

Start Recording

- Today:
 - Course Overview and Policies
 - Why is language a challenge?
 - Sets and Counting
 - Assignment 1: due Thursday July 26

Welcome to Ling 473

- Dates of Instruction
- Instructor
- Prerequisites
- Getting Set Up
- Class Tools
- Software
- Programming Languages
- Assignments
- Grading

Calendar

Class Meetings: Tues., Thurs. 4:30 – 6:20 pm
 July 19 – September 6, 2018

Three lecture attendance options:

- In person
- Live broadcast via Zoom
- Recordings posted after each class

Since viewing class recordings does not allow for interactive participation, this attendance option should be used sparingly

Office Hours: Tues., Thurs. 2:30-3:30
 (or by appointment)
 GUG 407 or Zoom

Instructor

Kristen Howell

kphowell@uw.edu

Education:

B.A. Linguistics and Cognitive Science from Scripps/Pomona College

PhD student in Computational Linguistics at UW

Research Interests:

Developing precision grammars based on typological generalizations

Inferring typological information from linguistically annotated data

Ling 473 Prerequisites

CSE 326 or 373: Data Structures

Abstract data types and their implementations as data structures. Efficient use of algorithms employing these data structures; asymptotic analyses. Dictionaries: balanced search trees, hashing. Priority queues: heaps. Disjoint sets with union, find. Graph algorithms: shortest path, minimum spanning tree, topological sort, search. Sorting.

STAT 390 or 391: Probability and Statistics for Computer Science

Fundamentals of probability and statistics from the perspective of the computer scientist. Random variables, distributions and densities, conditional probability, independence. Maximum likelihood, density estimation, Markov chains, classification. Applications in computer science.

(or equivalents)



Ling 473 is required for the NLT Certificate Program



CLMS Program: determination by placement test

Getting Set Up

- UW NetID
- Husky Card
- Computational Linguistics Cluster Account
- Keypad access to “Treehouse” lab in Guggenheim
 - (linghelp@uw.edu)
- Recommended textbooks

Daniel Jurafsky and James H. Martin. (2008) *Speech and Language Processing (2nd edition)*. New Jersey: Prentice-Hall.

Christopher D. Manning and Hinrich Schütze. (1999) *Foundations of Statistical Natural Language Processing*. Cambridge: MIT Press.



Both of these textbooks are required for CLMS and NLT students when taking Ling 570/571. If doing so in the fall, purchase now.

Class Tools

- Course Website on Canvas
 - <https://canvas.uw.edu/courses/1205954>
- Treehouse wiki
 - <http://depts.washington.edu/uwcl/twiki/bin/view.cgi/Main/WebHome>
- CLMS Survival Guide:
 - <http://depts.washington.edu/uwcl/twiki/bin/view.cgi/Main/CLMASurvivalGuide>
- Computational Linguistics Linux cluster
 - Interactive head nodes: patas.ling.washington.edu; dryas.ling.washington.edu
 - About 50 processors on ~20 machines for batch jobs
 - Files for certain assignments may be found in /opt/dropbox
 - Licensed corpora database: <https://vervet.ling.washington.edu/db/index.php>
 - Condor batch submission system; Network File System

Software

- \LaTeX
 - Consider if you're continuing in MS or Ph.D
 - Can create PDF files
 - On Windows: Install MiKTeX, Texmaker, TexStudio
 - Use Overleaf in browser
- Ability to create PDF files
 - Adobe Acrobat (academic pricing at UW Bookstore)
 - Open Office?
 - PDF creator
 - <http://depts.washington.edu/uwcl/twiki/bin/view.cgi/Main/WordToPDF>
- SSH Terminal Programs available from UWICK
 - <http://www.washington.edu/uware/uwick/>
 - WinSCP
 - SSH Tectia
 - Tera Term
 - Putty

Programming Languages

- You may use the following programming languages:

C# (mono)

Java

Python

Assignment solutions will be provided in C#.

If you wish to use another language and have a good reason for doing so, talk to me before starting the first assignment

- This is not a basic programming class. We will not cover how to create, edit, compile, and run programs.

Written Assignments

- Probability and statistics problems
- Required submission format: PDF
- Please do not write homework by hand
 - Now is the time to learn LaTeX or an equation layout tool (Word has an equation tool!)

