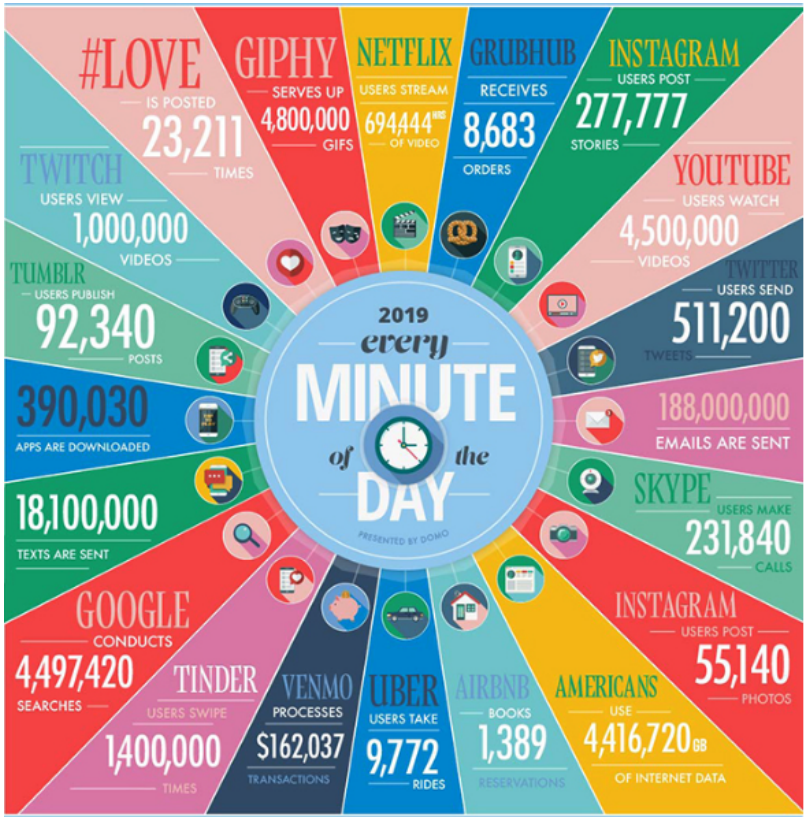
Welcome to 2nd year project!

Natural Language Processing and Deep Learning

Data Never Sleeps

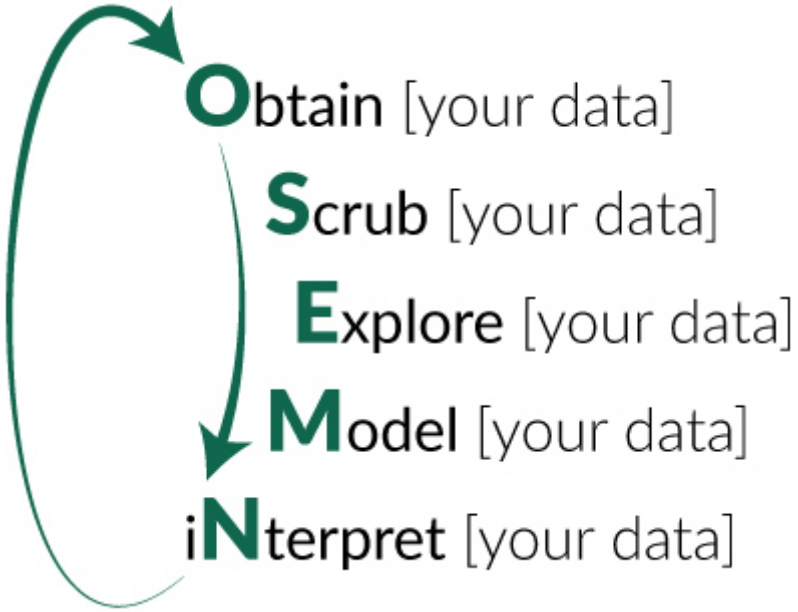
E.g., In 2019, how many tweets were send *every minute*?



Src: DOMO Data never sleeps 7.0 (https://www.domo.com/learn/data-never-sleeps-7) [https://www.domo.com/learn/data-never-sleeps-7 (https://www.domo.com/learn/data-never-sleeps-7) [https://www.domo.com/learn/data-never-sleeps-7)]

Data Science is OSEMN!

