LT2222 Machine learning for NLP: intro, Winter 2022

Lecture 0: Introduction; formal foundations

Asad Sayeed
with some material from `https://github.com/jonsafari/lt1`

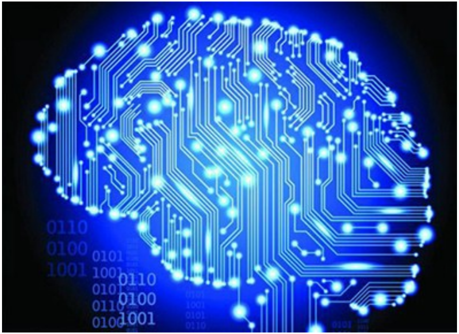
University of Gothenburg

Welcome! Today's agenda:

1. A lot of boring review
2. Format and topics of the course
3. Administrative details

Part 1: Some boring review...

Statistical approaches to Natural Language Processing are ubiquitous.



Something that statistical NLP is not.

It's easy to think of statistical NLP in the consumer market. . . or is it?

The ubiquity of statistical NLP

- Most internet search-related services (especially so if you include information retrieval).

The ubiquity of statistical NLP

- Most internet search-related services (especially so if you include information retrieval).
- Popular question-answering systems on your smartphones, e.g., Siri, Cortana.

The ubiquity of statistical NLP

- Most internet search-related services (especially so if you include information retrieval).
- Popular question-answering systems on your smartphones, e.g., Siri, Cortana.
- Basically: any language technology that must deal with highly variable input involves statistical NLP in practice — and that's practically all of them worth talking about.

But what is statistical NLP?

- Catch-all term for the use of predictive and/or discriminative techniques in accomplishing human-language tasks, such that

But what is statistical NLP?

- Catch-all term for the use of predictive and/or discriminative techniques in accomplishing human-language tasks, such that
  - the predictions and classifications are not explicitly "hard-coded" but based on implicit discovery of relationships inside the data.

But what is statistical NLP?

- Catch-all term for the use of predictive and/or discriminative techniques in accomplishing human-language tasks, such that
  - the predictions and classifications are not explicitly "hard-coded" but based on implicit discovery of relationships inside the data.
  - that discovery takes place based on some kind of machine-learning technique, often very simple.

But what is statistical NLP?

- Catch-all term for the use of predictive and/or discriminative techniques in accomplishing human-language tasks, such that
  - the predictions and classifications are not explicitly “hard-coded” but based on implicit discovery of relationships inside the data.
  - that discovery takes place based on some kind of machine-learning technique, often very simple.
- Statistical relationships are often used in co-operation with “formal” approaches, where the underlying linguistic logic is made “explicit”.

But what is statistical NLP?

- Catch-all term for the use of predictive and/or discriminative techniques in accomplishing human-language tasks, such that
  - the predictions and classifications are not explicitly "hard-coded" but based on implicit discovery of relationships inside the data.
  - that discovery takes place based on some kind of machine-learning technique, often very simple.
- Statistical relationships are often used in co-operation with "formal" approaches, where the underlying linguistic logic is made "explicit".
- Language science (e.g. psycholinguistics, sociolinguistics) often also uses statistical NLP techniques.

But what is statistical NLP?

- Catch-all term for the use of predictive and/or discriminative techniques in accomplishing human-language tasks, such that
  - the predictions and classifications are not explicitly "hard-coded" but based on implicit discovery of relationships inside the data.
  - that discovery takes place based on some kind of machine-learning technique, often very simple.
- Statistical relationships are often used in co-operation with "formal" approaches, where the underlying linguistic logic is made "explicit".
- Language science (e.g. psycholinguistics, sociolinguistics) often also uses statistical NLP techniques.

Let's illustrate what this really means... (you saw this last semester...)

Q: Does language have anything to do with the weather?

A: Yes. But first...

. . . a tongue-twister in English.

How much wood could a woodchuck chuck if a woodchuck could chuck wood?

. . . a tongue-twister in English.

How much wood could a woodchuck chuck if a woodchuck could chuck wood?

One possible answer:

As much wood as a woodchuck could chuck.

Can we calculate how likely that answer is?

Can we calculate how likely that answer is?

That depends ... on what you mean by "likely".

Can we calculate how likely that answer is?

That depends . . . on what you mean by "likely".
To estimate the likelihood of an answer (in the form of a sentence), you need:

- An evidentiary basis.
  $\Rightarrow$ in modern statistical natural language processing, we use large corpora.

Can we calculate how likely that answer is?

That depends . . . on what you mean by "likely".

To estimate the likelihood of an answer (in the form of a sentence), you need:

- An evidentiary basis.
  $\Rightarrow$ in modern statistical natural language processing, we use large corpora.
- A theory that connects the evidence to the likelihood you're trying to estimate.

Can we calculate how likely that answer is?

That depends . . . on what you mean by "likely".
To estimate the likelihood of an answer (in the form of a sentence), you need:

- An evidentiary basis.
  $\Rightarrow$ in modern statistical natural language processing, we use large corpora.
- A theory that connects the evidence to the likelihood you're trying to estimate.
  - Assume sentences are made of words.
  - So the probability of a sentence might have something to do with the probability of the words in the sentence.

Can we calculate how likely that answer is?

That depends . . . on what you mean by "likely".
To estimate the likelihood of an answer (in the form of a sentence), you need:

- An evidentiary basis.
  ⇒ in modern statistical natural language processing, we use large corpora.
- A theory that connects the evidence to the likelihood you're trying to estimate.
  - Assume sentences are made of words.
  - So the probability of a sentence might have something to do with the probability of the words in the sentence.
- A means to combine the pieces of evidence.
  ⇒ if words matter, then we need a theory of sentence structure from words.