Programming Projects

- Code must run on UW Comp. Ling. Cluster
 - or you will receive no credit. This is because some projects may reference licensed corpora which you may not copy
 - I won't spend time figuring out why your code doesn't run.
 - You can develop on a home machine, but make sure you test thoroughly on the cluster
- Follow submission instructions
- TAR and submit to Canvas:
 - all required source code and files (except public files)
 - point to public files where possible
 - output file captured from stdout
 - prose description (write-up) of your work
- Sample solutions will be posted after the deadline

Project Submission

- All assignments are due at the deadline and the Canvas timestamp is the final arbiter
- In fairness to other students, solutions will be made available right away— as a result, late submissions will not be accepted
- Test, tar and upload early! Then go back and keep working if you have time
- Note: Canvas only allows me to view your most recent submission— do not overwrite an on time submission with a late one

Grading

- 3 Written Assignments: 30%
- 5 Programming Projects: 65%
- Class and GoPost Participation: 5%
- Policies:
<https://canvas.uw.edu/courses/1205954>

Computational Linguistics

- A Quick Survey of the field
- Why is language hard?
- A resource: The Penn Treebank (PTB)

What is Computational Linguistics?

Applying quantitative techniques to the analysis and processing of human (“natural”) languages.

- Naturally, computers are well-suited to this sort of Natural Language Processing (NLP)
- Cross-disciplinary
 - Linguistics
 - Computer science
 - Mathematics
 - Electrical Engineering
- Computational linguists come from all of these backgrounds

Linguistics

- Nearly all research areas in linguistics can benefit from computational techniques:
 - Phonetics
 - Phonology
 - Morphology
 - Syntax
 - Semantics
 - Pragmatics
 - Discourse Analysis
 - Information Structure
 - Typology
 - Language Documentation

Computer Science

- All fundamental CS data structures and principles apply: Landau (“Big-O”) notation
- Large scale text processing; RegEx; strings; character encoding; data conversion
- Databases; high-throughput computing
- Specialized data structures and techniques
 - String hashes
 - Trie
 - Dynamic programming

Mathematics

- Probability
 - Modeling the occurrence(s) of events
- Statistics
 - Application of probabilistic models
- Set Theory
 - Intersection, Union, Exclusion
- Logic
 - Boolean logic
 - First order logic (predicate calculus)
 - Markov (probabilistic) logic, entailment
- Information Theory
 - Entropy
 - Shannon channels
- Advanced Math
 - Kernel functions (for Support Vector Machines...)
 - Matrix decomposition (i.e. Singular Value Decomposition...)

Ambitions and Jargon

- Natural Language Processing (NLP)
 - Processing human language in some capacity with a computer
 - Can be clever or not
- Natural Language Understanding (NLU)
 - Processing human language in a way that the computer “understands” what the language is saying
- Natural Language Technology (NLT)
 - Generic term for NLP/NLU

Natural Language Understanding

- What is “understanding”?
 - How can you measure it?
 - Do we have a theory of what human beings know when they have an understanding of something?
- “Understanding” is in some way linked to the real world
 - When you know something, you can apply it novel situations in the world
 - Computational linguistics is very poorly linked to real world models
 - True real-world models (that extend being “circle” vs “triangle”) don’t really exist right now

NLP / NLU Subfields

- IE: Information Extraction
- IR: Information Retrieval
- MT: Machine Translation
- Automatic Document Summarization
- ASR: Automatic Speech Recognition
- Speech Synthesis
- QA: Question Answering
- NLG: Natural Language Generation
- CALL: Computer-Assisted Language Learning
- Alternative Input Methods

NLP / NLU Subtasks

- Tokenization (word breaking)
- Sentence Breaking
- Morphological Analysis
 - POS tagging
 - Stemming
- NER: Named Entity Recognition
- WSD: Word Sense Disambiguation
- Anaphora and reference resolution
- Parsing and generation
- Dialogue management and discourse analysis
- Clustering
- Classification
- Treebanking and Corpora curatorship
- ...

NLP Approaches

- Analytical (rule-based)
 - Compose a set of rules that govern the application
 - From data or from linguistic knowledge
 - Implement them
 - Evaluate
- Statistical
 - Create a model that alters performance based on training data
 - Train the model
 - Use the model to make predictions about unseen inputs

Rule Based NLP

Intuition: In human brains, language operates by a set of rules.
Let's infer these and implement them in a computer.

