CS447: Natural Language Processing

*http://courses.grainger.illinois.edu/cs447*

# Lecture 2: Corpora, Words, Tokenization

Julia Hockenmaier

*juliahmr@illinois.edu*

ILLINOIS

# Today's class

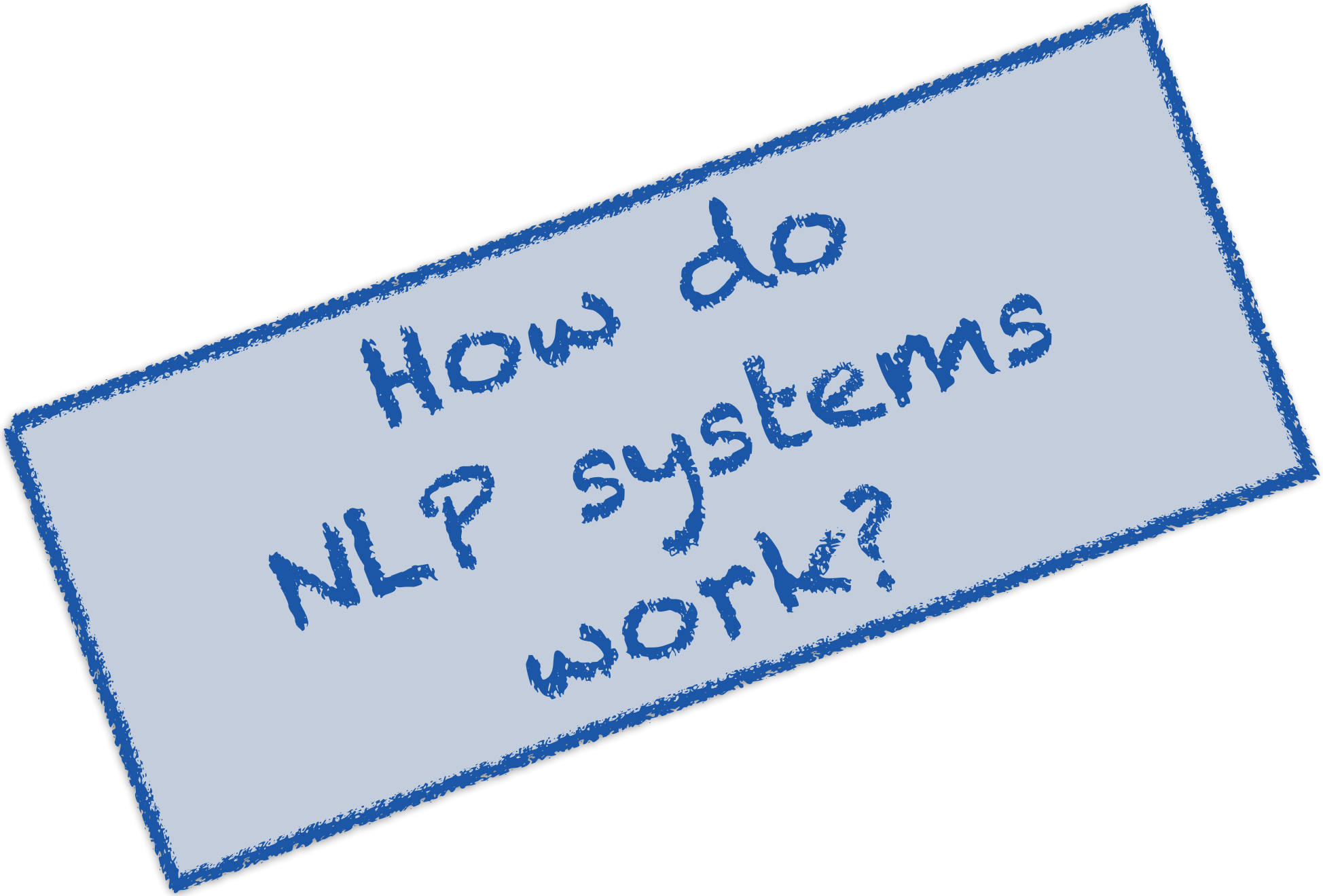
Wrap-up from last lecture: **The NLP pipeline**
How do (traditional) NLP systems work

**Corpora:** the importance of (having and understanding) appropriate data

**Words and Tokenization**

**Additional key concepts:**

Ambiguity, CoverageType/token distinction


How do
NLP systems
work?