- OSEMN model (Hilary Mason, 2010)
- Pronounced 'awesome'

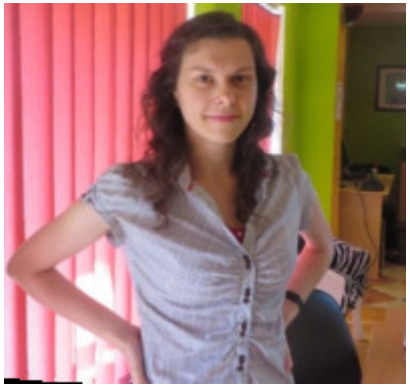

Obtain [your data]
Scrub [your data]
Explore [your data]
Model [your data]
iNterpret [your data]

What is Natural Language Processing (NLP)?

- an **interdisciplinary** research field
- **Goal:** enabling computers to **understand** and **generate** language, just as we humans do
- Deep understanding of **broad** language
    - not just string processing or keyword matching

Course practicalities

Who are we?



Teachers

- Barbara Plank, http://bplank.github.io (http://bplank.github.io)
- Rob van der Goot, http://www.robvandergoot.com (http://www.robvandergoot.com)

Teaching assistants

- Marija Stepanović (TA lead)
- Kristian Nørgaard Jensen
- Nadia Krag

Who are you?

Books

(also see references on LearnIt)

Course outcomes

- Discuss, clearly explain, and reflect upon central concepts, algorithms, and challenges in natural language processing (NLP) and deep learning (DL).
- Organize, plan, and carry out collaborative work in a smaller project group.
- Obtain, scrub, explore and preprocess a wide range of relevant raw data for a given problem. - Identify and analyze the relevant options for data collection and preprocessing and select the most suitable ones.
- Design and implement a sound experiment in NLP
- Distinguish and evaluate the advantages of different design choices or approaches to the same task (e.g., traditional versus deep-learning based solutions)
- Evaluate the achieved solution and carry out a detailed error analysis, relating the findings back to the overall problem domain
- Explain in writing (project group report) adhering to academic standards in writing
- Succinctly present the results of the project, discuss findings and limitations
- Reflect upon ethical considerations that arise in the deployment of language technology

Schedule

Part I (week 5-11): Lecture phase

- Lectures:
    - Tuesday 10:00-12:00 room 2A56 (Aud 3)
    - Friday 10:00-12:00 room 2A56 (Aud 3)
- Labs:
    - Tuesday 12:00-14:00 room **4A14** + 4A20
    - Friday 12:00-14:00 room **4A20** + 4A22
    - Note: boldface (lower number) room is the main room, rest for overflow
    - Each week we release exercises which are mandatory (more information next)

Group formation (week 11): Project phase starts

- Group formation day and project kick-off day on March 13

Part II (week 12-19): Project phase

- Tuesday 12:00-14:00 check in (*group needs to present an update*)
- Friday 12:00-14:00 work on project
- Continuous project work in weeks 12-19, check-ins and project work during exercise hours
- Required use of group-specific project github repository hosted on github.itu.dk

Course Project: Robust Sentiment Analysis & Opinion Mining

- The web is full of opinionated texts such as review text. The aim of the project is to extract sentiment and opinions from large scale web resources, addressing a fundamental challenge:
- How to make a Sentiment analysis systems **more robust** over **text domains**?

Course Project: Robust Sentiment Analysis & Opinion Mining

Methods involved, amongst others:

- Data scrubbing and cleaning (processing)
- Modeling: Comparison of traditional ML versus Neural Approaches, Error Analysis

Computing:

- New this year: HPC cluster (high-performance computing cluster)!