Q: Where is this going to have trouble?

1. We don't know how language operates in human brains
2. What kinds of rules should we use? Which linguistic theory?
3. Can these rule work on hardware the way they do on wetware?
4. How will we know if we're on the right track?

Analytical MT: a case study

The first goal of NLP was Machine Translation (MT)

When I look at an article in Russian, I say: “This is really written in English, but it has been coded in some strange symbols.”

Weaver 1947

Analytical MT: a case study

Another 19 Years Later...

There has been no machine translation of general scientific text, and none is in immediate prospect... After 8 years of work, the project] had to resort to post-editing [which] took slightly longer to do and was more expensive than conventional human translation.”

ALPAC Report 1966



Recommendation of the ALPAC Report: suspend government funding for Machine Translation

Analytical MT: a case study

41 Years Later: realistic progress

Progress on combining rule-based and data-driven approaches to MT will depend on a sustained stream of state-of-the-art, MT-oriented linguistics research... Despite frequent cycles of overly high hopes and subsequent disillusionment, [MT] is the type of application that may demand knowledge-heavy, 'deep' approaches to NLP for its ultimate, long-term success.

Oepen et al. 2007

Why is language so hard?

- As a simplification, let's just consider text
(speech processing has a separate set of problems)

I saw a man with a telescope.

- a bunch of symbols
- coming in order (a string) in Unicode or some other encoding
- broken into “words”
- forming a “sentence”
- with some related semantic proposition

Convention for linguistic examples

- Sentences marked ungrammatical are marked with an asterisk
**This example sentence of ungrammatical is.*
- Sentences judged marginal are marked with a question mark
?A student emailed me yesterday who was almost unable to enroll.
- Grammatical sentences can be semantic nonsense (sometimes marked with a pound sign)
#Colorless green ideas sleep furiously.
- Grammatical sentences which are infelicitous
#It's raining outside but I don't believe that it's raining.

Words

Isa wa manwi thate lesco pe.

Observation: it appears that, *in this language*, one role of the space character is to partition symbols (letters) into units called “words.”




Note: we always try to be aware of, and explicitly state, any assumptions that may not be cross-linguistically valid


Words

It is not always the case that identifying “words” is straightforward:

ผมเห็นผู้ชายกับโทรทรรศน์				
ผม	เห็น	ผู้ชาย	กับ	โทรทรรศน์
p ^h ǒm	hě̌n	p ^h ǔː tɕ ^h aːj	kàp	tʰoː rá tʰát
1-sg	see	man	with	telescope
“I saw (a) man (who was) with a telescope.”				

 The International Phonetic Alphabet (IPA), is the standard form of phonetic transcription.

“person-male” : 1 word or 2?

 This type of formatting for linguistic examples is called “Interlinear Glossed Text,” or IGT

Word Classes

- Can we identify a closed set of word classes to generalize about them?
 - These are called Parts of Speech.
 - In computational linguistics: POS tags
 - Closed and open classes

Closed Word Classes

- A closed class
 - Has a limited number of members
 - Is generally not open to new member production
- Closed word classes
 - Conjunction (and, or, but, ...)
 - Determiner (the, a, this, ...)
 - Pronoun (he, we, they, ...)
 - Auxiliary verb (have, do, be, ...)
 - Preposition (in, of, over, ...)
 - ...and others

Open Word Classes

- An open class:
 - Has a large number of members
 - May accept new members over time
 - May allow new members to be heuristically generated
- Open word classes: may allow compounding, inflecting, or productive morphology
 - Noun
 - Verb
 - Adjective
 - Adverb
 - ...and others

Sentences

**the carnival. I saw a man with a*



Observation: it appears that, in this language, one role of the period is to partition words into sentences.

Ok, let's dismiss those issues too. We'll assume we're given the text of one sentence, unambiguously composed of words. Good enough?

Syntax

**Man a with saw telescope l a.*

Observation: the ordering of words appears to be important in this language.

English is generally Subject-Verb-Object (SVO), which is less common among world languages than SOV. In Russian, word order is more restricted in transitive clauses. Some languages, such as Datooga in Tanzania have free word order. Hixkaryana and Tamil are a few of examples of the rare OVS languages.

More data

With a telescope I saw a man.

#With a man I saw a telescope.

#I saw a telescope with a man.

*A man saw a telescope with I.