# The traditional NLP pipeline

A (traditional) NLP system may use some or all of the following steps:

**Tokenizer/Segmenter**

to identify words and sentences

**Morphological analyzer/POS-tagger**

to identify the part of speech and structure of words

**Word sense disambiguation**

to identify the meaning of words

**Syntactic/semantic Parser**

to obtain the structure and meaning of sentences

**Coreference resolution**

to keep track of the various entities mentioned

# What does it take to understand text?

死亡谷测得54.4摄氏度高温
美国加州名胜或破世界纪录

Çavuşoğlu'ndan Atina'ya uyarı: Bazı ülkelerin dolduruşuna gelip, kendinizi riske atmayın

รอยัลลิสต์มาร์เก็ตเพลส: เฟซบุ๊กเตรียมดำเนินทางการกฎหมายกับรัฐบาลไทย หลังบังคับบล็อกการเข้าถึงกลุ่มปิดที่พูดคุยเกี่ยวกับราชวงศ์

ኣብ ሳዋ ዝወሃብ መበል 12 ክፍሊ ትምህርቲ ክቋረጽ ጎስጓስ ይካየድ ኣሎ

Qabiyyeen xalayaa dhimma Obbo Lidatu Ayyaaloorratti MM Abiyyiif barraa'e maali?

'Dim angen cau tafarndai a bwytai i ailagor ysgolion'

# Task: Tokenization/segmentation

死亡谷测得54.4摄氏度高温
美国加州名胜或破世界纪录

รอยัลลิสต์มาร์เก็ตเพลส: เฟซบุ๊กเตรียมดำเนินทางการกฎหมายกับรัฐบาลไทย หลังบังคับบล็อกการเข้าถึงกลุ่มปิดที่พูดคุยเกี่ยวกับราชวงศ์

We need to split text into words and sentences.

- Languages like Chinese or Thai don't have spaces between words.
- Even in English, this cannot be done deterministically: *There was an earthquake near D.C. You could even feel it in Philadelphia, New York, etc.*

NLP task:

What is the ***most likely*** **segmentation/tokenization?**

# Task: Part-of-speech-tagging

***Open the pod door, Hal.***

Verb Det Noun Noun , Name .

***Open the pod door , Hal .***

***open*:**
Verb, adjective, or noun?
- Verb: **open** the door
- Adjective: the **open** door
- Noun: in the **open**

# How do we decide?

We want to know **the most likely tags** $T$
for the sentence $S$

$$\underset{T}{\operatorname{argmax}} P(T|S)$$

We need to **define a statistical model** of $P(T \mid S)$, e.g.:

$$\begin{aligned} \underset{T}{\operatorname{argmax}} P(T|S) &= \underset{T}{\operatorname{argmax}} P(T)P(S|T) \\ P(T) &=_{def} \prod_i P(t_i|t_{i-1}) \\ P(S|T) &=_{def} \prod_i P(w_i \mid t_i) \end{aligned}$$

We need to **estimate the parameters** of $P(T|S)$, e.g.:

$$P(\, t_i = V \mid t_{i-1} = N\,) = 0.312$$

# Disambiguation requires statistical models

**Ambiguity** is a core problem for any NLP task

Statistical models* are one of the main tools to deal with ambiguity.

*A lot of the models (classifiers, structured prediction models) you learn about in a machine learning class can be used for this purpose.
We won't assume you have taken a machine learning class.

These models need to be **trained** (estimated, learned) before they can be **used** (tested, evaluated).

# *"I made her duck"*

What does this sentence mean?

"I made her crouch",
"I cooked duck for her",
"I cooked her [pet] duck (perhaps just for myself)", …

***"duck"***: noun or verb?

***"make"***: "*cook X*" or "*cause X to do Y*" ?

***"her"***: "*for her*" or "*belonging to her*" ?

# Ambiguity in natural language

Language has different kinds of ambiguity, e.g.:

## **Structural** ambiguity

*"I eat sushi **with tuna**"* vs. *"I eat sushi **with chopsticks**"*

"*I saw the man **with the telescope on the hill**"*

## **Lexical (word sense)** ambiguity

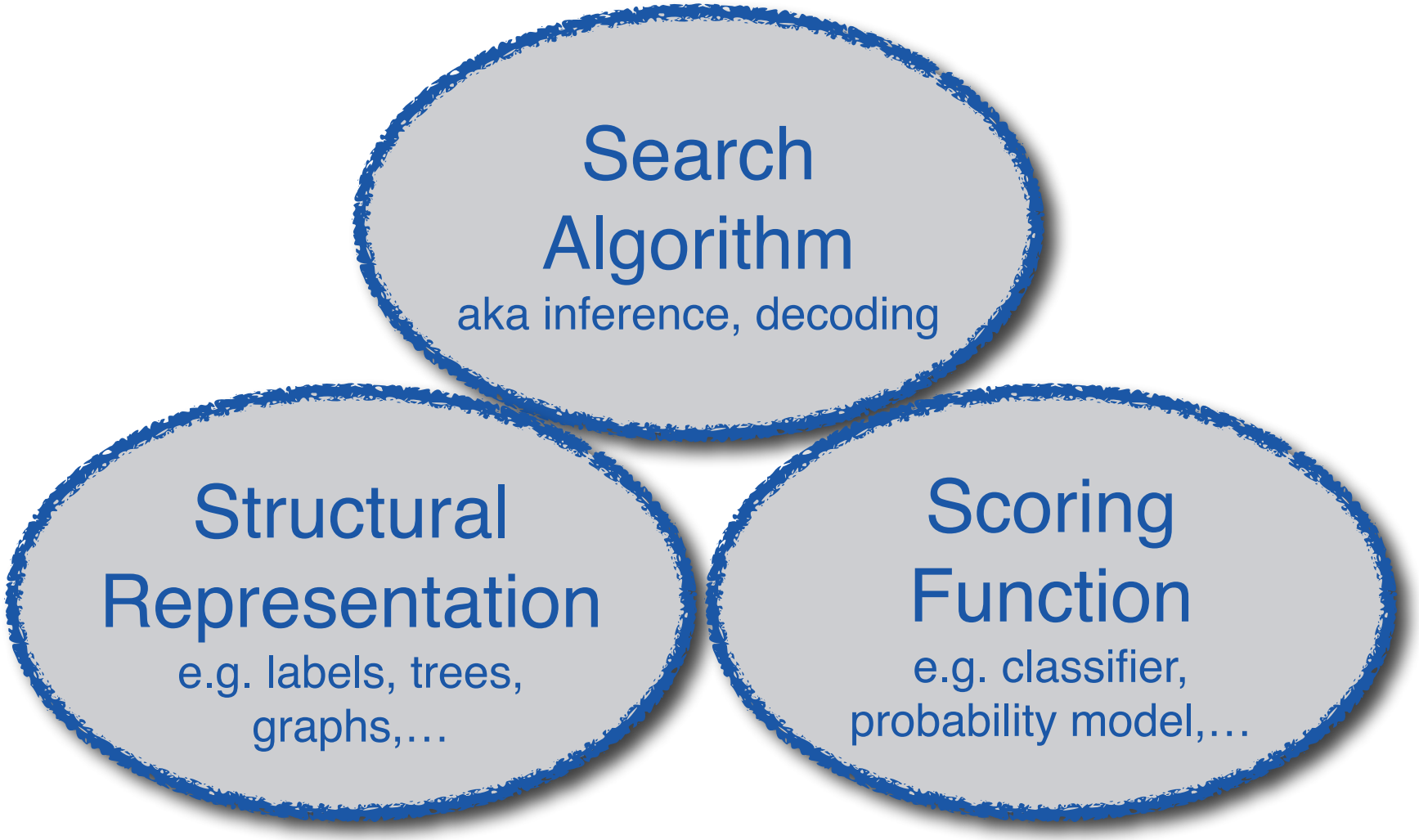
"*I went to the **bank**"*: financial institution or river bank?

## **Referential** ambiguity

"***John** saw **Jim**. **He** was drinking coffee."*

Who was drinking coffee?