Main Data Provider:

Surprise by Customer (phase 2)

Guest lecturers

- Marija Stepanović and Andreas Søeborg Kirkedal (March 3 and 6) *speech data*
- Dirk Hovy (March 10) from Bocconi University on Bias and Fairness in NLP

Course Materials

Lecture Preparation

- Read the provided background material
- Do the exercises in the lab!
- There might be quizzes in class

Course Assessment and Course structure

- Part I (lectures and labs):
    - **prepares** you for the project phase
    - Labs include exercises to help get you deeper into the material and are obligatory (they are checked but not graded)
    - Work in week t on your lab exercise of the week, TAs will check parts of your solution in week $t + 1$ (randomly draw on part)
    - No need to upload your solutions, but bring them to the next lab in week $t + 1$

Assessment

- **Final project (100%)**, to be completed in a group of 4-5 students
    - Released: in week 11, hand-in: **May 12, 2019** at 14:00 (learnIt) [this is a preset, fixed data, no extension!]

Late Hand-In

- Late hand-ins **cannot be accepted**
- Exceptions can be made in rare cases, e.g. due to illness with doctor's notice
    - Get in touch with SAP at least one working day in advance see here (https://studyguide.itu.dk/ds/your-programme/exams/illness)

Plagiarism

- Don't do it
- Don't enable it
- Check rules and consequences (https://student-ambassador.ku.dk/rights/avoid-plagiarism/) if unclear

Final project report and Exam

- Group presentation (slides) based on **group report (final project report)** handed in on May 12 (group report has to use a scientific LaTeX paper style, more details later) and developed **code** (including data)
- Group presentation is followed by individual exam
- External examiner
- Exam dates: June 2,3 and 4, 2020 -- **reserve these dates in your calendar!!**

Python

- Lectures, lab exercises and assignments focus on **Python** (Anaconda Python version >3.6)
- Python is a leading language for data science, machine learning etc., with many relevant libraries
- Labs and assignments focus on the development of a mix of: standalone Python code, use of Unix command line tools and development within jupyter notebooks (http://jupyter.org/)

Important Dates 2020 - Summary

Note: keep an eye on LearnIt for details

- January 31 to March 10 Lecture Phase
- March 13 group formation day & Project kick-off
    - phase 1 - group presentations March 27 (Friday)
    - phase 2 kick-off March 31 (external guest) - group presentations April 17 (Friday, external guest)
    - phase 3 - final report due on May 12
- Exam (oral, incl. external censor): June 2-4, 2020

How to reach us?

Slack

- We have a **dedicated Slack channel**: https://2ndyearproject-2020.slack.com (https://2ndyearproject-2020.slack.com)
- Please post questions there (instead of private emails)
- We give low priority to **questions already answered** in previous lectures, tutorials and posts,
    - and to **pure programming related issues**
- We expect you to **online-search** for answers before.
- You are highly encouraged to participate and **help each other** on the channel and during the labs.
- The teaching team will check the discussion on slack regularly **within normal working hours**
    - do not expect answers late in the evenings and on weekends
    - **start working on your problem sets and project early**
    - come to the lab sessions and ask questions there

Any questions?

Introduction to Natural Language Processing (NLP)

After this lecture you should:

- know what Natural Language Processing (NLP) is and why language is so challenging
- have refreshed your memory on regular expressions

What's so special about human language?


What we say to dogs
Okay, Ginger! I've had it! You stay out of the garbage! Understand, Ginger? Stay out of the garbage, or else!


What they hear
blah blah GINGER blah blah blah blah blah blah blah GINGER blah blah blah blah...

- most important *distinctive* human characteristics
- the hard part in AI (intelligence)
- *communication* was central in human development

NLP: Where are we now?

Can you think of an NLP application that you use regularly?

I'm sure there is at least one. ;-)

Do you know these...?

Personal assistants - Everywhere!



