Natural Language Processing

Artificial Intelligence @ Allegheny College

Janyl Jumadinova

October 20–22, 2021

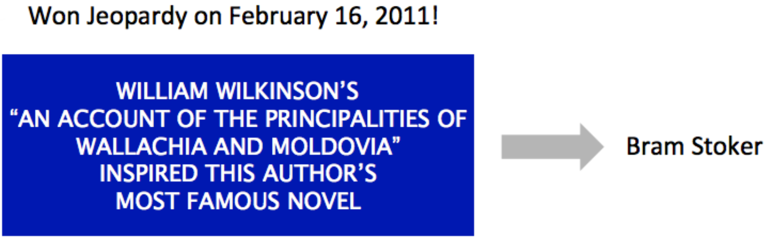
Credit: NLP Stanford

NLP

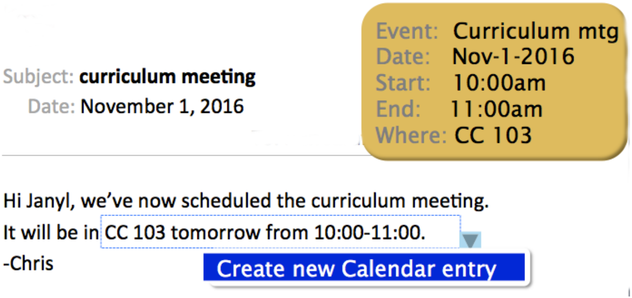
Natural Language Processing

Understand, interpret and manipulate natural language

Question Answering: IBM's Watson

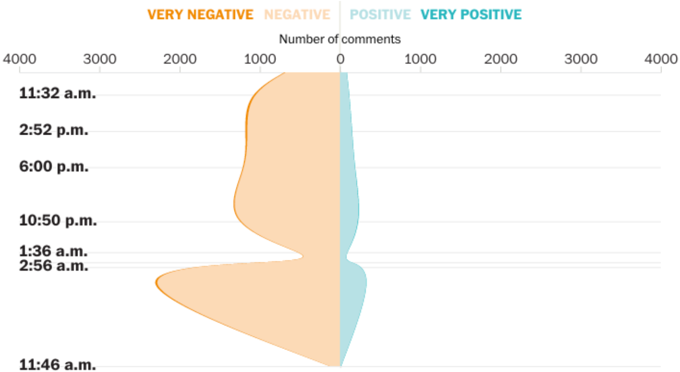


Information Extraction



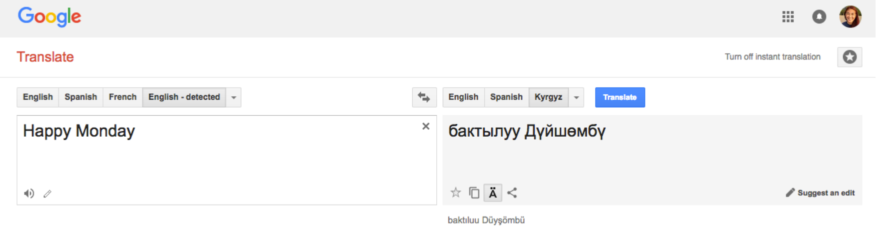
Sentiment Extraction

2016 Election

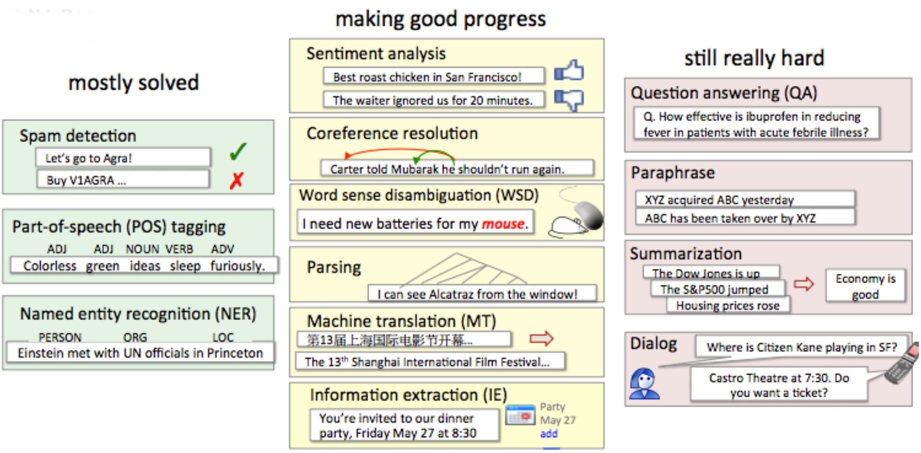


Source: Washington Post

Machine Translation



Language Technology



Ambiguity makes NLP hard

Ambiguity makes NLP hard

- Teacher Strikes Idle Kids

Ambiguity makes NLP hard

- Teacher Strikes Idle Kids
- Red Tape Holds Up New Bridges

Ambiguity makes NLP hard

- Teacher Strikes Idle Kids
- Red Tape Holds Up New Bridges
- Juvenile Court to Try Shooting Defendant

Ambiguity makes NLP hard

- Teacher Strikes Idle Kids
- Red Tape Holds Up New Bridges
- Juvenile Court to Try Shooting Defendant
- Local High School Dropouts Cut in Half

Other NLP Difficulties

Progress

- What tools do we need?
    - Knowledge about language
    - Knowledge about the world
    - A way to combine knowledge sources

Progress

- What tools do we need?
  - Knowledge about language
  - Knowledge about the world
  - A way to combine knowledge sources
- How we generally do this:
  - Probabilistic models built from language data
  - P("maison" $\rightarrow$ "house") $\rightarrow$ high
  - P("L'avocat general" $\rightarrow$ "the general avocado") $\rightarrow$ low

Word tokenization

Every NLP task needs to do text normalization:

1. **Segmenting/tokenizing words in running text**
2. Normalizing word formats
3. Segmenting sentences in running text