# Dealing with ambiguity



# *"I made her duck cassoulet"*

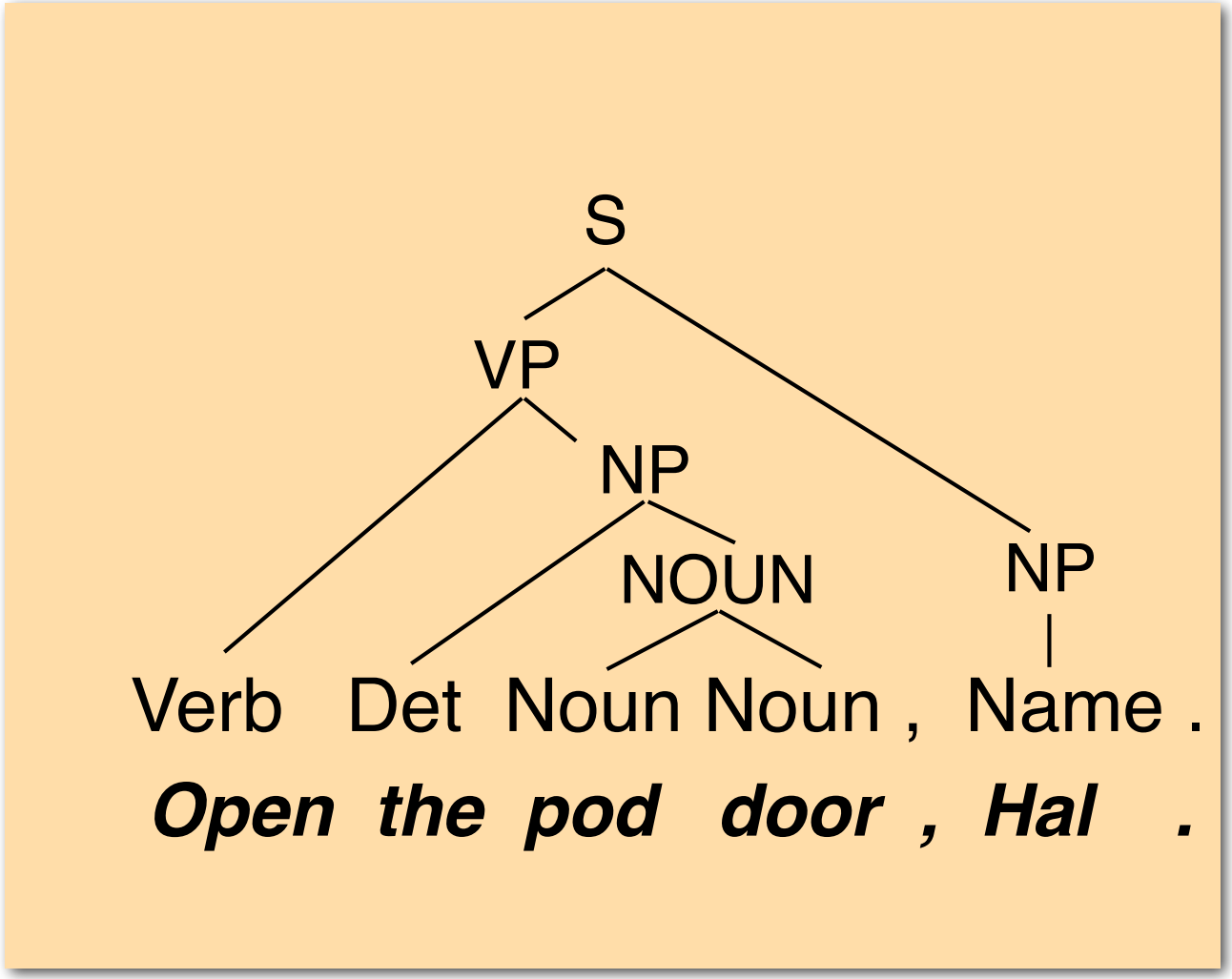
(Cassoulet = a French bean casserole)

The second major problem in NLP is **coverage**:
We will always encounter unfamiliar words and constructions.

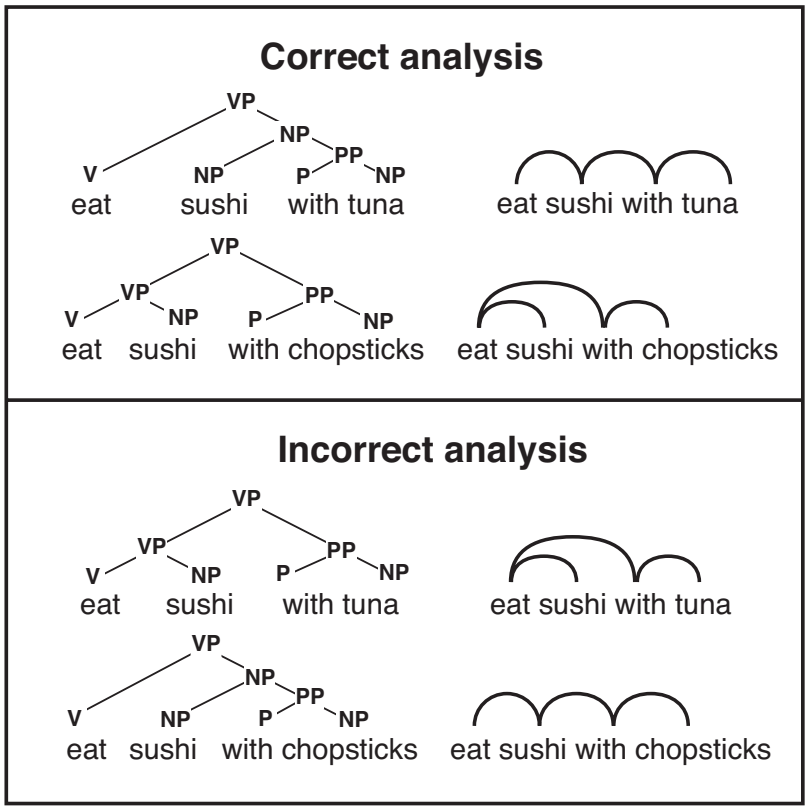
Our models need to be able to deal with this.