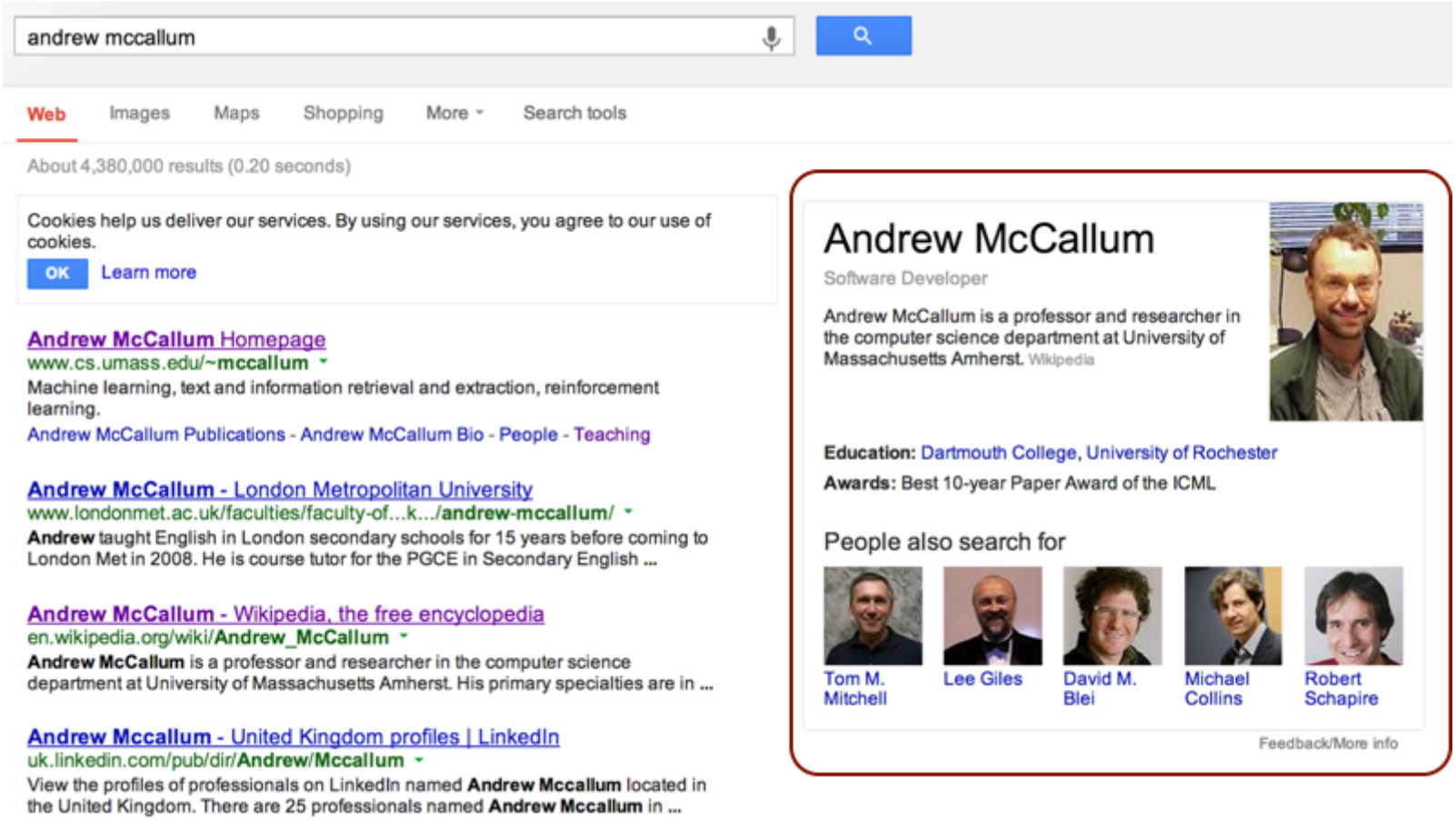

Speech Recognition

Speech Recognition is usually not considered core NLP. We will delve a bit into it in March.