Why do we want a likelihood?

Consider natural language processing systems in real life. E.g., machine translation:

- Translate "How much wood could a woodchuck chuck?" to French.
  - The word "could": possibility in French expressible with two different grammatical forms (*"peut"*/*"pourrait"*).
  - Choose better one *in context*.
  - Hard to do over all words deterministically $\leftarrow$ years of effort to create the "rules", but never succeed.
- Countless other applications: such as answering a question. . . .

So how do we get the evidence?

Count words.

how much wood could a woodchuck chuck if a woodchuck could chuck wood ?

Assume that this is our corpus. Total number of words: 14 (incl. the "?").

So how do we get the evidence?

Count words.

how much wood could a woodchuck chuck if a woodchuck could chuck wood ?

Assume that this is our corpus. Total number of words: 14 (incl. the "?").

word type	token count
a	2
chuck	2
could	2
how	1
if	1

word type	token count
much	1
wood	2
woodchuck	2
?	1

So how do we get the evidence?

Count words.

how much wood could a woodchuck chuck if a woodchuck could chuck wood ?

Assume that this is our corpus. Total number of words: 14 (incl. the "?").

word type	token count	p(word)
a	2	0.14
chuck	2	0.14
could	2	0.14
how	1	0.07
if	1	0.07

word type	token count	p(word)
much	1	0.07
wood	2	0.14
woodchuck	2	0.14
?	1	0.07

Then calculate probability per type of word as count/14.

Calculate the probability of expressions.

word type	token count	p(word)
a	2	0.14
chuck	2	0.14
could	2	0.14
how	1	0.07
if	1	0.07

word type	token count	p(word)
much	1	0.07
wood	2	0.14
woodchuck	2	0.14
?	1	0.07

The joint probability of multiple words: how likely they are to occur in the same text.

$$p(w_1, w_2, \ldots) = p(w_1)p(w_2)\ldots$$

Calculate some joint probabilities:

- $p(\text{if,woodchuck}) =$
- $p(\text{wood,woodchuck}) =$
- $p(\text{how,could,a}) =$

Calculate the probability of expressions.

word type	token count	p(word)
a	2	0.14
chuck	2	0.14
could	2	0.14
how	1	0.07
if	1	0.07

word type	token count	p(word)
much	1	0.07
wood	2	0.14
woodchuck	2	0.14
?	1	0.07

The joint probability of multiple words: how likely they are to occur in the same text.

$$p(w_1, w_2, \ldots) = p(w_1)p(w_2)\ldots$$

Calculate some joint probabilities:

- p(if,woodchuck) = 0.07 x 0.14 = 0.01
- p(wood,woodchuck) = 0.14 x 0.14 = 0.02
- p(how,could,a) = 0.07 x 0.14 x 0.14 = 0.001

Calculating the probability of expressions.

word type	token count	p(word)
a	2	0.14
chuck	2	0.14
could	2	0.14
how	1	0.07
if	1	0.07

word type	token count	p(word)
much	1	0.07
wood	2	0.14
woodchuck	2	0.14
?	1	0.07

Now we can calculate the joint probability of our answer.

As much wood as a woodchuck could chuck.

- $p(\text{as},\text{much},\text{wood},\text{as},\text{a},\text{woodchuck},\text{could},\text{chuck}) =$

Calculating the probability of expressions.

word type	token count	p(word)
a	2	0.14
chuck	2	0.14
could	2	0.14
how	1	0.07
if	1	0.07

word type	token count	p(word)
much	1	0.07
wood	2	0.14
woodchuck	2	0.14
?	1	0.07

Now we can calculate the joint probability of our answer.

As much wood as a woodchuck could chuck.

- $p(\text{as,much,wood,as,a,woodchuck,could,chuck}) = 0 \times \ldots$

Calculating the probability of expressions.

word type	token count	p(word)
a	2	0.14
chuck	2	0.14
could	2	0.14
how	1	0.07
if	1	0.07

word type	token count	p(word)
much	1	0.07
wood	2	0.14
woodchuck	2	0.14
?	1	0.07

Now we can calculate the joint probability of our answer.

As much wood as a woodchuck could chuck.

- $p(\text{as,much,wood,as,a,woodchuck,could,chuck}) = 0 \times \ldots$
- Uh oh: there's no "as" in our probability table.
  $\Rightarrow$ we will get to missing items soon.

Calculating the probability of expressions.

word type	token count	p(word)
a	2	0.14
chuck	2	0.14
could	2	0.14
how	1	0.07
if	1	0.07

word type	token count	p(word)
much	1	0.07
wood	2	0.14
woodchuck	2	0.14
?	1	0.07

Now we can calculate the joint probability of our answer.

As much wood as a woodchuck could chuck.

- $p(\text{as,much,wood,as,a,woodchuck,could,chuck}) = 0 \times \ldots$
- Uh oh: there's no "as" in our probability table.
  $\Rightarrow$ we will get to missing items soon.
- So, try $p(\text{much,wood,a,woodchuck,could,chuck}) =$

Calculating the probability of expressions.

word type	token count	p(word)
a	2	0.14
chuck	2	0.14
could	2	0.14
how	1	0.07
if	1	0.07

word type	token count	p(word)
much	1	0.07
wood	2	0.14
woodchuck	2	0.14
?	1	0.07

Now we can calculate the joint probability of our answer.

As much wood as a woodchuck could chuck.

- $p(\text{as,much,wood,as,a,woodchuck,could,chuck}) = 0 \times \ldots$
- Uh oh: there's no "as" in our probability table.
  $\Rightarrow$ we will get to missing items soon.
- So, try $p(\text{much,wood,a,woodchuck,could,chuck}) =$
  $0.07 \times 0.14 \times 0.14 \times 0.14 \times 0.14 \times 0.14 = 3.76\text{e-}05$

Words come in an order.