This means that our models need to be able to *generalize* from what they have been trained on to what they will be used on.

# Task: Syntactic parsing

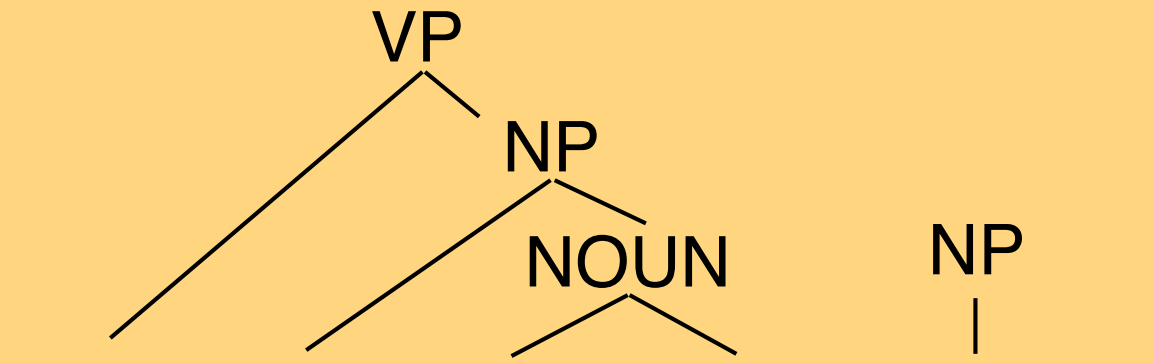


# Observation: Structure corresponds to meaning



# Task: Semantic Analysis

```
∃x∃y(pod_door(x) & Hal(y)
  & request(open(x, y)))
```




***Open the pod door , Hal .***

# Representing meaning

If a natural language understanding system needs to return a symbolic representation (or data structure) of the meaning of text, it needs a pre-defined **meaning representation language**.

**"Deep" semantic analysis:** (Variants of) **formal logic**

$\exists x \exists y(\text{pod\_door}(x) \& \text{Hal}(y) \& \text{request}(\text{open}(x,y)))$

**"Shallow" semantic analysis: Template-filling**
(Often used in information extraction)

- Named-Entity Recognition: identify all organizations, locations, dates,...
- Event Extraction: identify specific events (e.g. 'protest', 'purchase', …)

We also distinguish between

**Lexical semantics** (the meaning of words) and

**Compositional semantics** (the meaning of sentences)

# Understanding texts

> On Monday, John went to Einstein's. He wanted to buy lunch. But the store was closed. That made him angry, so the next day he went to Green Street instead.

Can you answer the following questions?

*Was Einstein's open for lunch on Monday?* [No]

This requires the ability to identify that "*Einstein's*" and "*the store*" refer to the same entity. (**coreference resolution**).

*On which day did John go to Green Street?* [On Tuesday].

This requires the ability to understand the **implicit** information that "the next day" means really "the next day after Monday" (and the knowledge that that is a Tuesday).

# The traditional NLP pipeline

A (traditional) NLP system may use some or all of the following steps:

**Tokenizer/Segmenter**

to identify words and sentences

**Morphological analyzer/POS-tagger**

to identify the part of speech and structure of words

**Word sense disambiguation**

to identify the meaning of words

**Syntactic/semantic Parser**

to obtain the structure and meaning of sentences

**Coreference resolution**

to keep track of the various entities mentioned

# NLP Pipeline: Assumptions

Each step in the NLP pipeline embellishes the input with **explicit information about its linguistic structure**

- POS tagging: Parts of speech of word,
- Syntactic parsing: Grammatical structure of sentence,….