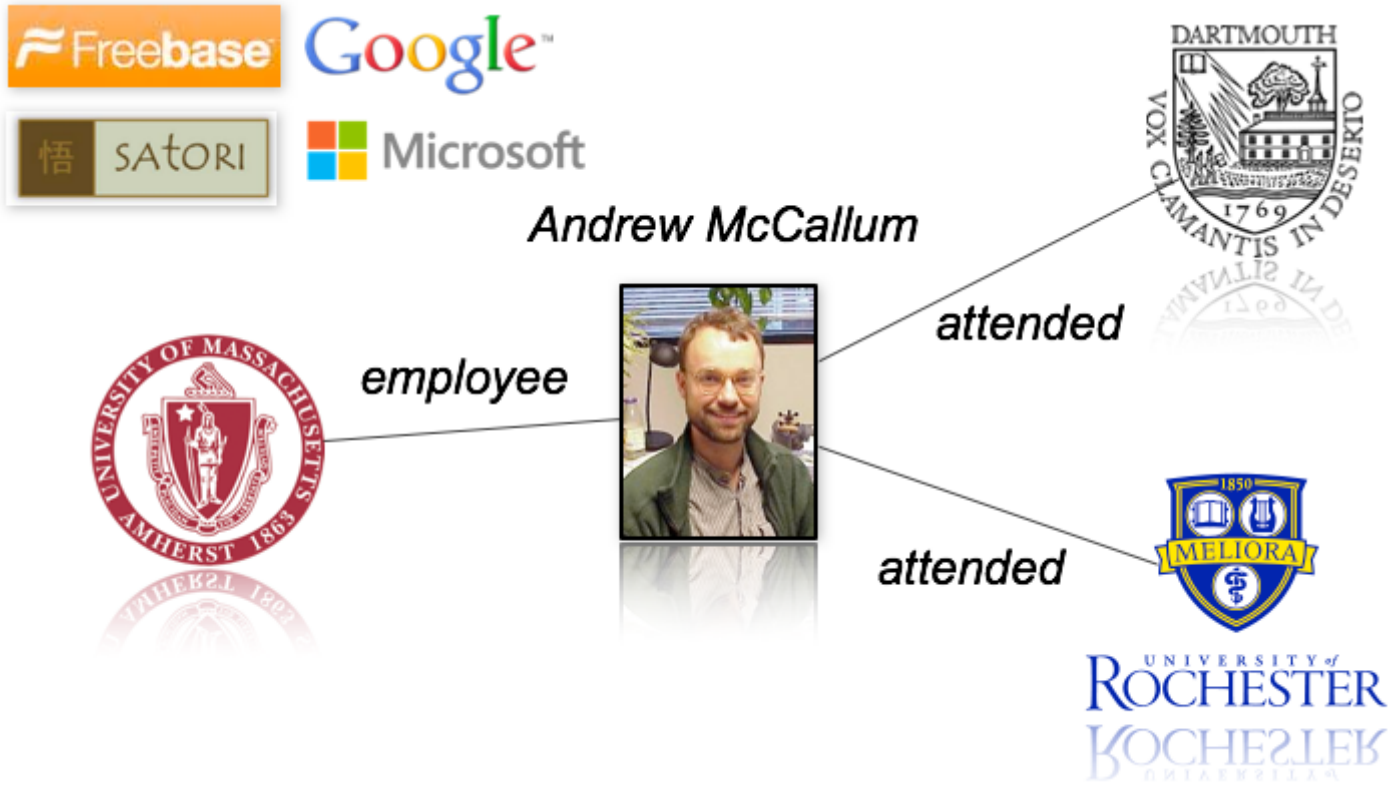
Machine Translation

http://translate.google.com/ (http://translate.google.com/)