How Many Words?

N = number of tokens

V = vocabulary = set of types

$|V|$ is the size of the vocabulary

Church and Gale (1990): $|V| > O(N^{½})$

	Tokens = N	Types = |V|
Switchboard phone conversations	2.4 million	20 thousand
Shakespeare	884,000	31 thousand
Google N-grams	1 trillion	13 million

N - all words

V - distinct words

Normalization

Every NLP task needs to do text normalization:

1. Segmenting/tokenizing words in running text
2. **Normalizing word formats**
3. Segmenting sentences in running text

Issues in Tokenization

- Finland's capital $\rightarrow$ Finland Finlands Finland's
- what're, I'm, isn't $\rightarrow$ What are, I am, is not
- Hewlett-Packard $\rightarrow$ Hewlett Packard
- state-of-the-art $\rightarrow$ state of the art
- Lowercase $\rightarrow$ lower-case lowercase lower case
- San Francisco $\rightarrow$ one token or two?

Issues in Tokenization

- Finland's capital → Finland Finlands Finland's
- what're, I'm, isn't → What are, I am, is not
- Hewlett-Packard → Hewlett Packard
- state-of-the-art → state of the art
- Lowercase → lower-case lowercase lower case
- San Francisco → one token or two?
- **Language Issues**: French, German, Japanese, Chinese,...

Issues in Tokenization

- Finland's capital $\rightarrow$ Finland Finlands Finland's
- what're, I'm, isn't $\rightarrow$ What are, I am, is not
- Hewlett-Packard $\rightarrow$ Hewlett Packard
- state-of-the-art $\rightarrow$ state of the art
- Lowercase $\rightarrow$ lower-case lowercase lower case
- San Francisco $\rightarrow$ one token or two?
- **Language Issues**: French, German, Japanese, Chinese,...

Normalization:
merging of different forms of a token into a canonical normalized form.
- ex.: "Mr.", "Mr", "mister", and "Mister" into a single form.