Each step in the NLP pipeline requires **its own explicit ("symbolic") output representation:**

- POS tagging requires a **POS tag set**
  (e.g. NN=common noun singular, NNS = common noun plural, …)
- Syntactic parsing requires **constituent** or **dependency labels**
  (e.g. NP = noun phrase, or nsubj = nominal subject)

These representations should capture **linguistically appropriate generalizations/abstractions**

- Designing these representations requires linguistic expertise

# NLP Pipeline: Shortcomings

Each step in the pipeline relies on a **learned model** that will return the *most likely* representations

- This requires a lot of **annotated training data** for each step
- Annotation is **expensive** and sometimes **difficult** (people are not 100% accurate)
- These models are **never 100% accurate**
- Models make more mistakes if their input contains mistakes

How do we know that we have captured the "***right***" **generalizations** when designing representations?

- Some representations are **easier to predict** than others
- Some representations are **more useful** for the next steps in the pipeline than others
- But we won't know how easy/useful a representation is until we have a model that we can plug into a particular pipeline

# Sidestepping the NLU pipeline

Many current neural approaches for natural language understanding and generation go directly from the raw input to the desired final output.

With large amounts of training data, this often works better than the traditional approach.

— We will soon discuss why this may be the case.

But these models don't solve everything:

— How do we incorporate knowledge, reasoning, etc. into these models?

— What do we do when don't have much training data? (e.g. when we work with a low-resource language)


Corpora
matter!

# What is a corpus?

A corpus (plural: corpora) is a collection of natural language data (i.e. a data set)

**Raw corpora** have only minimal (or no) processing:

- Sentence boundaries may or may not be identified.
- There may or may not be metadata
- Typos (written text) or disfluencies (spoken language) may or may not be corrected.

**Annotated corpora** contain some labels (e.g. POS tags, sentiment labels), or linguistic structures (e.g syntax trees, semantic interpretations), etc.

# Corpus information (datasheets)

**Basic statistics:** corpus size, etc.

**Intellectual Property** (copyright, licenses)

What **language (variety)** is represented?

English, British English, African American English, French,

What **genre** is represented in this corpus?

Newswire, social media, fiction, poetry, essays, conversations,

When and in what **situation** was the text produced?

Web crawls vs. conversations recorded in a lab, vs. crowdsourced tasks (e.g image descriptions), etc.

Speaker/Author **demographics**?

**Motivation/purpose** for data collection

# Corpus information (datasheets)

**Annotation Process**

Annotation guidelines,

Annotator demographics and training

Quality assessment (inter-annotator agreement, adjudication)


Words and
Tokenization

# Tokenization: Identifying word boundaries

Text is just a **sequence of characters**:

> Of course he wants to take the advanced course too. He already took two beginners' courses.

How do we split this text into **words** and **sentences**?

> [ [Of, course, he, wants, to, take, the, advanced, course, too, .],
> [He, already, took, two, beginners', courses, .]]

# How do we identify the words in a text?

For a language like English, this *seems* like a really easy problem:

A word is any sequence of alphabetical characters between whitespaces that's not a punctuation mark?

That works to a first approximation, but…

… what about abbreviations like *D.C.*?
… what about complex names like *New York*?
… what about contractions like *doesn't* or *couldn't've*?
… what about *New York-based* ?
… what about names like *SARS-Cov-2*, or R2-D2?
… what about languages like Chinese that have no whitespace, or languages like Turkish where one such "word" may express as much information as an entire English sentence?

# Words aren't just defined by blanks

## Problem 1: Compounding

"ice cream", "website", "web site", "New York-based"

## Problem 2: Other writing systems have no blanks

*Chinese:* 我开始写小说 = 我 开始 写 小说
*I start(ed) writing novel(s)*

## Problem 3: Contractions and Clitics

English: "doesn't" , "I'm" ,

Italian: "dirglielo" = dir + gli(e) + lo
*tell + him + it*

# Tokenization Standards

Any actual NLP system will assume a particular tokenization standard.

Because so much NLP is based on systems that are trained on particular corpora (text datasets) that everybody uses, these corpora often define a de facto standard.

**Penn Treebank 3 standard:**

**Input:**

"The San Francisco-based restaurant,"
they said, "doesn't charge $10".

**Output:**

“_ The _ San _ Francisco-based _ restaurant _ , _” _
they_ said_ ,_ "_ does _ n’t _ charge_ $_ 10 _ " _ . _

# Aside: What about sentence boundaries?

How can we identify that this is two sentences?

Mr. Smith went to D.C. Ms. Xu went to Chicago instead.

Challenge: punctuation marks in abbreviations (Mr., D.C, Ms,...)