Calculating the joint probability of unigrams (single words): is it a good model?

Words come in an order.

Calculating the joint probability of unigrams (single words): is it a good model?
Backwards...

chuck could woodchuck a as wood much as

...is not an English sentence.

Words come in an order.

Calculating the joint probability of unigrams (single words): is it a good model?
Backwards. . .

chuck could woodchuck a as wood much as

. . . is not an English sentence.

- Joint unigram probability: the same, no matter what, as "as much wood as a woodchuck could chuck".

Words come in an order.

Calculating the joint probability of unigrams (single words): is it a good model?

Backwards...

chuck could woodchuck a as wood much as

...is not an English sentence.

- Joint unigram probability: the same, no matter what, as "as much wood as a woodchuck could chuck".
- We definitely don't want that to be true. So our theory must include sequences.

And this is what language has to do with the weather.

What was the weather like two years ago in Holland?

Average temperature at Amsterdam Schiphol:

18.11.2014
8 C

And what was it the day before that?

Average temperature at Amsterdam Schiphol:

17.11.2014	18.11.2014
10 C	8 C

And before that?

Average temperature at Amsterdam Schiphol:

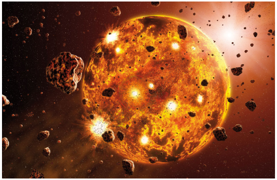
16.11.2014	17.11.2014	18.11.2014
9 C	10 C	8 C

It's as though we know something about the next day from the previous days!

But how many days do we need?

Surely not to the beginning of the Earth!

Average temperature at Amsterdam Schiphol:

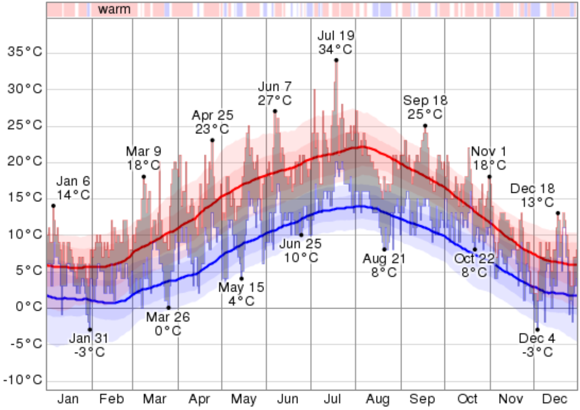


. . .

16.11.2014	17.11.2014	18.11.2014
9 C	10 C	8 C

We have expectations about changes.

We know that yesterday is a good clue about today.
Temperatures in Amsterdam in 2014:



The daily temperature is a Markov process.

Let T_d = temperature T on day d.
We can represent the probability conditionally.

Probability of today's temperature given universe

$p(T_d | T_{d-1}, T_{d-2}, \ldots, T_{d-\infty})$

The daily temperature is a Markov process.

Let T_d = temperature T on day d.
We can represent the probability conditionally.

Probability of today's temperature given 2 previous days

$$p(T_d | T_{d-1}, T_{d-2}, \ldots, T_{d-\infty}) \approx p(T_d | T_{d-1}, T_{d-2})$$

But we only need a few days to give us a trend. So we make a Markov assumption.

The daily temperature is a Markov process.

Let T_d = temperature T on day d.
We can represent the probability conditionally.

Probability of today's temperature given 2 previous days

$p(T_d | T_{d-1}, T_{d-2}, \ldots, T_{d-\infty}) \approx p(T_d | T_{d-1}, T_{d-2})$

But we only need a few days to give us a trend. So we make a Markov assumption.
Then we can calculate the joint probability of a sequence of days:

Markov chain

$p(T_d, T_{d-1}, T_{d-2}) =$
$p(T_d | T_{d-1}, T_{d-2}) p(T_{d-1} | T_{d-2}, T_{d-3}) p(T_{d-2} | T_{d-3}, T_{d-4})$

Getting Markovian with language.

Let's make a Markov assumption over sentences. So how many words previous to "chuck" do we need?

As much wood as a woodchuck could **chuck**.

Getting Markovian with language.

Let's make a Markov assumption over sentences. So how many words previous to "chuck" do we need?

As much wood as a woodchuck could **chuck**.

- "could" is an auxiliary that selects for a verb.

Getting Markovian with language.

Let's make a Markov assumption over sentences. So how many words previous to "chuck" do we need?

As much wood as a woodchuck could **chuck**.

- "could" is an auxiliary that selects for a verb.
- "woodchuck" – maybe. We're asking if woodchucks can chuck, it's in the corpus.

Getting Markovian with language.

Let's make a Markov assumption over sentences. So how many words previous to "chuck" do we need?

As much wood as a woodchuck could **chuck**.

- "could" is an auxiliary that selects for a verb.
- "woodchuck" – maybe. We're asking if woodchucks can chuck, it's in the corpus.
- "much"? No, probably not.

Getting Markovian with language.

Let's make a Markov assumption over sentences. So how many words previous to "chuck" do we need?

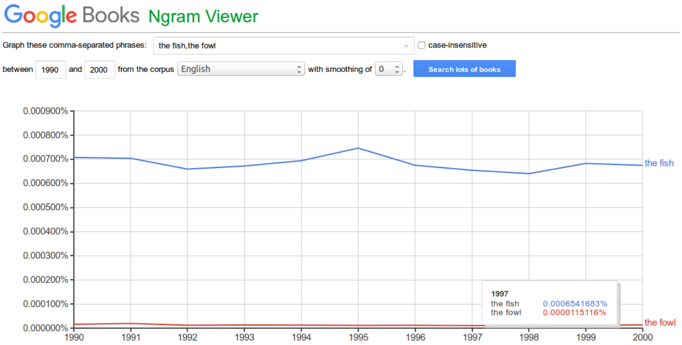
As much wood as a woodchuck could **chuck**.

