

# Analyzing Natural Language

```
import natural from "natural"
```

"I work with a deep neural network to develop a powerful natural language processing algorithm"

Lecture 9

EECS 498: Winter 2020

# Review: Scoping

- Scoping review grades released (good job)
- Sprint Review 1 soon
  - Don't worry about timing (we understand it's a crunch)
  - **Expectation:** at least a demo in the platform showing examples working
  - **Hopeful:** simple BLS integration
    - (protip: don't use assembly for BLS)

*You may as well give up now.*

## Web Development With Assembly



O'REILLY®

*Bob Johnson  
with His Therapist*

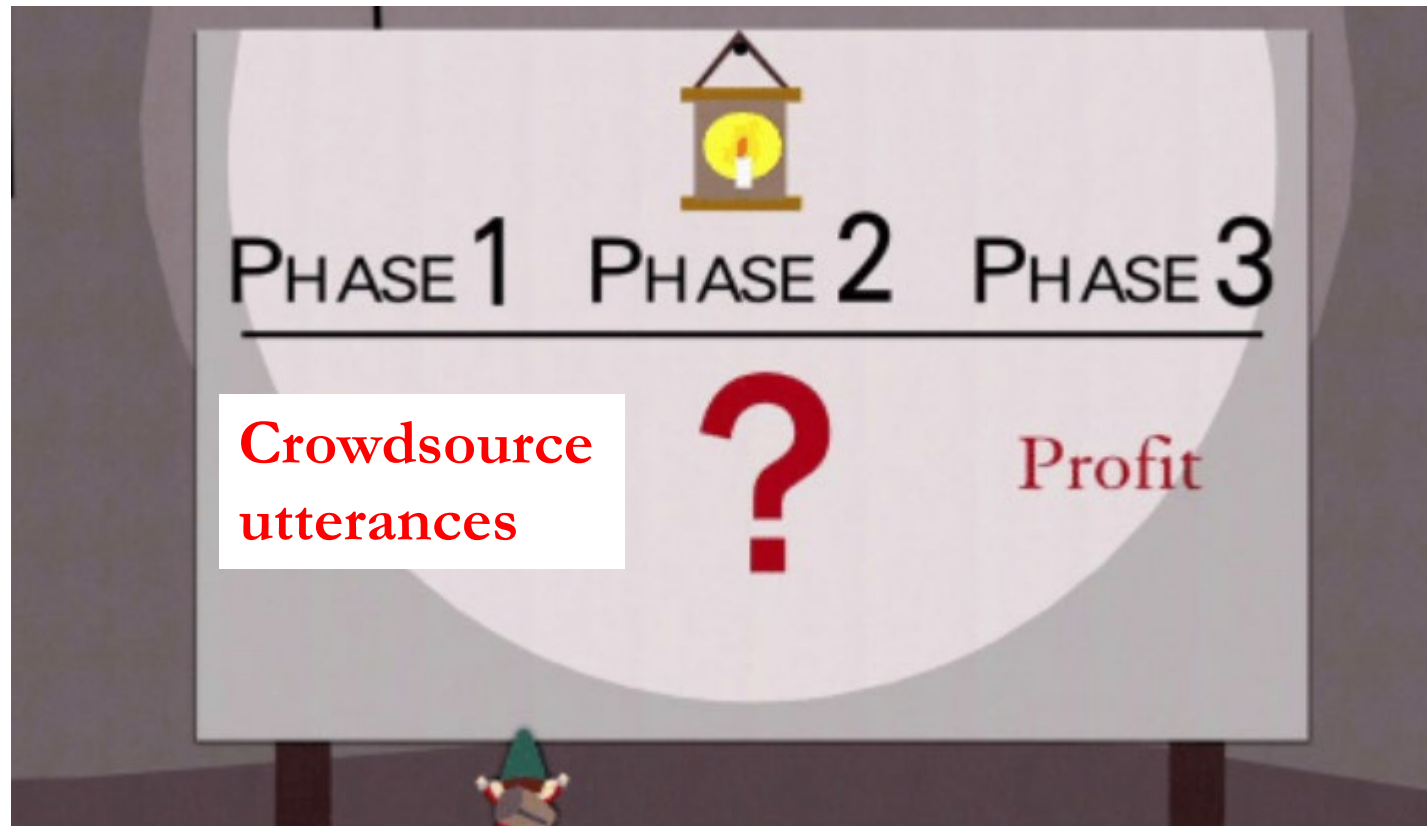
# Review: Natural Language

- Goal: **Empower** software to **understand** spoken **language**
- How? Use **Deep Learning** to enable **data-driven** understanding
  - Deep Neural Networks achieve **high performance**
    - **Accurate** classification, slot extraction, and mapping
    - **Fast** inference time (e.g., after training)
  - Deep learning requires **lots of training**
    - Can take hours, weeks, months depending on how many examples are required

