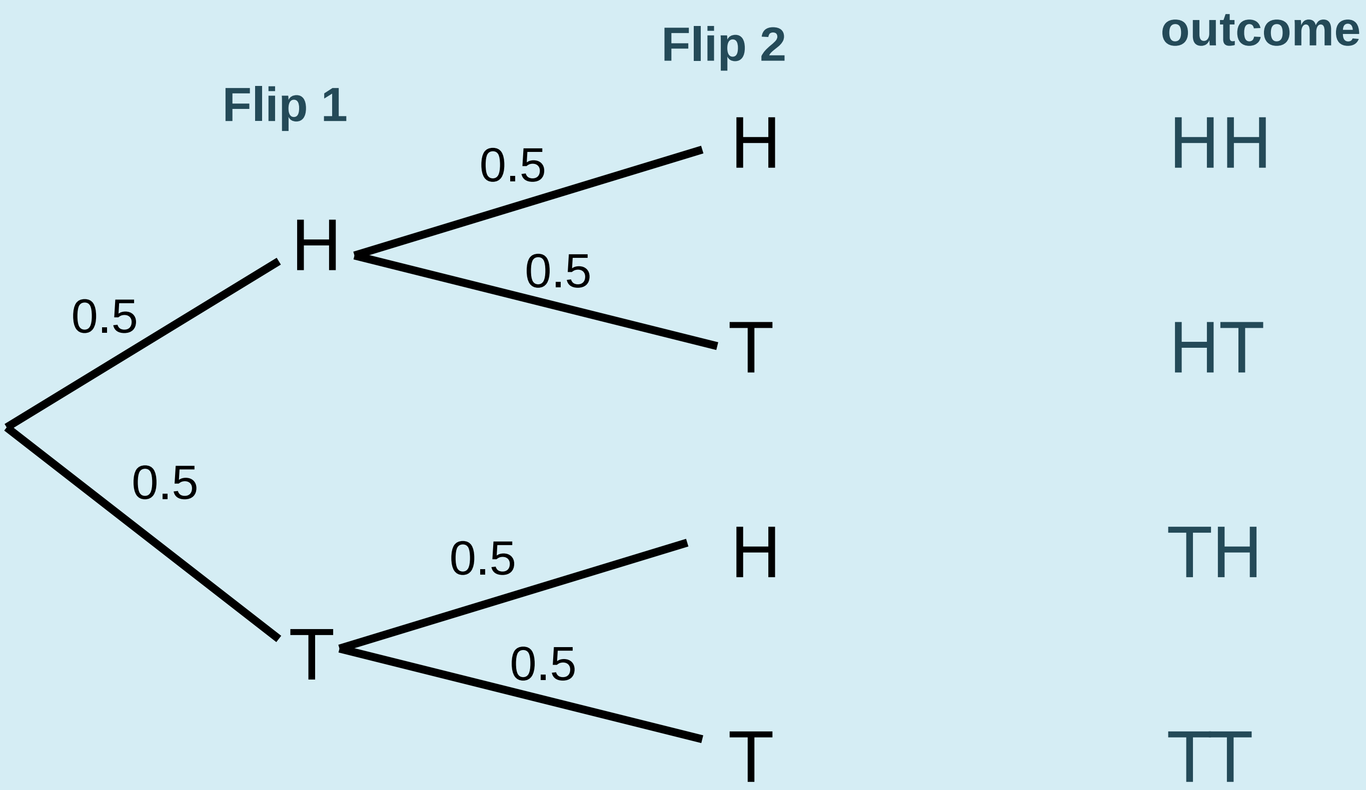
Two different ways to obtain this: {throw1=H, throw2=T} OR {throw1=T, throw2=H}

Only one way to obtain this: both throws give T

- Are they equally likely?
  - No!

Probability Trees

- **Tree diagram** may be used to represent a probability space
- Four equally likely outcomes:



So the answer to our problem

- There are actually 4 equally likely outcomes when you flip a coin twice.
  - HH, HT, TH, TT
- What's the probability of getting 2 heads?
  - P(HH) = ¼ = 0.25
- What's the probability of getting head and tail?
  - P(HT OR TH) = 2/4 = 0.5

The stability of the relative frequency

Laplace’s principle is not always true!