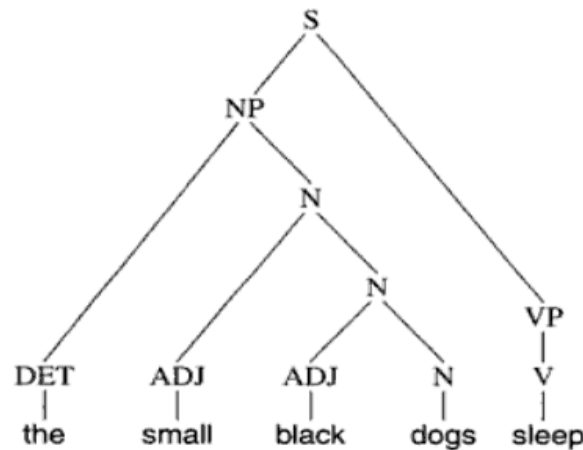
I, with a telescope saw a man.

Observation: some orderings have more felicity than others

Can we find consistent generalizations to capture this? If so, what would we call such a set?

Constituents

- Can sentences be analyzed as containing sub-units which consist of one or more words?



(Hausser 1998)

This is a phrase structure tree.

Hypothesis: The class of grammatical constituents is closed.

Constituent Types

- Constituents are characterized by the part-of-speech of their main word, the “head.”
- Thus, we notice that for many languages, a sentence comprises a:
 - subject (a Noun Phrase or NP)
 - predicate (a Verb Phrase or VP)
- These may be composed of other constituents
 - Prepositional phrase (PP)
 - Determiner phrase (DP)

Constituent Construction

- Noun phrases (NPs)

(DET NN)	<i>The ostrich</i>
(NNP)	<i>Kim</i>
(NN NN)	<i>container ship</i>
(DET JJ NN)	<i>A purple lawnmower</i>
(DET JJ NN)	<i>That darn cat</i>

- Verb phrases (VPs)

(VB)	tango
(VBD NP NP)	gave the dog a bone
(VBD NP PP)	gave a bone to the dog

Syntax

The set of rules governing permissible constructions in a language.

- Syntax constrains the ways in which words may be combined to form constituents and sentences.
- Syntax forms one part of the description, or *grammar*, of a language.



Prescriptive v. descriptive grammar

- Prescriptive
 - Rules against certain usages. Few if any rules for what *is* allowed.
 - Don't end a sentence with a preposition. (where are you from?)
 - Don't use “ain't”, “irregardless”, etc.
- Descriptive
 - Rules characterizing what people *do* say.
 - Goal is to characterize all and only what speakers find acceptable.
 - Based on the scientific method

Prescriptive Grammar

- Prescriptive rules are artificial
 - Often reflect social status associated with speaker dialect
 - African American English (AAE) tends to bear the brunt of prescriptivist rules in the U.S.
 - Stigmatized varieties can be found in most languages
 - Even phenomena common to the standard dialect can be prescribed
 - e.g. singular “they” dates back to Chaucer

Artificiality of prescriptive rules

- Fill in the blanks: *he/his, they/their, or something else?*

Everyone insisted that ___ record was unblemished.

Everyone drives ___ own car to work.

Everyone was happy because ___ passed the test.

Everyone left the room, didn't ___ ?

Everyone left early. ___ seemed happy to get home.

(Taken from Bender, Sag, Wasow)

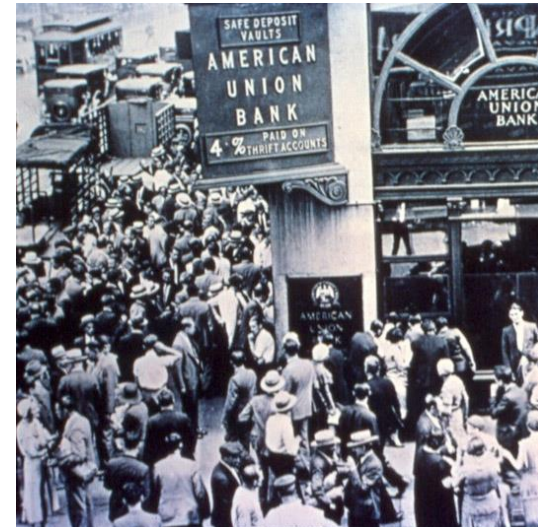
Two kinds of ambiguity

1. Lexical ambiguity

The bank is crumbling.