# One-Slide Summary: Language Preprocessing

- We want to **leverage deep learning** to do classification and slot extraction
- We need to *prepare* data so they are **consumable** examples
- What do **examples** look like?
  - Intent classification: ( **utterance, intent\_class** ) tuples
  - Slot extraction: ( **word, part\_of\_speech** ) and ( **word, slot\_name** ) tuples
- We **tokenize** inputs by splitting on spaces, converting to lowercase, removing punctuation, identifying stop words, etc.
- We **stem** tokens so that verb tense or quantity don't form **ambiguities**
- We **model** language using statistical analyses to predict likelihoods of words occurring

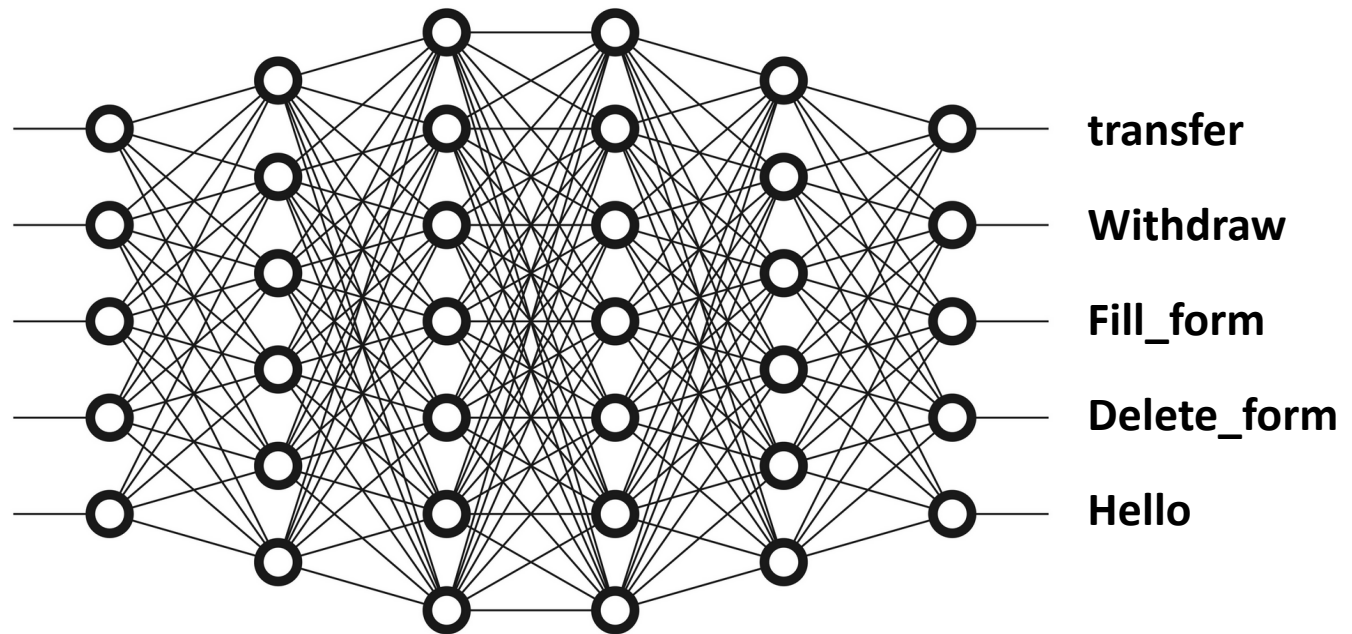
# How do we prepare data for use in AI?



# Language Preprocessing: Why?

- **Curating** and **cleaning** data is critically important
- Statistical models, classical machine learning, and deep learning are all based on the same type of **data** as inputs

“Do my taxes.”