Teaser: violations of Laplace's principle

- You randomly pick out a word from a corpus containing 1000 words of English text.
- Are the following equally likely:
  - word will contain the letter *e*
  - word will contain the letter *h*

E is the most frequent letter in Engish orthography

- What about:
  - word will be *audacity*
  - word will be *the*

The is far more frequent than *audacity*

- In both cases, prior knowledge or experience gives good reason for assuming unequal likelihood of outcomes.

Unequal likelihoods

- When the Laplace Principle is violated, how do we estimate probability?
  - We often need to rely on prior experience
  - In general, for language events, P is unknown
  - We need to estimate P
  - By looking at evidence about what P must be based on a sample of data
- Example:
  - In a big corpus, count the frequency of *e* and *h*
  - Take a big corpus, count the frequency of *audacity* vs. *the*
  - Use these estimates to predict the probability on a new 1000-word sample.

Example

- Suppose that, in a corpus of 1 million words:
  - C(*the*) = 50,000
  - C(*audacity*) = 2

- Based on frequency, we estimate probability of each outcome of interest:
  - frequency / total
  - P(*the*) = 50,000/1,000,000 = 0.05
  - P(*audacity*) = 2/1,000,000 = 0.000002

Interpretation of probability

- The core assumption in statistical NLP, where we estimate probabilities based on frequency in corpora:

  Given that a certain event of interest occurs ***m*** times in ***n*** identical situations, its probability is **m/n**

- Stability of relative frequency:
  - we tend to find that if ***n*** is large enough, the relative frequency of an event ***m*** is quite stable across samples

- In language, this may not be so straightforward:
  - word frequency depends on text genre
  - word frequencies tend to “flatten out” the larger your corpus

Corpus Based Statistical NLP

- Based on counting in large corpora and developing different kinds of models
- It has become very popular these days
- This approach only became possible with the Web
  - Google Translate: works because its seen tons of examples
  - Spelling Correction: also works because its seen tons of examples of words in context
- Started out with simple word and POS based approaches, now NLP challenges more to do with meaning

More Probability…

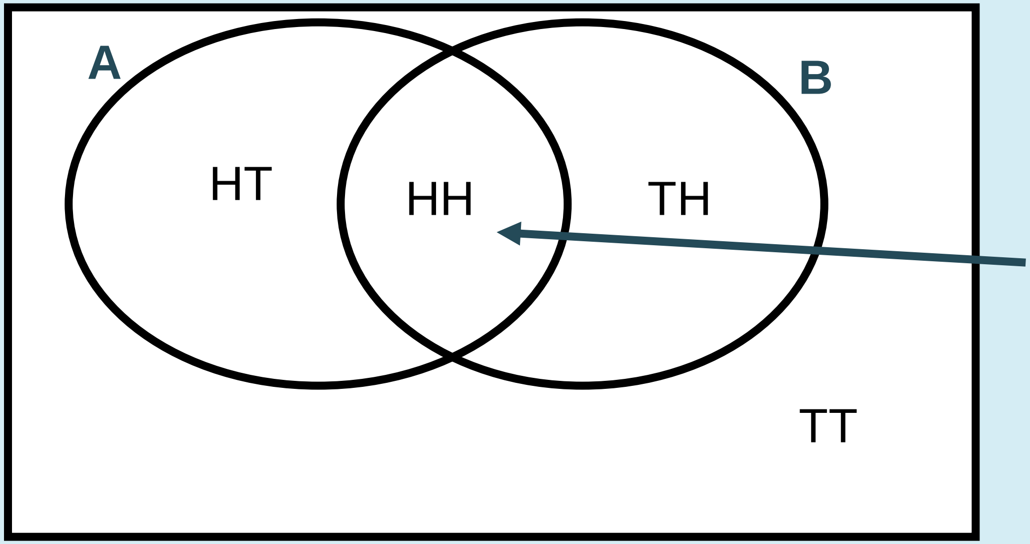
The addition rule

- You flip 2 coins, what's the probability that you get at least one head?
- The first intuition:
  - P(H on first coin) + P(H on second coin)
  - P(H) = 0.5 in each case, so the total P is 1
  - What's wrong?