?



Two kinds of ambiguity

2. Structural ambiguity

I saw a man with a telescope.



?



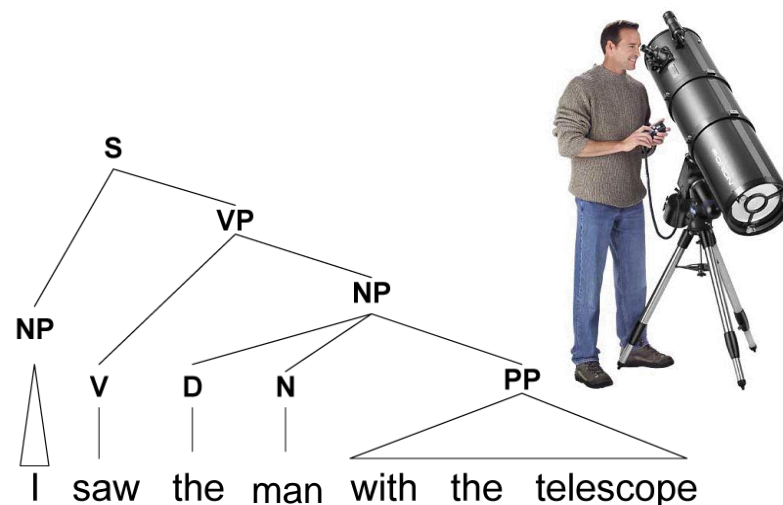
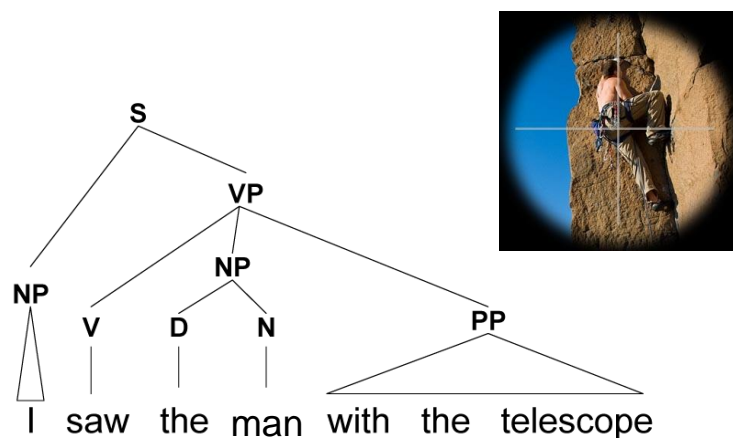
Is that all?



I (often) saw a man with a telescope.

Constituents

- Constituents help us understand and classify syntactic structure
- Here, phrase structure trees show us how constituent structure helps us characterize ambiguity



Trees: Dan Jinguji

Analytical NLP: Summary

- Language is extremely complex, despite our ease using it
- Ambiguity is an inherent feature of language
- Inferring the syntactic rules of a language is difficult
- Individual syntactic rules can be implemented easily, but there are a lot of them

Statistical NLP

- Large gains in practical application in the 1990s
- Computing power was increasing
- NLP accuracy was not
- Idea: Maybe the rules behind language are too hard to fully codify
- Instead let's use algorithms to find likely patterns and correlations in existing data (clever math)
- Most visible example: MT
 - Bing Translate
 - Google Translate

Statistical NLP

- A model will self-learn from language data
- Where is this going to have trouble?
 - What is the basic model? What are the assumptions of the model?
 - How does the model know when to give up? (bad input)
 - What data will you use to train it?
 - What if we don't have enough data?
 - Can it perform on data that is related to but different from the input?

Latest Work: Hybrid analytical/statistical systems

- Hybrid systems (if carefully designed) can mitigate the shortcomings of both approaches
 - Hybrid machine translation (Oepen et al. 2007)
<http://www.mt-archive.info/TMI-2007-Oepen.pdf>
 - Parse ranking in the English Resource Grammar
 - Unsupervised Rule Learning (Poon and Domingos 2009)
<http://www.aclweb.org/anthology-new/D/D09/D09-1001.pdf>

Corpus Linguistics

- A corpus is a collection of text (or data generally)
- Create the rules (by hand) or derive statistical models (algorithmically) according to a corpus of language data
- The rule or the model are created and judged based on their performance against the corpus.