# Tokenization

- First, consider how natural language is formed
  - **Words** are atomic semantic **units**
- **Tokenization** is the process of turning a **sequence of words** to extract semantic objects called **tokens**
  - **TL;DR** split on the spaces
- **Not** letter-by-letter
  - (for now)



```
static int IsNegative(float arg)
{
    char*p = (char*) malloc(20);
    sprintf(p, "%f", arg);
    return p[0]=='-';
}
```

# Tokenization: other considerations

- Mostly convert things to **lowercase**
- Some algorithms will throw away **punctuation**, others will keep it
  - In ASR systems, you won't get punctuation anyway
    - btw: Recovering punctuation is an open research problem
- Input: "Last summer, I went to New York."
- Output: [ 'last', 'summer', 'i', 'went', 'to', 'new', 'york' ]



# Tokenization: Sanitizing data

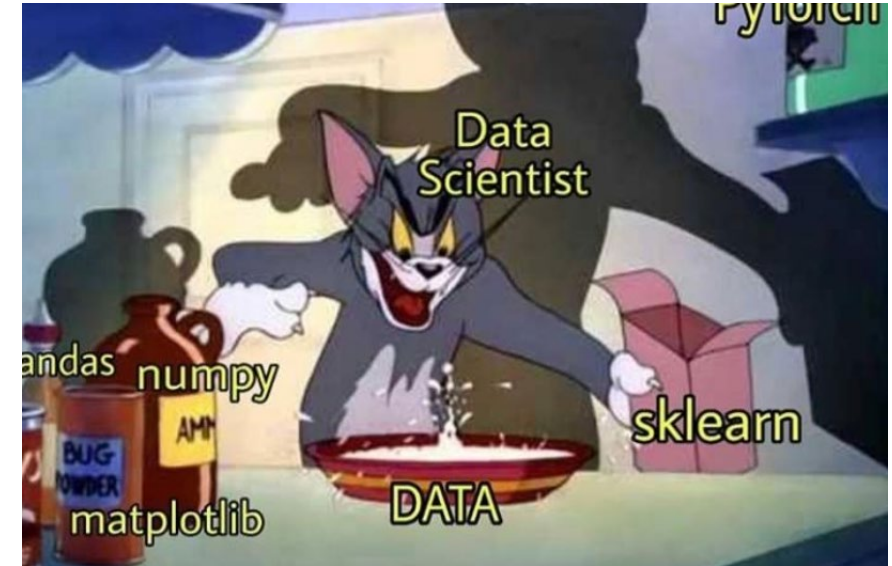
- Scraping data from online sources can yield a wealth of information
  - But! What if the data isn't just words (or words in one language)?
    - “His nickname is 金链哥”
    - “I <strong>really</strong> like this picture.”
- Usually implement tokenization with **regular expressions**
  - Specify constraints of language... for English, a “token” could be:
    - [a-z]+, possibly [A-Za-z]+, or even [A-Za-z0-9-]+
    - (just import re)

# After tokenization: First thoughts...

- Can we just **represent** words as **sequences** of numbers?
  - e.g., ASCII codes
- Hello -> [ 72, 101, 108, 108, 111 ]
- Hi -> [ 72, 105 ]
- Greetings -> [ 71, 114, 101, 101, 116, 105, 110, 103, 115 ]
- How do we tell if words are related?
- Misspellings, capitalization?

# Semantics

- “I went **running** yesterday.”
- “**We** went on a **run** yesterday.”
- “**They ran** all day yesterday.”



- “Running” is pretty different from “ran” by characters...
  - a -> u, extra n, suffix -ing...
  - But aren’t they the same?
- Probably need a bit more sophistication than just letters

# Stemming

- **Morphological normalization** is the process of eliminating variety of words that mean the same thing
  - We want the “stems” of words
- “**I** went **running** yesterday.” -> I went run yesterday
- “**We** went on a **run** yesterday.” -> We went on a run yesterday
- “**They** **ran** all day yesterday.” -> They run all day yesterday
- Gerunds removed (e.g., -ing)
- Tense change (e.g., ran -> run)

# Stemming algorithms

- Different stemmers produce different results
  - Language-dependent (e.g., Chinese does not really need stemming)
  - Some stemmers are more or less aggressive
- **Porter Stemmer:**
  - Insight: represent words as sequences of C or V for Consonants/Vowels
    - The OG stemmer. 5 rules, sometimes causes **under-stemming**
  - Connect
    - Connections, Connected, Connecting, Connection -> connect
  - Friend
    - Friends, friendly -> friend; friendship -> friendship (potential understemming)
  - Alumnus (classic under-stemming example)
    - Alumnus, alumni, and alumnae are not stemmed by Porter