  - We're counting the probability of getting two heads twice! ☺
  - Possible outcomes: {HH, HT, TH, TT}
  - The P(H) = 0.5 for the first coin includes the case where our outcome is HH. If we also assume P(H) = 0.5 for the second coin, this too includes the case where our outcome is HH. So, we count HH twice.

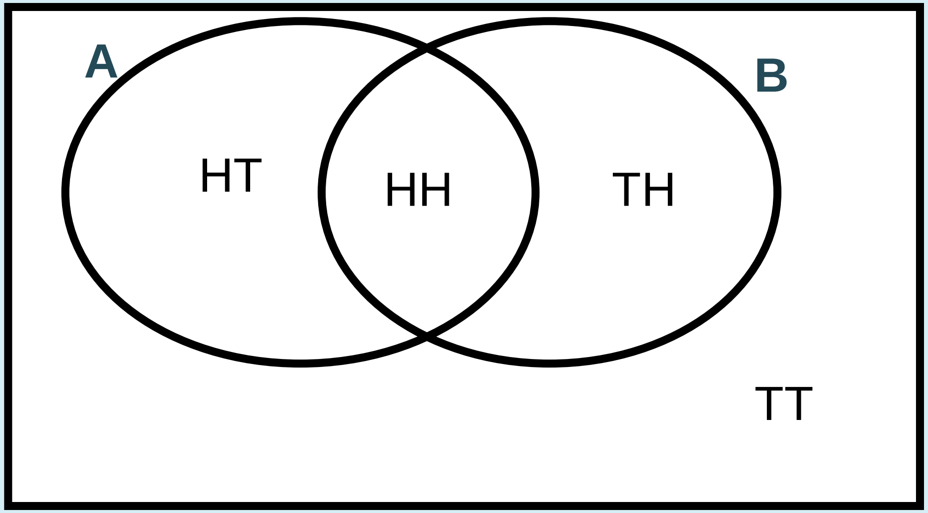
Venn diagram representation

Set A represents outcomes where first coin = H.
Set B represents outcomes where second coin = H
A and B are our outcomes of interest. (TT is not in these sets)



A and B have a nonempty intersection, i.e. there is an event which is common to both. Both contain two outcomes, but the total unique outcomes is not 4, but 3.

Some notation



$A \cup B$ = events in A **and** events in B

$A \cap B$ = events which are in **both** A and B

$P(A \cup B)$ = probability that something which is **either in** A OR B occurs

$P(A \cap B)$ = probability that something which is **in both** A AND B occurs

Addition rule

- To estimate probability of A OR B happening, we need to remove the probability of A AND B happening, to avoid double-counting events.

$$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$

- In our case:
  - P(A) = 2/4
  - P(B) = 2/4
  - P(A AND B) = ¼
  - P(A OR B) = 2/4 + 2/4 – ¼ = ¾ = 0.75

Conditional Probability & Independence

Prior knowledge

Sometimes, an estimation of the probability of an event is affected by what is known (which changes the sample space)

- Example:
  - A box of 10 candies
  - **5 red and sweet**, **3 green and sour**, **2 green and sweet**
  - You close your eyes and randomly pick a candy, what is the probability that it is sweet?
    - 7/10 = 0.7
  - After you pick the candy, you open your eyes and see it is red, now what is the probability that it is sweet?
    - 5/5 = 1

Part-of-speech tagging example

- One of the classic tasks in statistical NLP
  - Assign a label indicating the grammatical category to every word in a corpus of running text.
- Statistical POS taggers are first trained on data that has been previously annotated. Yields a **language model**.
- Language models vary based on the n-gram window (size of the sequence):
  - Unigrams: probability based on tokens (a lexicon)

    e.g. input = the_DET tall_ADJ man_NN

    model represents the probability that the word *man* is a noun (it could also be a verb)
  - Bigrams: probabilities across a span of 2 words

    model represents the probability that a DET is followed by an adjective, adjective is followed by a noun, etc.
  - Also: Trigrams, Quadrigrams, etc.

POS tagging continued

- Suppose we've trained a tagger on the annotated data. It has:
  - a lexicon of unigrams:
    - P(the=DET), P(man=NN), etc
  - a bigram model
    - P(DET is followed by ADJ), etc
  - Assume we've trained it on a large input sample.
  - We now feed it a new phrase:
    - *the audacious alien*
- Our tagger knows that the word *the* is a DET, but it's never seen the other words.
- It can:
  - Make a wild guess (not very useful!)
  - Alternative: **estimate the probability** that *the* is followed by an ADJ, and that an ADJ is followed by a NOUN

Prior knowledge revisited

- Given that I know that *the* is DET, what's the probability that the following word *audacious* is ADJ?