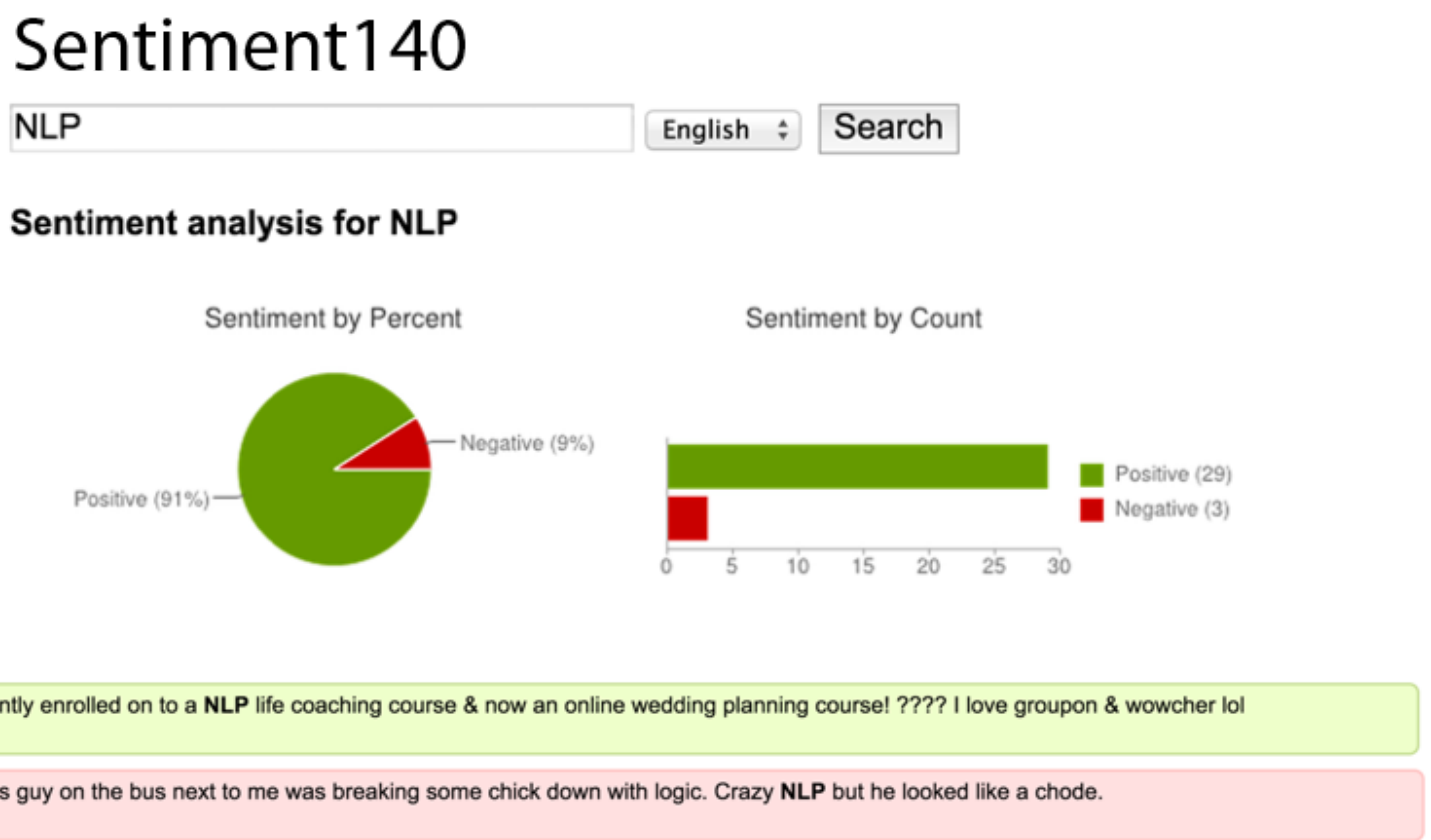
Information Extraction



Information Extraction



Sentiment Analysis



Why is it so difficult?

Are these sentences ok?

- The cat sat on the mat.
- Sat the cat mat the on.
- Mary went store.
- Mary goed to the store.
- The store went to Mary.

What is so difficult about NLP?

Example: *I made her duck*

What is the meaning of this sentence?

(Is there only one meaning?)

1. I cooked a duck for her
2. I cooked a duck that belonged to her
3. I created a (plastic?) duck she owns
4. I caused her to quickly lower her upper body
5. I turned her into a duck (magic!)

Ambiguity is everywhere

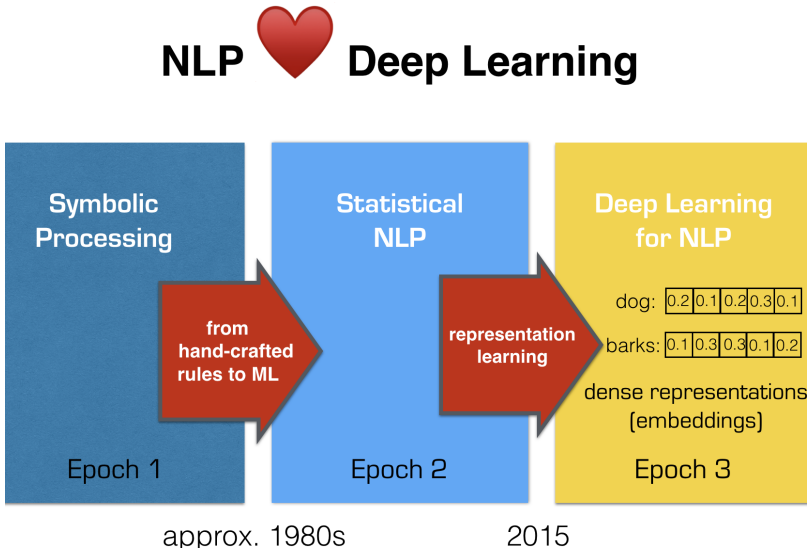
- Fed **raises** interest rates 0.5% in effort to control inflation
- Fed raises **interest** rates 0.5% in effort to control inflation
- Fed raises interest **rates** 0.5% in effort to control inflation