- "could" is an auxiliary that selects for a verb.
- "woodchuck" – maybe. We're asking if woodchucks can chuck, it's in the corpus.
- "much"? No, probably not.

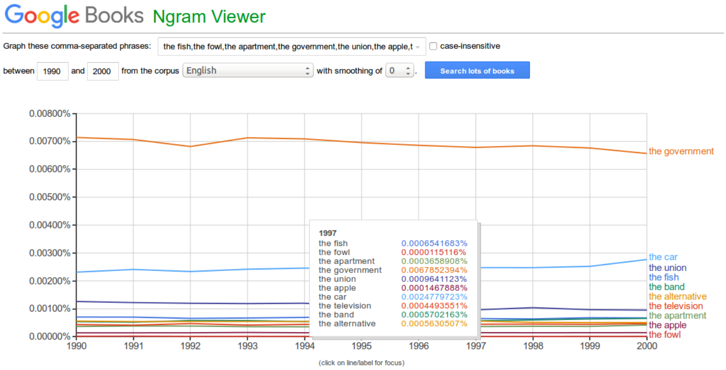
Two words back seems to be a common choice.

We can check a bigger corpus.

Leave aside the woodchucks for a moment. Let's try a couple of 2-word expressions. "The fish" vs "the fowl.".
The Google Books Ngram viewer:



But lots of things follow “the”.



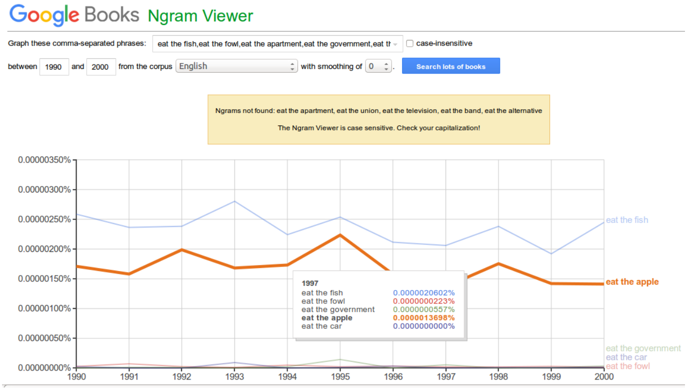
It's not hugely informative...

...because the whole category of nouns can follow "the".

It's not hugely informative. . .

. . . because the whole category of nouns can follow "the".

So what if we add another word, "eat":



The additional word is hugely informative!

So this is a way language is **not** like the weather.

The additional word is hugely informative!

So this is a way language is **not** like the weather.

- Sure, tomorrow will resemble today, in terms of temperature.
  - But knowing what happened yesterday doesn't drastically change the estimate.

The additional word is hugely informative!

So this is a way language is **not** like the weather.

- Sure, tomorrow will resemble today, in terms of temperature.
    - But knowing what happened yesterday doesn't drastically change the estimate.
- But make your **bi**gram into a **tri**gram:
    - The distribution radically changes.
    - "eat" is very informative.

We can even look for 4-grams.

Thus we just call these *n*-grams, for any *n*.
So when we look for 4-grams starting with "quickly eat the fish/apple/car"?

We can even look for 4-grams.

Thus we just call these *n*-grams, for any *n*.
So when we look for 4-grams starting with "quickly eat the fish/apple/car"?

Google Ngrams doesn't find anything! (for 1990-2000).

We can even look for 4-grams.

Thus we just call these *n*-grams, for any *n*.
So when we look for 4-grams starting with "quickly eat the fish/apple/car"?

Google Ngrams doesn't find anything! (for 1990-2000).

Not even "quickly eat the apple"!

We can even look for 4-grams.

Thus we just call these *n*-grams, for any *n*.
So when we look for 4-grams starting with "quickly eat the fish/apple/car"?

Google Ngrams doesn't find anything! (for 1990-2000).

Not even "quickly eat the apple"!

It's not always the case that trigrams work, but they're often practical because of sparsity.

And that involved a lot of things we're going to talk about during the rest of the course.

Part 2: the course

Who are we?

You already know me, but:

Who are we?

You already know me, but:

- Asad Sayeed ("course organizer")
  - Senior Lecturer with FLoV (CLASP, GRIPES projects)
  - Ph.D., Computer Science, University of Maryland (2011).
  - Research areas: machine learning for NLP, computational psycholinguistics, and more

Who are we?

You already know me, but:

- Asad Sayeed ("course organizer")
  - Senior Lecturer with FLoV (CLASP, GRIPES projects)
  - Ph.D., Computer Science, University of Maryland (2011).
  - Research areas: machine learning for NLP, computational psycholinguistics, and more
- Warrick Macmillan
  - Teaching assistant.
  - Student from previous edition of course (LT2212 V19).

Format of the course

The “rhythm” of the course will look approximately like this, with occasional exceptions:

- Mondays: Q&A (on any previous video lecture)
- Wednesdays: help sessions for assignments (starting when assignments first given.
- Thursdays (starting Feb 3): sometimes in-person live demo

All content will be posted as video lectures, slides, etc on the Canvas page.

Evaluation of the course

- Deliverables (by you :)):
  - 3 assignments, including programming and problem-solving, each expected to take 2-3 weeks (can break this up based on student preference)
  - There is **no** final written exam.
- Tentative course grade: 33% per assignment plus 1 free percent.
- Tentative standard:
  - Pass (G): Complete 3 assignments, obtain min 50% on each assignment.
  - Pass-with-distinction (VG): Complete 3 assignments, get min 90% average overall.