  - This is very different from asking what's the probability that *audacious* is ADJ out of context.

  - We have **prior knowledge** that DET has occurred. This can significantly change the estimate of the probability that *audacious* is ADJ.

  - We therefore distinguish:
    - **Prior probability**: "Naïve" estimate based on long-run frequency
      - The unconditional probability that is assigned before any relevant evidence is taken into account.
    - **Posterior probability**: probability estimate based on prior knowledge after some observation
      - The **conditional probability** that is assigned after the relevant evidence or background is taken into account.

Conditional Probability

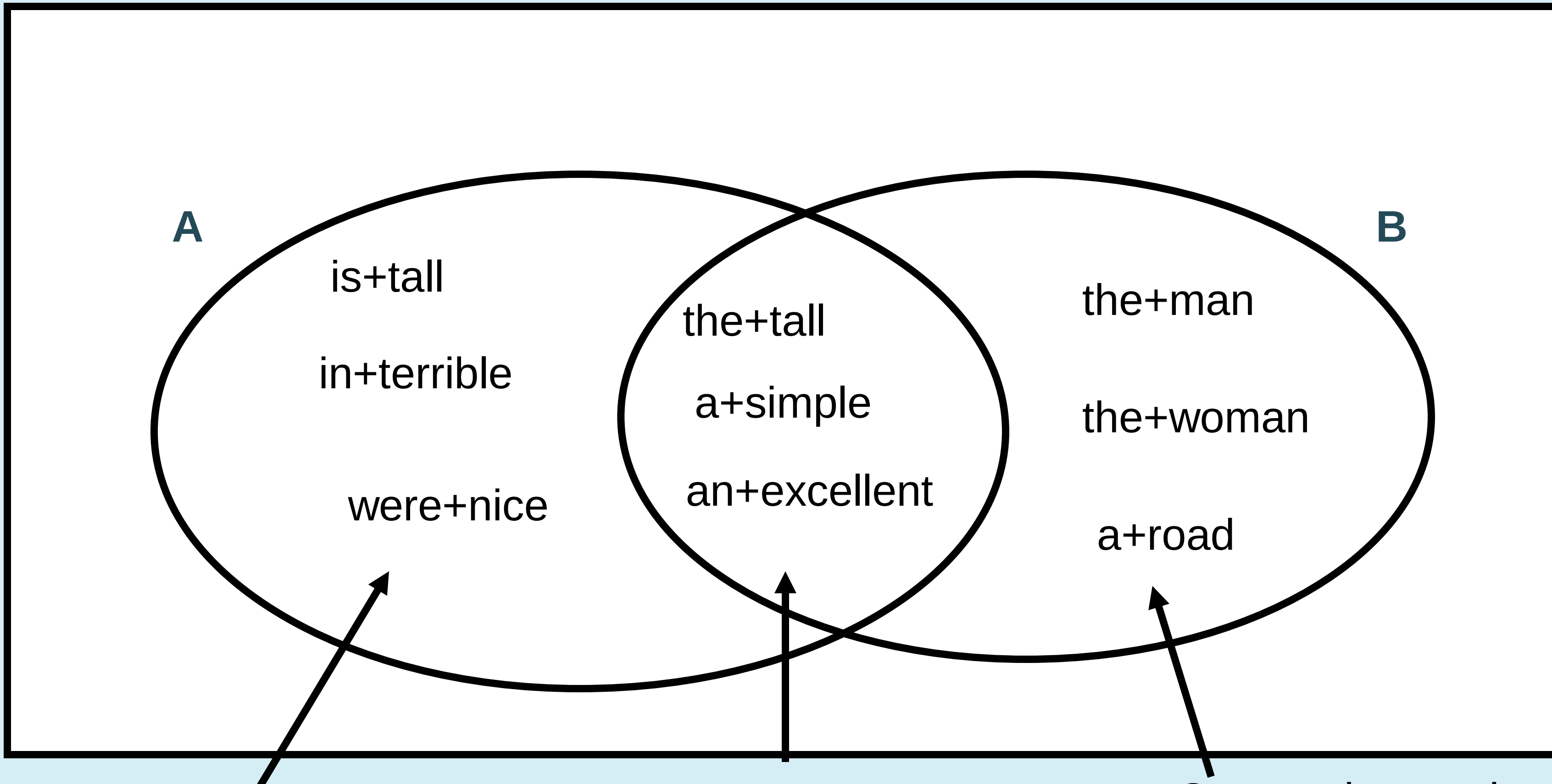
- In our example, we were estimating:
  - P(ADJ|DET) = probability of ADJ given DET
  - P(NN|ADJ) = probability of NN given ADJ
  - …
- In general:
  - The conditional probability P(A|B) is the probability that A occurs, given that we know that B has occurred