Ambiguity of language is manifested at many linguistic levels:

- lexical
- syntax
- semantics (world knowledge)
- pragmatics

Further challenges: multilinguality, morphology ...

My one slide history of the field:



"Neural Networks: A Tool for Doing Hard Things." (Graham Neubig)

What's a word? - Tokenization

- Recap: tokenization is to identify the **words** in a string of characters.

In Python you can tokenise a text via `split` :

In [3]:
```
text = """Mr. Bob Dobolina is thinkin' of a master plan.
Why doesn't he quit?"""
text.split(" ")
```

Out[3]:
```
['Mr.',
 'Bob',
 'Dobolina',
 'is',
 "thinkin'",
 'of',
 'a',
 'master',
 'plan.\nWhy',
 "doesn't",
 'he',
 'quit?']
```

Why is this suboptimal?

Python allows users to construct tokenisers using

Regular Expressions

- A formal language for specifying text strings (an algebraic notation for characterizing a set of strings)
- Can be used to find subsets of text, or in tokenization to define **patterns** at which to split tokens.

Regular Expressions: Disjunctions

- Letters inside square brackets []

Pattern	Matches
`[wW]oodchuck`	Woodchuck, woodchuck
`[1234567890]`	Any digit

- Ranges `[A-Z]`

Pattern	Matches	
`[A-Z]`	An upper case letter	Drenched Blossoms
`[a-z]`	A lower case letter	my beans were impatient
`[0-9]`	A single digit	Chapter 1: Down the Rabbit Hole

Slides from J&M (https://web.stanford.edu/~jurafsky/slp3/slides/2_TextProc.pdf)

Regular Expressions: Negation in Disjunction

- Negations `[^Ss]`
  - Carat means negation only when first in []

Pattern	Matches	
`[^A-Z]`	Not an upper case letter	`Oyfn pripetchik`
`[^Ss]`	Neither 'S' nor 's'	`I have no exquisite reason"`
`a^b`	The pattern a carat b	`Look up a^b now`

Slides from J&M (https://web.stanford.edu/~jurafsky/slp3/slides/2_TextProc.pdf)

Regular Expressions: More Disjunction

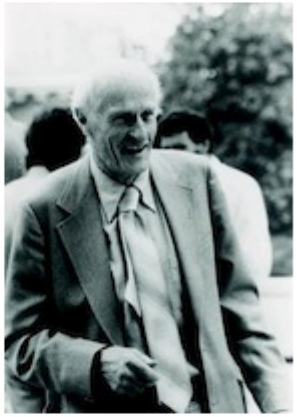
- Woodchucks is another name for groundhog!
- The pipe | for disjunction

Pattern	Matches
`groundhog|woodchuck`	
`yours|mine`	yours mine
`a|b|c`	= [abc]
`[gG]roundhog|[Ww]oodchuck`	

Slides from J&M (https://web.stanford.edu/~jurafsky/slp3/slides/2_TextProc.pdf)

Regular Expressions: ? * + .

Pattern	Matches	
colou?r	Optional previous char	color colour
oo*h!	0 or more of previous char	oh! ooh! oooh! ooooh!
o+h!	1 or more of previous char	oh! ooh! oooh! ooooh!
baa+		baa baaa baaaa baaaaa
beg.n		begin begun begun beg3n



Stephen C Kleene

Kleene *, Kleene +

Slides from J&M (https://web.stanford.edu/~jurafsky/slp3/slides/2_TextProc.pdf)

Regular Expressions: Anchors ^ $

Pattern	Matches
^[A-Z]	Palo Alto
^[^A-Za-z]	1 "Hello"
\.$	The end.
.$	The end? The end!