Treebank

- A collection of text with syntactic structure annotation for each sentence
 - Human-annotated
 - Machine generated by precision grammar
- Penn Treebank (PTB) (Marcus et al. 1994)
<http://www.cis.upenn.edu/~treebank/>
- Linguistic Data Consortium (LDC)
<http://www ldc.upenn.edu/>



PTB Example

```
( (S (NP (NNP John) )  
    (VP (VBZ loves)  
        (NP (NNP Mary) ) )  
    (. .) ) )
```

PTB tag set

CC	Coordinating conjunction	PRP	Possessive pronoun
CD	Cardinal number	RB	Adverb
DT	Determiner	RBR	Adverb, comparative
EX	Existential there	RBS	Adverb, superlative
FW	Foreign word	RP	Particle
IN	Preposition or subordinating conjunction	SYM	Symbol
JJ	Adjective	TO	to
JJR	Adjective, comparative	UH	Interjection
JJS	Adjective, superlative	VB	Verb, base form
LS	List item marker	VBD	Verb, past tense
MD	Modal	VBG	Verb, gerund or present participle
NN	Noun, singular or mass	VCN	Verb, past participle
NNS	Noun, plural	VBP	Verb, non 3rd person singular present
NNP	Proper noun, singular	VBZ	Verb, 3rd person singular present
NNPS	Proper noun, plural	WDT	Wh-determiner
PDT	Predeterminer	WP	Wh-pronoun
POS	Possessive ending	WP	Possessive wh-pronoun
PRP	Personal pronoun	WRB	Wh-adverb

Counting

- Corpus linguistics is essentially about counting and arranging collections and sequences of elements (words, characters).
- Elements are collected together into sets or bags
 - This can be done in a number of ways as we'll see
- It's important to keep track of what we are counting. We can count:
 - The number of elements in a collection
 - This is fairly straightforward. This is called the “cardinality” of the collection
 - The number of occurrences of a particular element in a collection
 - We can distinguish this by calling it a “tally.”
 - The number of collections of a certain type that are found in a corpus
 - The number of collections of a certain type that could possibly be formed from another collection of elements

Collections

- Consider a collection of n elements. It can be ordered or unordered, and distinct (all elements are unique), or with repeated elements.

	Ordered	Unordered
Distinct	(i.e. MRU cache)	vocabulary, alphabet, set
Not-Distinct	sentence, string, vector, sequence, word	bag, multiset



We distinguish ordering from sorting. Ordering refers to whether the order matters in the modeling context, not whether the elements happen to be in some sorted order for practical (programming) purposes

Distinctness

- Distinct, i.e. no repetition (“set,” “vocabulary”)
 `{ apple, pear, banana, orange, pomelo }`
 we can count:
 - The number of distinct elements (“cardinality,” “vocabulary size”)
- With repetition (“bag” or “multiset”)
 `{ apple, apple, banana, apple }`
 we can count:
 - The number of distinct elements
 - The number of times each element appears (“tally” or “multiplicity”)
- In mathematics, the word ‘set’ generally means unordered and distinct unless otherwise specified
- Parentheses are usually used when order matters (n-tuples), braces are used when order doesn’t matter { sets }.

Ordering

Ordered, (“string,” “vector,” “sequence,” or “n-tuple”)

(a, man, gave, sandy, a, pomelo)
≠
(a, a, gave, kim, pomelo, sandy)

Unordered, Not-distinct (“bag”)

{ a, man, gave, sandy, a, pomelo }
=
{ sandy, pomelo, man, gave, a, a }

Unordered, Distinct (“vocabulary”)

{ a, man, gave, sandy, pomelo }

- Distinct collections where the order matters are less common. An example is the set of (distinct) words used in a document, by order of appearance. This is a form of Most Recently Used (MRU) cache.

Vocabulary

The distinct set of elements which appear in some other set

- Examples:
 - The words used in a document or sentence
 - All the words of a language
 - The characters used in a word
 - The English alphabet is the ‘vocabulary’ of symbols used in writing the language

Tallying

Tallies:

The count (number of occurrences) for a each distinct element of a non-distinct set

- Each element becomes associated with its count
- Tallying creates a distinct set of tuples (a language model) from a non-distinct set (a corpus)
- Tallying is probably the most commonly used programming pattern in corpus linguistics

Tally Example

"a man gave sandy a pomelo"

(a, 2)

(man, 1)

(gave, 1)

(sandy, 1)

(pomelo, 1)

A tally is a count. We can divide each tally by the vocabulary size to obtain *frequencies*.