# Stemming Algorithms

- **Lancaster Stemming**

- 120 aggressive rules: heavier, **over-stemming** possible

- Connect

- Connections, connected, connecting -> connect (same as Porter)

- Friend

- Friendship, friends, friendly -> friend (possible over-stemming on friendship)

- Universe (classic over-stemming example)

- Universal, university -> univers

# Stop words

- English has a lot of connecting words. Do we need them all?
  - “I went **running** yesterday.” -> went run yesterday
  - “**We** went on a **run** yesterday.” -> went run yesterday
  - “**They ran** all day yesterday.” -> run day yesterday
- **remove** subjects (“I”, “they”) and function words (“the”, “a”, etc.)
- Drawback: *intent* may change based on stop words
  - e.g., “put money in *my* checking account” vs. “put money in *their* checking account”

# Lemmatization



- What about parts of speech?
  - e.g., English is notoriously arbitrary about verb conjugation
  - *Why can't we just add 了 to everything?*
  - “I go **running** Mondays.” -> I go run Mondays
  - “**We** went on a **run** yesterday.” -> We go on a run yesterday
  - “**They ran** all day yesterday.” -> They run all day yesterday
- “go” and “went” share the lemma “go”
  - Multi-word conjugations like “will have gone” are lexed first, thus lemmatize to “will have go”



**Lemma 1.1.**

*Let  $M$  be an irreducible  $R$ -module, then  $\text{End}_R(M)$  is a division algebra.*

# Lemmatization

- Usually requires database support
  - e.g., to map “am” “are” “were” to “be” (no procedural technique)
- WordNet
  - Giant database of words, **inflections**, and *concepts* (more on that later)
  - When encountering a stemmed word, use WordNet to find the lemma



# WordNet

Word to search for:

Display Options:

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

Display options for sense: (gloss) "an example sentence"

## Noun

- **S: (n)** [go](#), [spell](#), [tour](#), [turn](#) (a time period for working (after which you will be relieved by someone else)) *"it's my go"; "a spell of work"*
- **S: (n)** [Adam](#), [ecstasy](#), [XTC](#), [go](#), [disco biscuit](#), [cristal](#), [X](#), [hug drug](#) (street names for methylenedioxymethamphetamine)
- **S: (n)** [crack](#), [fling](#), [go](#), [pass](#), [whirl](#), [offer](#) (a usually brief attempt) *"he took a crack at it"; "I gave it a whirl"*
- **S: (n)** [go](#), [go game](#) (a board game for two players who place counters on a grid; the object is to surround and so capture the opponent's counters)

## Verb

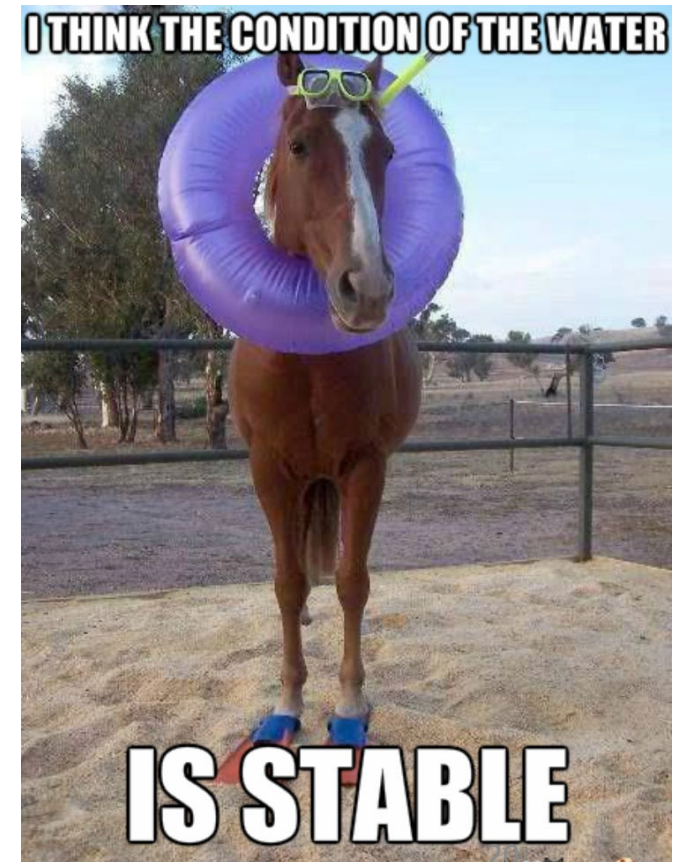
- **S: (v)** [travel](#), [go](#), [move](#), [locomote](#) (change location; move, travel, or proceed, also metaphorically) *"How fast does your new car go?"; "We travelled from Rome to Naples by bus"; "The policemen went from door to door looking for the suspect"; "The soldiers moved towards the city in an attempt to take it before night fell"; "news travelled fast"*
- **S: (v)** [go](#), [proceed](#), [move](#) (follow a procedure or take a course) *"We should go farther in this matter"; "She went through a lot of trouble"; "go about the world in a certain manner"; "Messages must go through diplomatic channels"*
- **S: (v)** [go](#), [go away](#), [depart](#) (move away from a place into another direction) *"Go away before I start to cry"; "The train departs at noon"*
- **S: (v)** [become](#), [go](#), [get](#) (enter or assume a certain state or condition) *"He became annoyed when he heard the bad news"; "It must be getting more serious"; "her face went red with anger"; "She went into ecstasy"; "Get going!"*