Assignments

- Individual submission, can help each other in small groups.
- coding projects, mostly Python, possibly some mathematics
- submission via Canvas.
- We will aim for 15 business days of turnaround from submission, if submitted on time. Otherwise, latest by the summer.
- First assignment will be out very soon.

Please review the university's academic integrity policies.

Emphasis of the course

- Practical skills in statistical NLP.
  - Emphasis on the data pipeline.
  - Goal: getting from data to analysis/model/application.
  - Gain familiarity with some statistical NLP tools.
- Foundational theoretical skills.
  - Getting an intuitive grasp of the mathematical underpinnings of statistical NLP techniques.
  - Just enough for application.
  - (Preliminary knowledge required for machine learning course in Fall!)

Tentative topic list

Based on how we progress (I sometimes change direction based on student request/preferences/progress).

- Math review.
- More programming topics and computing skills for statistical NLP.
- Working with text data.
- Introduction to modern machine learning techniques (incl. deep learning)
- Applications in various areas of NLP (e.g., document classification, machine translation) based on an *ad hoc* basis.

Technical details

I'm assuming you have a background in basic Python programming from previous courses.

- Programming: mostly Python 3.x, using nltk, scikit-learn, pandas, and other relevant Python packages.
- We will introduce neural networks via PyTorch.
- Recommended to have your own laptop with a Linux installation, but we will also use mltgpu or eduserv.
- We will introduce git and make use of command-line techniques.

This course is contiguous with the Machine Learning for NLP advanced course in the fall.

Readings/textbook

- No regular readings in the beginning, this is principally a practical course.
- Students are encouraged to do their own research on the Internet to find clarificatory material on topics (there is a wealth of it, and it is an important skill in this business).
- Main textbook: "Natural Language Processing with PyTorch", readings will start in the middle of the course. (also used in first half of Machine Learning course) – a bit outdated now.

Student discussion of course desiderata

We will return with video lecture.

Part 3: More boring review

Formal Languages

- Once Upon a Time...

Formal Languages

- Once Upon a Time...
- Mathematicians started to think about language...

Formal Languages

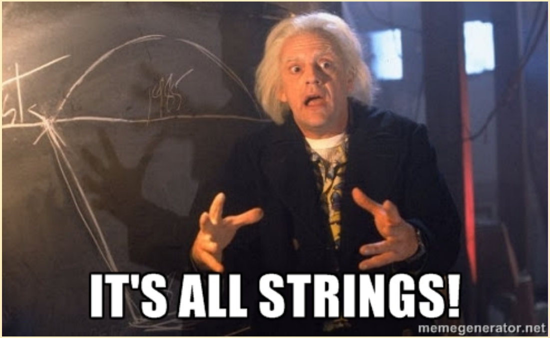
- Once Upon a Time...
- Mathematicians started to think about language...
- They used ideas from logic to represent linguistic objects...

Formal Languages

- Once Upon a Time...
- Mathematicians started to think about language...
- They used ideas from logic to represent linguistic objects...
- They had a really, mmm, audacious idea...

Formal Languages

- Once Upon a Time...
- Mathematicians started to think about language...
- They used ideas from logic to represent linguistic objects...
- They had a really, mmm, audacious idea...



Strings

What's a String?

- A **string** in this context is just a sequence of words

Strings

What's a String?

- A **string** in this context is just a sequence of words
- A **formal language** (L) is a subset of all the possible strings

Strings

What's a String?

- A **string** in this context is just a sequence of words
- A **formal language** (L) is a subset of all the possible strings
- An **vocabulary** (Σ, also sometimes called *alphabet*) here is a set of all the words in the language

Strings

What's a String?

- A **string** in this context is just a sequence of words
- A **formal language** (L) is a subset of all the possible strings
- An **vocabulary** (Σ, also sometimes called *alphabet*) here is a set of all the words in the language
- Words here don't need to correspond to words used for natural languages

Strings

What's a String?

- A **string** in this context is just a sequence of words
- A **formal language** (L) is a subset of all the possible strings
- An **vocabulary** (Σ, also sometimes called *alphabet*) here is a set of all the words in the language
- Words here don't need to correspond to words used for natural languages
- For example, this set:

  { ☺, 🌳, ☃ }

  is a perfectly valid vocabulary for a formal language.

Strings

What's a String?

- A **string** in this context is just a sequence of words
- A **formal language** (L) is a subset of all the possible strings
- An **vocabulary** (Σ, also sometimes called *alphabet*) here is a set of all the words in the language
- Words here don't need to correspond to words used for natural languages
- For example, this set:

  { 😊, 🌳, ☃ }

  is a perfectly valid vocabulary for a formal language. But we usually use boring symbols like {a, b, c}

Strings

What's a String?

- A **string** in this context is just a sequence of words
- A **formal language** (L) is a subset of all the possible strings
- An **vocabulary** (Σ, also sometimes called *alphabet*) here is a set of all the words in the language
- Words here don't need to correspond to words used for natural languages
- For example, this set:

  is a perfectly valid vocabulary for a formal language. But we usually use boring symbols like {a, b, c}
- (Similar to musical/poetic form analysis)

Formal Grammar

- A formal grammar is a way of telling what a valid string is in a formal language
- Formal grammars can also **generate** valid strings

Formal Grammar

- A formal grammar is a way of telling what a valid string is in a formal language
- Formal grammars can also **generate** valid strings
- If two different grammars can generate/accept the same formal languages, then they have the same **weak generative capacity**

Formal Grammar

- A formal grammar is a way of telling what a valid string is in a formal language
- Formal grammars can also **generate** valid strings
- If two different grammars can generate/accept the same formal languages, then they have the same **weak generative capacity**
- If two different grammars can generate/accept the same structures as well, then they have the same **strong generative capacity**

Formal Language Hierarchy

	Formal Language
	Non-Turing-acceptable
0:	**Recursively enumerable**
	Recursive/ Decidable
1:	Context-sensitive
	Indexed
	Mildly context-sensitive
2:	**Context-free**
	Deterministic context-free
3:	**Regular**
	Finite

Formal Language Hierarchy

	Formal Language
	Non-Turing-acceptable
0:	**Recursively enumerable**
	Recursive/ Decidable
1:	Context-sensitive
	Indexed
	Mildly context-sensitive
2:	**Context-free**
	Deterministic context-free
3:	**Regular**
	Finite

This is extended from the older *Chomsky hierarchy*.