Example continued

- If I've just seen a DET, what's the probability that my next word is an ADJ?

- Need to take into account two events:
  - A: occurrences of ADJ in our training data
    - VV+ADJ (*was beautiful*), PP+ADJ (*with great concern*), DET+ADJ etc
  - B: occurrences of DET in our training corpus
    - DET+N (*the man*), DET+V (*the loving husband*), DET+ADJ (*the tall man*)

Venn Diagram representation of the bigram training data



Cases where w is ADJ NOT preceded by DET

Cases where *w* is a *DET* followed by *ADJ*

Cases where w is a DET NOT followed by ADJ

Estimation of conditional probability

- Intuition:
  - P(A|B) is a ratio of the chances that both A and B happen, divided by the chances of B happening alone

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

  - P(ADJ|DET) = P(DET+ADJ) / P(DET)

Another example

- If we throw a dice, what's the probability that the number we get is even, **given that the number we get is greater than 3?**
  - A: the number is even → {2, 4, 6}
  - B: the number is greater than 3 → {4, 5, 6}
  - A ∩ B : the number is even and is greater than 3 → {4, 6}
- $P(A|B) = P(A \cap B) / P(B)$
- $P(\text{even}|>3) = P(\text{even} \cap >3)/P(>3)$

  $= (2/6) / (3/6)$

  $= 2/3$
- **Note the difference from simple, prior probability.**

The Multiplication Rule

Multiplying probabilities

- Often, we're interested in switching the conditional probability estimate around.
  - Suppose we know P(A|B) or P(B|A)
  - We want to calculate P(A AND B)
  - For both A and B to occur, they must occur in some sequence

Estimating P(A AND B)

$$P(A \cap B) = P(A)P(B \mid A)$$

Probability that both A and B occur

Probability of A

happening overall

Probability of B happening given that A has happened

Multiplication rule: example 1

- We have a standard deck of 52 cards
- What's the probability of pulling out two aces in a row?
  - Standard deck has 4 aces
- Let A1 stand for "an ace on the first pick", A2 for "an ace on the second pick"
- We're interested in P(A1 AND A2)

Example 1 continued

- $P(A1 \text{ AND } A2) = P(A1)P(A2|A1)$
- $P(A1) = 4/52$
  - (since there are 4 aces in a 52-card pack)
- If we do pick an ace on the first pick, then we diminish the odds of picking a second ace (there are now 3 aces left in a 51-card pack).
  - $P(A2|A1) = 3/51$
- Overall: $P(A1 \text{ AND } A2) = (4/52)\,(3/51) = .0045$
  - in the event of A1, the chances of A2 are diminished
  - the multiplication rule takes this into account

Some observations

- In this example, the two events are not independent of each other
  - occurrence of one affects likelihood of the other
  - e.g. drawing an ace first diminishes the likelihood of drawing a second ace
  - Sampling without replacement

  - if we put the ace back into the pack after we've drawn it, then we have sampling with replacement
    - In this case, the probability of one event doesn't affect the probability of the other.