# Summary: Lex, Stem, Lemmatize

- **Lexical analysis** (tokenization) uses regular expressions to split long utterances to individual semantic *tokens*
- **Stemming** uses linguistic properties (e.g., consonant/vowel) to remove suffices from words to get *stems*
- **Stop words** are connective glue that are (sometimes) useful to remove
- **Lemmatization** converts words into individual, simplistic *lemmas*
- **Morphological Normalization** refers to techniques like these that *normalize* text's representation for further analysis

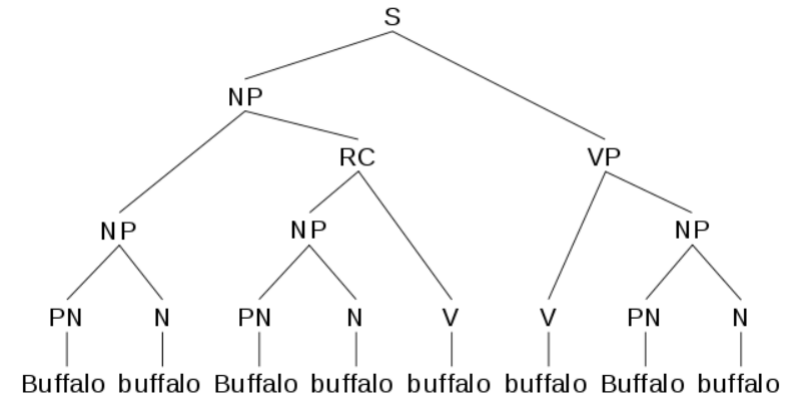
# Issues with morphological normalization

- The simplification comes at a cost: lost semantics
  - “they put the horses in the stable” -> put hors stabl
  - “the marriage has stability” -> marry stabl
- (depends on which stemmer)
- Does *stability* (of a marriage) really relate to *stables* (as in horse storage)?



# Other linguistic blunders

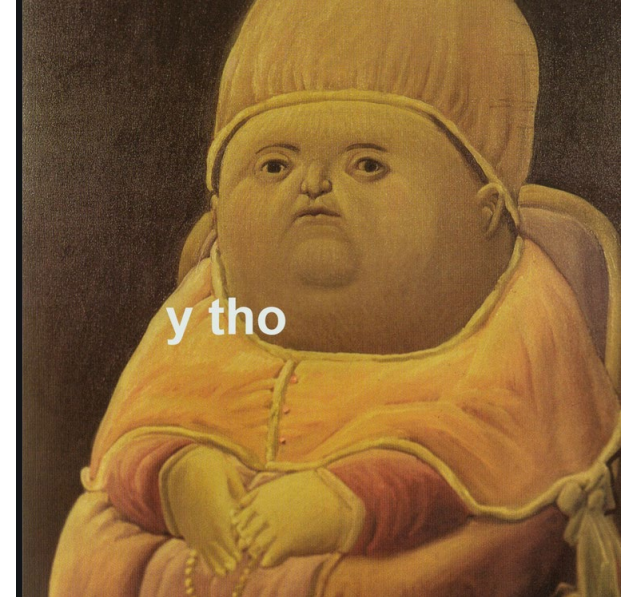
**"Buffalo buffalo Buffalo buffalo buffalo buffalo Buffalo buffalo"** is a grammatically correct sentence in American English, often presented