(a, .33)

(man, .16)

(gave, .16)

(sandy, .16)

(pomelo, .16)

Combining Counts

- Fundamental counting principle:

The cross product of the sets

$\{m_0 \dots m_{c_m}\}$ and $\{n_0 \dots n_{c_n}\}$

has $c_m \times c_n$ members

- That is, there are $m \times n$ ways to combine exactly one element each from sets of size m and n .
- Combine independent counts by multiplying

Combining Counts

- It follows that if we choose m times from the same set of n elements without depletion (or with replacement), there are n^m possible results. (This includes all possible orderings)

Example:

How many four-letter words can be formed with the English alphabet?

$$26^4 = 456,976$$

Counting Sets: Combinatorics

- The basics of counting and arranging sequences:

Permutations:

all possible orderings of n elements

Combinations:

all (unordered) sets of k elements that can be selected from n elements

Variations:

all orderings of k elements that can be selected from n elements.

Permutations

{ o, r, a, n, g, e }

6 choices	a					{ o, r, n, g, e }
5 choices	a	e				{ o, r, n, g }
4 choices	a	e	g			{ o, r, n }
3 choices	a	e	g	n		{ o, r }
2 choices	a	e	g	n	o	{ r }
1 choice	a	e	g	n	o	{ }

$$6 \times 5 \times 4 \times 3 \times 2 \times 1 =$$
$$6! = 720$$



This is the factorial function,
denoted by $n!$

Permutations

- The previous example, ‘orange’ had no repeated letters in the input.
- Permutation only rearranges elements, so it can never “generate” repetition (like we will see with combinations).
- But if the input contains repetitions of some value(s), we may want to eliminate duplicate outputs.
- To do this, we simply divide $n!$ by the number of those combinations that are indistinguishable from each other

Permutations with Repeated Input Values

Example:

mississippi

11 letters

repeated group { i, i, i, i } : cardinality 4

repeated group { s, s, s, s } : cardinality 4

repeated group { p, p } : cardinality 2

$$\frac{11!}{4! \times 4! \times 2!} = 34,650$$

Applying the fundamental counting principle to combine each set of repeated values $\{ r_{i,0}, r_{i,1}, \dots, r_{i,m} \}$ in the input, this denominator is given by:

$$\prod_i m_i!$$

Combinations

- Also called k -combinations
- How many unordered sets of size k can be formed from a set of cardinality n ?
- We've already considered the case of not depleting (i.e., replacing) when ordering of the output matters:

$$n^k \quad (\text{a.k.a. variations, with repetition})$$



And as for unordered sets of k from n with replacement, see Assignment 1 - extra credit

- If depleting, we can derive this answer from the fundamental counting principle

Combinations

Lets say we have seven books and we need to choose just three (maybe to take on vacation). How many different books can we take?

1st choice: 7 books to choose from

2nd choice: 6 remaining books

3rd choice: 5 remaining books.

$$7 \times 6 \times 5 = 210 \text{ sets.}$$

But ordering doesn't matter. Since each of these sets has 3! orderings, we divide by $3! = 6$, thus counting each set just once.

$$210 \div 6 = 35$$

Combinations

- Our numerator was $n(n - 1) \dots (n - k + 1)$
- The denominator was $k(k - 1) \dots (2)(1)$ or $k!$
- We can rewrite the numerator using factorials:

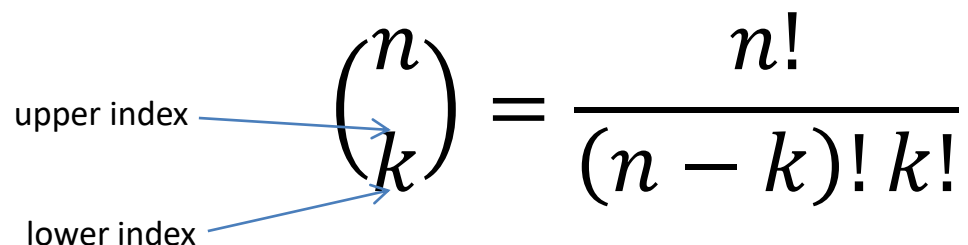
$$\frac{n!}{(n - k)!} = \frac{n(n - 1) \dots (n - k + 1)(n - k) \dots (2)(1)}{(n - k) \dots (2)(1)} \\ = n(n - 1) \dots (n - k + 1)$$