Formal Language Hierarchy

	Formal Language
	Non-Turing-acceptable
0:	**Recursively enumerable**
	Recursive/ Decidable
1:	Context-sensitive
	Indexed
	Mildly context-sensitive
2:	**Context-free**
	Deterministic context-free
3:	**Regular**
	Finite

This is extended from the older *Chomsky hierarchy*. We'll discuss the ones in boldface, as they're relevant to natural languages.

Why is this Stuff Relevant??

- Knowing what types of formal languages a grammar/automaton can generate & accept will give you an idea of what phenomena in natural languages that they can handle

Why is this Stuff Relevant??

- Knowing what types of formal languages a grammar/automaton can generate & accept will give you an idea of what phenomena in natural languages that they can handle
- For example: long-distance dependencies, complex reordering in machine translation, reduplication, etc.

Why is this Stuff Relevant??

- Knowing what types of formal languages a grammar/automaton can generate & accept will give you an idea of what phenomena in natural languages that they can handle
- For example: long-distance dependencies, complex reordering in machine translation, reduplication, etc.
- You can also get an idea of how fast or slow it will take for a computer (or human) to process sequential stuff (like natural language!)

Finite Languages

- In a finite language, there are a finite (ie not infinite) number of valid *sentences*.
- Time: constant (through hash-table lookup)
- Memory: constant (duh)

Finite Languages

- In a finite language, there are a finite (ie not infinite) number of valid *sentences*.
- Time: constant (through hash-table lookup)
- Memory: constant (duh)
- For natural language, this would correspond to having a finite number of possible sentences

Are Natural Languages Finite??!!

- It sounds rather odd to think that you could ever list all of the possible sentences of a natural language

Are Natural Languages Finite??!!

- It sounds rather odd to think that you could ever list all of the possible sentences of a natural language
- But...

Are Natural Languages Finite??!!

- It sounds rather odd to think that you could ever list all of the possible sentences of a natural language
- But...
- There's a big difference between a really large number and infinity

Are Natural Languages Finite??!!

- It sounds rather odd to think that you could ever list all of the possible sentences of a natural language
- But...
- There's a big difference between a really large number and infinity
- If a natural language has a vocabulary of, say, 100 million words ...

Are Natural Languages Finite??!!

- It sounds rather odd to think that you could ever list all of the possible sentences of a natural language
- But...
- There's a big difference between a really large number and infinity
- If a natural language has a vocabulary of, say, 100 million words ...
- And a sentence can have, say, up to 10,000 words in it, ...

Are Natural Languages Finite??!!

- It sounds rather odd to think that you could ever list all of the possible sentences of a natural language
- But...
- There's a big difference between a really large number and infinity
- If a natural language has a vocabulary of, say, 100 million words ...
- And a sentence can have, say, up to 10,000 words in it, ...
- Then there would be $10^{80,000}$ possible sentences

Are Natural Languages Finite??!!

- It sounds rather odd to think that you could ever list all of the possible sentences of a natural language
- But...
- There's a big difference between a really large number and infinity
- If a natural language has a vocabulary of, say, 100 million words ...
- And a sentence can have, say, up to 10,000 words in it, ...
- Then there would be $10^{80,000}$ possible sentences
- This number sounds way too big to be practical for either humans or computers to deal with!

Are Natural Languages Finite??!!

- It sounds rather odd to think that you could ever list all of the possible sentences of a natural language
- But...
- There's a big difference between a really large number and infinity
- If a natural language has a vocabulary of, say, 100 million words ...
- And a sentence can have, say, up to 10,000 words in it, ...
- Then there would be $10^{80,000}$ possible sentences
- This number sounds way too big to be practical for either humans or computers to deal with!
- But it's much smaller than infinity.

Are Natural Languages Finite??!!

- It sounds rather odd to think that you could ever list all of the possible sentences of a natural language
- But...
- There's a big difference between a really large number and infinity
- If a natural language has a vocabulary of, say, 100 million words ...
- And a sentence can have, say, up to 10,000 words in it, ...
- Then there would be $10^{80,000}$ possible sentences
- This number sounds way too big to be practical for either humans or computers to deal with!
- But it's much smaller than infinity.
- Much much smaller.

Are Natural Languages Finite??!!

- It sounds rather odd to think that you could ever list all of the possible sentences of a natural language
- But...
- There's a big difference between a really large number and infinity
- If a natural language has a vocabulary of, say, 100 million words ...
- And a sentence can have, say, up to 10,000 words in it, ...
- Then there would be $10^{80,000}$ possible sentences
- This number sounds way too big to be practical for either humans or computers to deal with!
- But it's much smaller than infinity.
- Much much smaller.
- (There's more discussion on the interwebs if you're interested)

(A digression on complexity)

- Processing different kinds of languages take different kinds of machines.

(A digression on complexity)

- Processing different kinds of languages take different kinds of machines.
- The automaton that recognizes a language represents an algorithm.

(A digression on complexity)

- Processing different kinds of languages take different kinds of machines.
- The automaton that recognizes a language represents an algorithm.
- Algorithms take up "space" units (memory to process) and "time" units (number of steps to do something).