[It's easy to handle a small number of known exceptions,
but much harder to identify these cases in general]

See also this headline from the NYT (08/26/20):

Anthony Martignetti ('Anthony!'), Who Raced Home for Spaghetti, Dies at 63

How many sentences are in this text?

"The San Francisco-based restaurant," they said, "doesn't charge $10".

Answer: just one, even though "they said" appears in the middle of another sentence.

Similarly, we typically treat this also just as one sentence:

They said: "The San Francisco-based restaurant doesn't charge $10".

# Spelling variants, typos, etc.

The same word can be written in different ways:

- with different **capitalizations**:
  lowercase "cat" (in standard running text)
  capitalized "Cat" (as first word in a sentence, or in titles/headlines),
  all-caps "CAT" (e.g. in headlines)
- with different **abbreviation** or **hyphenation** styles:
  US-based, US based, U.S.-based, U.S. based
  US-EU relations, U.S./E.U. relations, …
- with **spelling variants** (e.g. regional variants of English):
  labor vs labour, materialize vs materialise,
- with **typos** (teh)

Good practice: Be aware of (and/or document) any normalization (lowercasing, spell-checking, …) your system uses!

# Counting words: tokens vs types

When counting words in text, we distinguish between word **types** and word **tokens**:

— The **vocabulary** of a language
is the set of (unique) word types:
V = {a, aardvark, …., zyzzva}

— The **tokens** in a document include all occurrences of the word types in that document or corpus
(this is what a standard word count tells you)

— The **frequency** of a word (type) in a document
= the number of occurrences (tokens) of that type

# How many different words are there in English?

How large is the **vocabulary** of English
(or any other language)?

**Vocabulary size** = the number of distinct word types

Google N-gram corpus: 1 trillion tokens,
13 million word types that appear 40+ times

If you count words in text, you will find that…

…a few words (mostly closed-class) are very frequent
(the, be, to, of, and, a, in, that,...)

… most words (all open class) are very rare.

… even if you've read a lot of text,
you will keep finding words you haven't seen before.

**Word frequency**: the number of occurrences of a word type in a text (or in a collection of texts)

# Vocabulary size and corpus size

The number of distinct word types (vocabulary size) increases with the size of the corpus