- “Trump trumps the Trump trumpeters’ trumpets” after normalization:
  - “trump trump trump trump trump” #covfefe

# But why normalize?

- Morphological normalization **simplifies** subsequent analyses.
- Consider: “go” and “travel” are related words. If we don’t normalize, then we have to track the fact that:
  - Go, gone, went, etc. AND travel, traveling, travels, traveled, etc. are related
  - It’s easier to track the relation between normalized versions
    - *Especially* in large corpora of text
- Alternatively, how might we construct something like WordNet?
  - If words A and B are used in context X, we might consider A and B related
  - “We went to Detroit” and “We traveled to Detroit”



# Language modeling

- A **language model** is a collection of statistics learned over particular language
  - Often used to predict how likely a sequence of words is to occur in a given context
- Useful in speech recognition
  - “Recognize speech” vs. “Wreck a nice beach”: which is more likely?
  - “Call an ambulance” vs. “Get an amber lamps”



# Language modeling

- Goal: Maximize  $P(w_i | w_{i-1}, w_{i-2}, \dots, w_k)$ 
  - Find the word  $i$  that is **most likely** to occur next, given **observations** of previous words  $i-1, i-2, \dots$ , etc.
- More generally, the **probability** of some word given some **context**
- The probability function is almost always **empirically derived** from text corpora
- We can build statistics over a corpus of text!



# Language modeling with n-grams

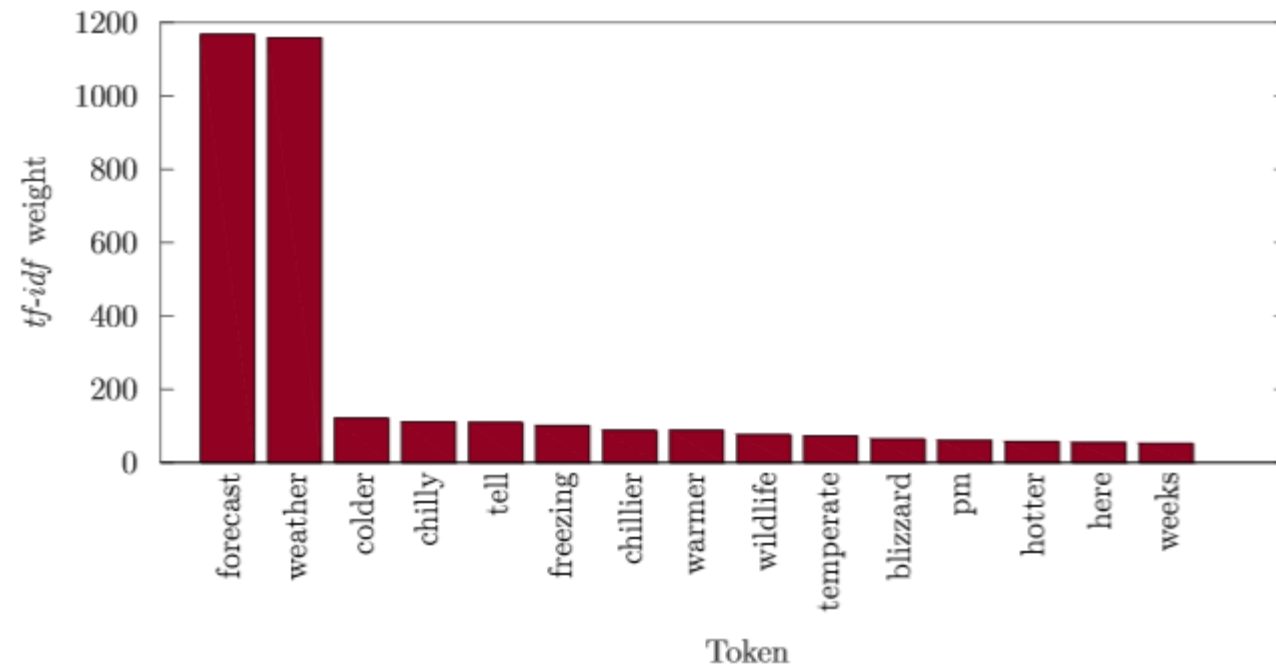
- An *n-gram* is a sequence of  $n$  words in an utterance
  - “I like steak” => [ ( I like ) ; ( like steak ) ] // 2-grams
  - “I like steak” => [ ( I ) ; ( like ) ; ( steak ) ] // 1-grams
  - “I like steak” => [ ( I like steak ) ] // 3-grams
- n-grams help maintain context!
- By counting frequencies of n-grams over large text corpora, we can build simplistic models.
  - If the two gram (I like) frequently occurs, then when we are given the token “I”, we can use our knowledge of the corpus to predict “like” to occur next