(A digression on complexity)

- Processing different kinds of languages take different kinds of machines.
- The automaton that recognizes a language represents an algorithm.
- Algorithms take up "space" units (memory to process) and "time" units (number of steps to do something).
- We can characterize what this means in terms of the length of the input string, which we'll call n.

(A digression on complexity)

- Processing different kinds of languages take different kinds of machines.
- The automaton that recognizes a language represents an algorithm.
- Algorithms take up "space" units (memory to process) and "time" units (number of steps to do something).
- We can characterize what this means in terms of the length of the input string, which we'll call n.
- Then we have something called big-$\mathcal{O}$ notation from computer science. To make a long story short:

$\mathcal{O}(1)$	"constant time"	# units unrelated to input
$\mathcal{O}(n)$	"linear time"	# units lin. proportional to input string
$\mathcal{O}(n^2)$	"quadratic time"	# unites quadrat. prop. to input string
...		

Regular Languages

- Ok, so maybe for now it's too difficult to list all possible sentences
- Let's assume that the vocabulary (Σ) is still fixed (or finite), but we can generate an infinite number of sentences from this fixed vocab
- Regular grammars have a fixed-length history, so they're limited in the types of long-distance phenomena they can handle

Regular Languages

- Ok, so maybe for now it's too difficult to list all possible sentences
- Let's assume that the vocabulary (Σ) is still fixed (or finite), but we can generate an infinite number of sentences from this fixed vocab
- Regular grammars have a fixed-length history, so they're limited in the types of long-distance phenomena they can handle
- For example: a a′ b b′ c c′

Regular Languages

- Ok, so maybe for now it's too difficult to list all possible sentences
- Let's assume that the vocabulary (Σ) is still fixed (or finite), but we can generate an infinite number of sentences from this fixed vocab
- Regular grammars have a fixed-length history, so they're limited in the types of long-distance phenomena they can handle
- For example: a a$'$ b b$'$ c c$'$
- Processing regular languages can be done in linear time ($\mathcal{O}(n)$), with a constant size of memory ($\mathcal{O}(1)$)

Deterministic Context-Free Languages

- Deterministic context-free (DCF) languages include longer-distance phenomena
- DCF grammars have a full-length history, as long as there's no ambiguity (ie. it can't backtrack)

Deterministic Context-Free Languages

- Deterministic context-free (DCF) languages include longer-distance phenomena
- DCF grammars have a full-length history, as long as there's no ambiguity (ie. it can't backtrack)
- Processing DCF languages can be done in linear time ($\mathcal{O}(n)$), with linear memory usage ($\mathcal{O}(n)$)

Context-Free Languages

- Context-free languages include phenomena like center embedding
- For example: a b c c′ b′ a′

Context-Free Languages

- Context-free languages include phenomena like center embedding
- For example: a b c c′ b′ a′
- Context-free grammars have a full-length history, and they can backtrack for ambiguous sentences

Context-Free Languages

- Context-free languages include phenomena like center embedding
- For example: a b c c′ b′ a′
- Context-free grammars have a full-length history, and they can backtrack for ambiguous sentences
- Processing CF languages can be done in about cubic time ($\mathcal{O}(n^3)$), with linear memory usage ($\mathcal{O}(n)$)

Mildly Context-Sensitive Languages

- Mildly context-sensitive (MCS) languages include phenomena like reduplication and cross-serial dependencies.
- Example: a b c a′ b′ c′

Mildly Context-Sensitive Languages

- Mildly context-sensitive (MCS) languages include phenomena like reduplication and cross-serial dependencies.
- Example: a b c a′ b′ c′

…das	mer	d'chind	em Hans	es huus	lönd	hälfe	aastriiche
…that	we	the children-ACC	Hans-DAT	house-ACC	let	help	paint

'…that we let the children help Hans paint the house'

Mildly Context-Sensitive Languages

- Mildly context-sensitive (MCS) languages include phenomena like reduplication and cross-serial dependencies.
- Example: a b c a′ b′ c′

…das	mer	d'chind	em Hans	es huus	lönd	hälfe	aastriiche
…that	we	the children-ACC	Hans-DAT	house-ACC	let	help	paint

'…that we let the children help Hans paint the house'

- Processing MCS languages can be done in about $\mathcal{O}(n^6)$ time, with quadratic memory usage $(\mathcal{O}(n^2))$

Mildly Context-Sensitive Languages

- Mildly context-sensitive (MCS) languages include phenomena like reduplication and cross-serial dependencies.
- Example: a b c a′ b′ c′

…das	mer	d'chind	em Hans	es huus	lönd	hälfe	aastriiche
…that	we	the children-ACC	Hans-DAT	house-ACC	let	help	paint

'…that we let the children help Hans paint the house'

- Processing MCS languages can be done in about $\mathcal{O}(n^6)$ time, with quadratic memory usage ($\mathcal{O}(n^2)$)
- Mildly context-sensitive is very different from context-sensitive, which is much more powerful
- Some grammar formalisms that can handle MCS langs:
  - Tree Adjoining Grammar (TAG)
  - Combinatory Categorial Grammar (CCG)
  - Linear Indexed Grammars (LIG) (easy to understand)

Recursively Enumerable Languages

- Recursively enumerable languages allow any string that a computer (or equivalent device) can generate/accept
- There's no guarantee that the computer will ever stop processing the sentence
- Essentially any word can occur in any place in the sentence

Recursively Enumerable Languages

- Recursively enumerable languages allow any string that a computer (or equivalent device) can generate/accept
- There's no guarantee that the computer will ever stop processing the sentence
- Essentially any word can occur in any place in the sentence
- Some grammar formalisms that allow recursively enumerable languages include:
    - Chomskyan grammars (due to transformations / moves)
    - Lexical Functional Grammar (LFG)
    - Head-driven Phrase Structure Grammar (HPSG) (due to SLASH features)

Recursively Enumerable Languages

- Recursively enumerable languages allow any string that a computer (or equivalent device) can generate/accept
- There's no guarantee that the computer will ever stop processing the sentence
- Essentially any word can occur in any place in the sentence
- Some grammar formalisms that allow recursively enumerable languages include:
    - Chomskyan grammars (due to transformations / moves)
    - Lexical Functional Grammar (LFG)
    - Head-driven Phrase Structure Grammar (HPSG) (due to SLASH features)
- Note that these grammar formalisms can place some restrictions on word order, but they still accept/generate recursively enumerable languages.