**Herdan's Law/Heap's Law:**

A corpus of $N$ tokens has a vocabulary of size

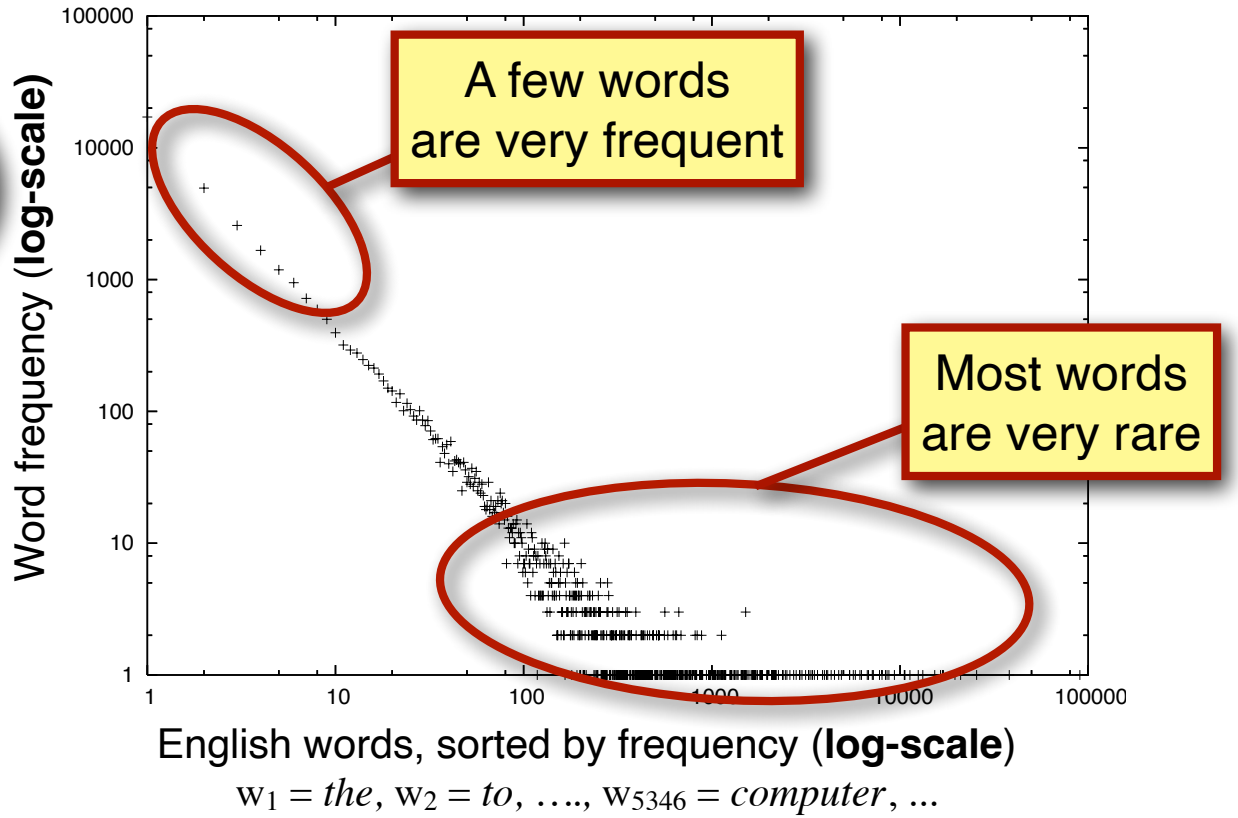
$$|V| = kN^{\beta}$$

for positive constants $k$ and $0 < \beta < 1$

# Zipf's law: the long tail

How many words occur once, twice, 100 times, 1000 times?

the *r*-th most common word $w_r$ has $P(w_r) \propto 1/r$




In natural language:

A small number of events (e.g. words) occur with high frequency

A large number of events occur with very low frequency

# Implications of Zipf's Law for NLP

**The good:**

Any text will contain a number of words that are very **common**.

We have seen these words often enough that we know (almost) everything about them. These words will help us get at the structure (and possibly meaning) of this text.

**The bad:**

Any text will contain a number of words that are **rare**.

We know *something* about these words, but haven't seen them often enough to know everything about them. They may occur with a meaning or a part of speech we haven't seen before.

**The ugly:**

Any text will contain a number of words that are **unknown** to us.

We have *never* seen them before, but we still need to get at the structure (and meaning) of these texts.

# Dealing with the bad and the ugly

Our systems need to be able to **generalize** from what they have seen to unseen events.

There are two (complementary) approaches to generalization:

— **Linguistics** provides us with insights about the rules and structures in language that we can exploit in the (symbolic) representations we use

E.g.: a finite set of grammar rules is enough to describe an infinite language

— **Machine Learning/Statistics** allows us to learn models (and/or representations) from real data that often work well empirically on unseen data

E.g. most statistical or neural NLP

# How do we represent words?

## Option 1: Words are **atomic symbols**

- Each (surface) word form is its own symbol
- Add some generalization by mapping different forms of a word to the same symbol
  - **Normalization:** map all variants of the same word (form) to the same canonical variant (e.g. lowercase everything, normalize spellings, perhaps spell-check)
  - **Lemmatization**: map each word to its lemma (esp. in English, the lemma is still a word in the language, but lemmatized text is no longer grammatical)
  - **Stemming**: remove endings that differ among word forms (no guarantee that the resulting symbol is an actual word)

# How do we represent words?

## Option 2: Represent the **structure** of each word

"books" => `"book N pl"` (or `"book V 3rd sg"`)

This requires a **morphological analyzer** (more later today)

The output is often a lemma (`"book"`)
plus morphological information (`"N pl"` i.e. plural noun)

This is particularly useful for highly inflected languages, e.g. Czech, Finnish, Turkish, etc. (less so for English or Chinese):

In Czech, you might need to know that *nejnezajímavějším*
is a regular, feminine, plural, dative adjective in the superlative.

# How do we represent unknown words?

Many NLP systems assume a fixed vocabulary, but still have to handle **out-of-vocabulary (OOV)** words.

## Option 1: the **UNK** token

Replace all rare words (with a frequency at or below a given threshold, e.g. 2, 3, or 5) in your training data with an UNK token (UNK = "Unknown word").

Replace *all* unknown words that you come across after training (including rare training words) with the same UNK token

## Option 2: **substring-based** representations

[often used in neural models]

Represent (rare and unknown) words ["Champaign"] as sequences of characters ['C', 'h', 'a',…,'g', 'n'] or substrings ["Ch", "amp", "ai", "gn"]

Byte Pair Encoding (BPE): learn which character sequences are common in the vocabulary of your language, and treat those common sequences as atomic units of your vocabulary