# Language modeling: pitfalls

- What words do you think are most common in English?
- **Term Frequency:** How many times does word X occur?
  - “the” lol
  - Even if stop words are removed, some are more common in some contexts
- **Inverse Document Frequency:** How many “documents” (utterances) does word X occur in?
  - Basically, a word is more important if it occurs in few places
    - But, maybe it’s just rare
- **tf-idf:** a combined metric. We want to identify words that are relatively rare (low document frequency, high idf), but frequently used in those documents (high tf)
  - Intuition: related “documents” can be “grouped” by highly-frequent words that are not present in other groups of “documents”

# Intent Classification with Language Models

- We can crudely classify intents using statistics over the example utterances
  - e.g., the 1-grams “forecast” and “weather” are associated with a “get\_weather” intent

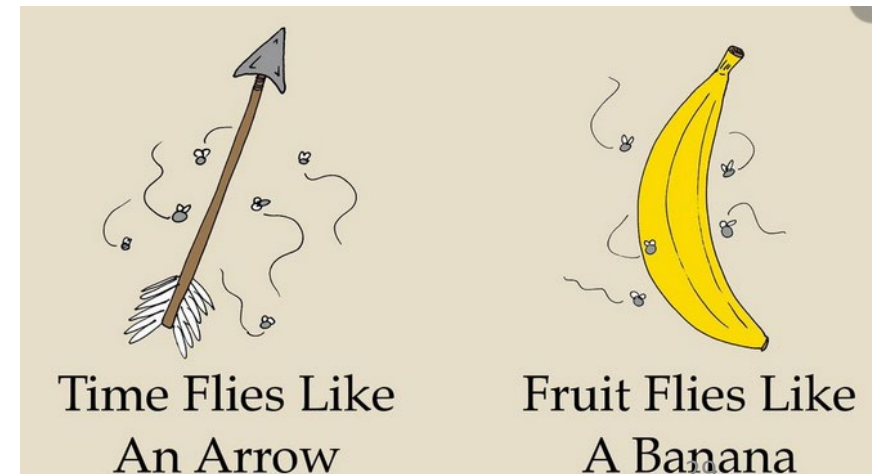


# Intent Classification

- Language models can be a crude way to classify intents
- Utterance data provides examples that can provide statistics
  - Intuition: more frequent word => important for that intent
- What about slots, though?

# Syntactic Analysis: Parsing

- We want to **extract parts-of-speech** to associate with each **word**
- Before/during normalization, parts-of-speech can be labeled
  - **Grammatical rules:** “a Noun-Phrase contains (1) an optional determiner, (2) zero or more adjectives, and (3) a noun, in sequence”
  - “The quick brown fox jumped over the lazy dog.”
    - NP(1) NP(2)
    - (could extract these two noun-phrases?)
- Ambiguities potentially occur
  - Is trump a verb or noun?
  - “Time flies like an arrow; fruit flies like a banana.”



# Part-of-speech tagging: Slot labeling

- Identifying slots can be thought of as a POS labeling task
  - Do we care about actual “part of speech”?
  - “I want to book a flight from **New York** to **San Francisco**.”
  - O O O O O O B-F I-F B-D I-D
- *Inside-Outside-Beginning* (IOB) notation:
  - “O” for tokens that don’t matter (i.e., that aren’t slots)
  - “B” for a token that begins a slot
  - “I” for a token in the middle of a slot (e.g., multi-word slots)
- We’ll revisit this with deep learning

# Alternatives to slot labeling

- During normalization, words can be **mapped**
  - e.g., there are a finite number of cities in the world... we could maintain a giant list of them and substitute “city” whenever they occur
- Not preferred: enforces static language
  - What if someone builds a new city?



# Summary

- Morphological normalization consists of:
  - Lexical analysis / tokenization
  - Lemmatization
  - Syntactic analysis / parsing / part-of-speech tagging
- We can build statistical models of language using textual corpora
  - Language models can help with intent classification
- We can tag important parts of text as slots
  - IOB notation useful for ground-truth labeled data
- btw just import nltk