Slides from J&M (https://web.stanford.edu/~jurafsky/slp3/slides/2_TextProc.pdf)

Example

- Find me all instances of the word “the” in a text.

```
the
```

Misses capitalized examples

```
[tT]he
```

Incorrectly returns `other` or `theology`

```
[^a-zA-Z][tT]he[^a-zA-Z]
```

Slides from J&M (https://web.stanford.edu/~jurafsky/slp3/slides/2_TextProc.pdf)

A **regular expression** is a compact definition of a **set** of (character) sequences.

Examples:

- `"Mr\."`: set containing only `"Mr."`
- `"  |\n|!!!"`: set containing the sequences `"  "`, `"\n"` and `"!!!"`
- `"[abc]"`: set containing only the characters `a`, `b` and `c`
- `"\s"`: set of all whitespace characters
- `"1+"`: set of all sequences of at least one `"1"`
- etc.

In [4]:
```
import re
re.compile('\s').split(text)
```

Out[4]:
```
['Mr.',
 'Bob',
 'Dobolina',
 'is',
 "thinkin'",
 'of',
 'a',
 'master',
 'plan.',
 'Why',
 "doesn't",
 'he',
 'quit?']
```

Problems:

- Bad treatment of punctuation.
- Easier to **define a token** than a gap.

Let us use `findall` instead:

In [5]:
```
re.compile('\w+|[.?]').findall(text)
```

Out[5]:
```
['Mr',
 '.',
 'Bob',
 'Dobolina',
 'is',
 'thinkin',
 'of',
 'a',
 'master',
 'plan',
 '.',
 'Why',
 'doesn',
 't',
 'he',
 'quit',
 '?']
```

Problems:

- "Mr." is split into two tokens, should be single.
- Lost an apostrophe.

Both is fixed below ...

In [6]:
```
re.compile('Mr.|[\w\']+|[.?]').findall(text)
```

Out[6]:
```
['Mr.',
 'Bob',
 'Dobolina',
 'is',
 "thinkin'",
 'of',
 'a',
 'master',
 'plan',
 '.',
 'Why',
 "doesn't",
 'he',
 'quit',
 '?']
```

Learning to Tokenise?

- For English simple pattern matching often sufficient.
- In other languages (e.g. Japanese), words are not separated by whitespace.

In [7]:
```
jap = "今日もしないといけない。"
```

Try lexicon-based tokenisation ...

In [8]:
```
re.compile('もし|今日|も|しない|と|けない').findall(jap)
```

Out[8]:
```
['今日', 'もし', 'と', 'けない']
```

Equally complex for certain English domains (eg. bio-medical text).

In [8]:
```
bio = """We developed a nanocarrier system of herceptin-conjugated nanoparticles
of d-alpha-tocopheryl-co-poly(ethylene glycol) 1000 succinate (TPGS)-cisplatin
prodrug ..."""
```

- d-alpha-tocopheryl-co-poly is **one** token
- (TPGS)-cisplatin are **five**:
    - (
    - TPGS
    -)
    - -
    - cisplatin

In [9]:
```
re.compile('\s').split(bio)[:15]
```

Out[9]:
```
['We',
 'developed',
 'a',
 'nanocarrier',
 'system',
 'of',
 'herceptin-conjugated',
 'nanoparticles',
 'of',
 'd-alpha-tocopheryl-co-poly(ethylene',
 'glycol)',
 '1000',
 'succinate',
 '(TPGS)-cisplatin',
 'prodrug']
```

Solution: Treat tokenisation as a **statistical NLP problem** (structured prediction). More on this later.

References

- Jurafsky & Martin, Speech and Language Processing (Third Edition) (https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf): Chapter 2, Regular Expressions, Text Normalization, Edit Distance.
- Manning, Raghavan & Schuetze, Introduction to Information Retrieval: Tokenization (http://nlp.stanford.edu/IR-book/html/htmledition/tokenization-1.html)