Extending the multiplication rule

- The logic of the “A AND B” rule is:
  - Both conditions, A and B have to be met
  - A is met a fraction of the time
  - B is met a fraction of the times that A is met
- Can be extended indefinitely
  - E.g. chances of drawing 4 straight aces from a pack
  - P(A1 & A2 & A3 & A4)

=

P(A1) P(A2|A1) P(A3|A1 & A2) P(A4|A1 & A2 & A3)

Extending The Addition Rule & The Subtraction Rule

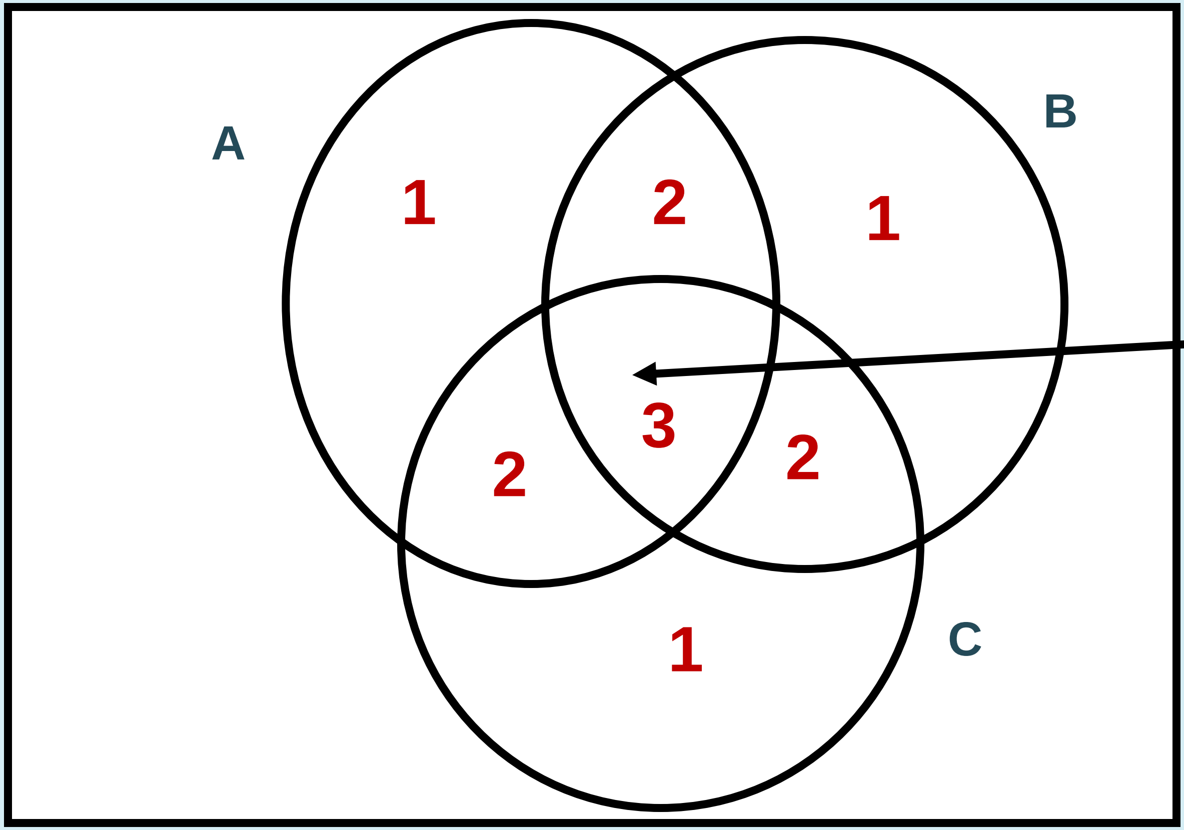
Extending the addition rule

- It was easy to extend the multiplication rule.
- Extending the addition rule isn't so easy. We need to correct for double-counting events.

$$
\begin{aligned}
P(A \cup B \cup C) = \quad & P(A) + P(B) + P(C) \\
& - P(A \cap B) - P(A \cap C) - P(B \cap C) \\
& + P(A \cap B \cap C)
\end{aligned}
$$

Example

- P(A OR B OR C)



Once we've discounted the 2-way intersection of A and B, etc, we need to recount the 3-way intersection!

Subtraction rule

- Fundamental underlying observation:

$$P(A) = 1 - P(\overline{A})$$

- E.g. Probability of getting at least one head in 3 flips of a coin (a three-set addition problem)
  - Can be estimated using the observation that:
  - P(Head out of 3 flips) = 1-P(no heads) = 1-P(3 tails)

Many tasks in NLP use these probability estimates