Recursively Enumerable Languages

- Recursively enumerable languages allow any string that a computer (or equivalent device) can generate/accept
- There's no guarantee that the computer will ever stop processing the sentence
- Essentially any word can occur in any place in the sentence
- Some grammar formalisms that allow recursively enumerable languages include:
  - Chomskyan grammars (due to transformations / moves)
  - Lexical Functional Grammar (LFG)
  - Head-driven Phrase Structure Grammar (HPSG) (due to SLASH features)
- Note that these grammar formalisms can place some restrictions on word order, but they still accept/generate recursively enumerable languages. How is that so?

Recursively Enumerable Languages

- Recursively enumerable languages allow any string that a computer (or equivalent device) can generate/accept
- There's no guarantee that the computer will ever stop processing the sentence
- Essentially any word can occur in any place in the sentence
- Some grammar formalisms that allow recursively enumerable languages include:
  - Chomskyan grammars (due to transformations / moves)
  - Lexical Functional Grammar (LFG)
  - Head-driven Phrase Structure Grammar (HPSG) (due to SLASH features)
- Note that these grammar formalisms can place some restrictions on word order, but they still accept/generate recursively enumerable languages. How is that so? Additional grammar rules can work around such restrictions to accept/generate the string

But that’s just strings. . .

But that's just strings. . .

- Why do we care how the strings are structured?

But that's just strings...

- Why do we care how the strings are structured?
- Because different structures enable different computations!

But that's just strings. . .

- Why do we care how the strings are structured?
- Because different structures enable different computations!
- For example: context-free languages harder to machine-learn than regular languages.

Meaning: something to do with language?

Meaning: something to do with language?

- In some sense, we want to get at the meaning in language.

Meaning: something to do with language?

- In some sense, we want to get at the meaning in language.
- Implicit or explicit meaning?

Meaning: something to do with language?

- In some sense, we want to get at the meaning in language.
- Implicit or explicit meaning?
  - Machine learning: perhaps just map structures in one language to structures in another? No meaning required.

Meaning: something to do with language?

- In some sense, we want to get at the meaning in language.
- Implicit or explicit meaning?
  - Machine learning: perhaps just map structures in one language to structures in another? No meaning required.
  - Computer vision – maybe we really want explicit descriptions of objects in human language.

Lexical representation

- Words have meanings. How do we describe what a word means?

Lexical representation

- Words have meanings. How do we describe what a word means?
- First attempt: use "features."

Lexical representation

- Words have meanings. How do we describe what a word means?
- First attempt: use "features."
  - "bachelor" = +male, +adult, -married

Lexical representation

- Words have meanings. How do we describe what a word means?
- First attempt: use "features."
  - "bachelor" = +male, +adult, -married
  - "husband" = +male, +adult, +married

Lexical representation

- Words have meanings. How do we describe what a word means?
- First attempt: use "features."
    - "bachelor" $=$ $+$male, $+$adult, -married
    - "husband" $=$ $+$male, $+$adult, $+$married
    - "bachelor" $=$ "husband" $\times$ (-married)

Lexical representation

- Words have meanings. How do we describe what a word means?
- First attempt: use "features."
  - "bachelor" = +male, +adult, -married
  - "husband" = +male, +adult, +married
  - "bachelor" = "husband" $\times$ (-married)
- Dictionary problem: what is the meaning of a feature? Define words in terms of other words?

Compositional and sentence meaning

- But sentences have meanings too!

Compositional and sentence meaning

- But sentences have meanings too!
- “The kitten is playing the violin” – DOER: kitten, THING DONE TO: violin, ACTION: play

Compositional and sentence meaning

- But sentences have meanings too!
- "The kitten is playing the violin" – DOER: kitten, THING DONE TO: violin, ACTION: play
- Common way of representing this: first-order predicate calculus.

Compositional and sentence meaning

- But sentences have meanings too!
- "The kitten is playing the violin" – DOER: kitten, THING DONE TO: violin, ACTION: play
- Common way of representing this: first-order predicate calculus.
  - $\exists x \exists y\, kitten(x) \& violin(y) \& play(x, y)$

Compositional and sentence meaning

- But sentences have meanings too!
- "The kitten is playing the violin" – DOER: kitten, THING DONE TO: violin, ACTION: play
- Common way of representing this: first-order predicate calculus.
  - $\exists x \exists y \, kitten(x) \& violin(y) \& play(x, y)$
  - Does this really represent the meaning relationships well?

Compositional and sentence meaning

- But sentences have meanings too!
- "The kitten is playing the violin" – DOER: kitten, THING DONE TO: violin, ACTION: play
- Common way of representing this: first-order predicate calculus.
  - $\exists x \exists y\, kitten(x) \& violin(y) \& play(x, y)$
  - Does this really represent the meaning relationships well?
- The main question of formal semantics: what do we need to reason about language?

To put a long story short. . .

- We want to model complex formal objects robustly.
- The rest of this course is about exploiting continuous distributions of values to do so.