- So the full rewrite looks like:

$$\frac{n!}{(n - k)! k!}$$

- This is called the binomial coefficient or the choose function

Combinations



The diagram shows the binomial coefficient formula $\binom{n}{k} = \frac{n!}{(n-k)!k!}$. Two blue arrows point from text labels to the variables in the formula: one from "upper index" to the n in the numerator of the fraction, and another from "lower index" to the k in the denominator.

$$\binom{n}{k} = \frac{n!}{(n-k)!k!}$$

- Binomial coefficient
- Pronounced “ n choose k ”
- How many subsets of size k can be formed from a set of size n

Variations

- Variations are permutations of combinations: ordered sub-collections of size k chosen from a set n .
- These are just cases where we don't need to cancel out repeated outputs. That is, order *does* matter.
- Choosing elements from the set more than once:

$$n^k$$

- Choosing elements from the set only once:

$$\frac{n!}{(n-k)!}$$

Repetition within a Collection

Ordered or unordered groups?

Distinct elements or repeated elements?

{ set }	unordered, distinct (no repetition)
{ multiset, bag }	unordered, repetition permitted
(tuple)	ordered, repetition permitted

- For combinations (not permutations) we can allow repetition in the *output*

Do we want the combinations of “{ a b c } choose 2” to include { a a }—in which case there are 6 output sets—or not—in which case there are only 3: { a b }, { a, c }, { b, c }

Repetition within a Collection

- If we are creating a set from an input, and the input has repeated values, we have to decide how to count repetition in the output.
- In *permutations*, we count repetitions separately. Permutations of $\{a, a, b\} = 3!$, including two instances of $(a\ a\ b)$, $(a\ b\ a)$, $(b\ a\ a)$.
 - Often we don't want both. Divide by the repetitions.
- In *combinations*, we count repetitions if they are present in the input. “ $\{a, a, b\}$ choose 2” includes two instances of $\{a, b\}$.
 - Do we want to count both? Depends: $\{a, b\}$ is twice as likely as $\{a, a\}$: that might be useful.

Permutations v. Variations

		output type	
		permutations	variations with repetition where lower index = input cardinality
input	distinct { a, b, c }	(a, b, c), (a, c, b), (b, a, c), ... (c, b, a) $n!$	(a, a, a) (a, a, b) (a, a, c) n^n ... (c, c, c)
	has repetition (a, a, b)	(a a b), (a b a) (a a b) , (b a a) (a b a) , (b a a) $\frac{n!}{\prod_i (m_i!)}$	(a, a, a) (a, a, b) (#distinct elements) ⁿ (a, b, a) ... (b, b, b)

Duplicate output tuples

- Input (has repetition)
(a, a, b)

Variations (lower index=3) allowing repetition in the output:

27 of them

(a a a) (a a a) (a a b) (a a a) (a a a) (a a b) (a b a)
(a b a) (a b b) (a a a) (a a a) (a a b) (a a a) (a a a)
(a a b) (a b a) (a b a) (a b b) (b a a) (b a a) (b a b)
(b a a) (b a a) (b a b) (b b a) (b b a) (b b b)

There are only 8 different output tuples

Combinatorics Summary

$\{a b c\}$

- Permutation: how many different orderings?

$(a b c)(a c b)(b a c)(b c a)(c a b)(c b a)$ $n!$

- Combination: how many different subsets (i.e. of 2)?

$\{a b\}\{a c\}\{b c\}$ $\binom{n}{k}$

allowing repetition in the output

$\{a a\}\{a b\}\{a c\}\{b b\}\{b c\}\{c c\}$

- Variations: how many different ordered subsets (i.e. of 2)?

$(a b)(a c)(b a)(b c)(c a)(c b)$ $\frac{n!}{(n-k)!}$

allowing repetition in the output

$(a a)(a b)(a c)(b a)(b b)(b c)(c a)(c b)(c c)$ n^k

Homework

- Assignment 1: Syntax, permutations and combinations
 - On Canvas
 - Upload a PDF
- Project 1: PTB, RegEX
 - On Canvas
 - Will discuss Tuesday

Next time

- Linux, Cluster Computing, Regular Expressions