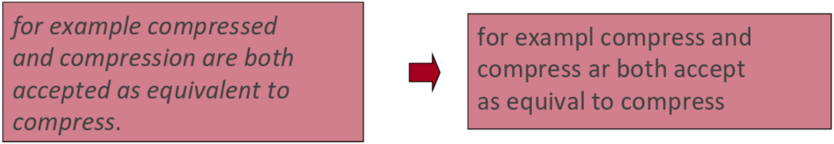
Stemming

Every NLP task needs to do text normalization:

1. Segmenting/tokenizing words in running text
2. Normalizing word formats
3. **Segmenting sentences in running text**

Stemming

- Reduce terms to their stems in information retrieval
- **Stemming** is crude chopping of affixes language dependent
- Example: **automate(s)**, **automatic**, **automation** all reduced to **automat**.



Porter's Algorithm

Most common English stemmer.

Step 1a

```
sses → ss    caresses → caress
ies  → i     ponies   → poni
ss   → ss    caress   → caress
s    → ø     cats     → cat
```

Step 1b

```
(*v*)ing → ø   walking   → walk
               sing      → sing
(*v*)ed  → ø   plastered → plaster
…
```

Step 2 (for long stems)

```
ational→ ate  relational→ relate
izer→ ize     digitizer → digitize
ator→ ate     operator  → operate
…
```

Step 3 (for longer stems)

```
al    → ø   revival    → reviv
able  → ø   adjustable → adjust
ate   → ø   activate   → activ
…
```

Sentence Segmentation

- !, ? are relatively unambiguous

Sentence Segmentation

- !, ? are relatively unambiguous
- Period "." is quite ambiguous
  - Sentence boundary
  - Abbreviations like Inc. or Dr.
  - Numbers like .02 or 4.3

Sentence Segmentation

- !, ? are relatively unambiguous
- Period "." is quite ambiguous
  - Sentence boundary
  - Abbreviations like Inc. or Dr.
  - Numbers like .02 or 4.3
- Build a binary classifier
  - Classifiers: hand--written rules, regular expressions, or machine--learning

Information Extraction (IE)

- Find and understand limited relevant parts of texts
- Gather information from many pieces of text
- Produce a structured representation of relevant information

Information Extraction

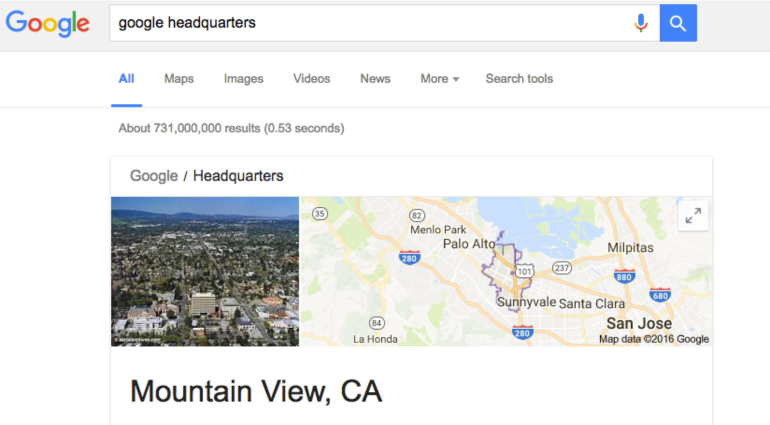
Goals:
- Organize information so that it is useful to people
- Put information in a semantically precise form that allows further inferences to be made by computer algorithms

Information Extraction

Goals:
- Organize information so that it is useful to people
- Put information in a semantically precise form that allows further inferences to be made by computer algorithms

Roughly: **Who did what to whom when?**

Low-level information extraction



Named Entity Recognition (NER)

A very important sub-task: **find** and classify names in text

The decision by the independent MP Andrew Wilkie to withdraw his support for the minority Labor government sounded dramatic but it should not further threaten its stability. When, after the 2010 election, Wilkie, Rob Oakeshott, Tony Windsor and the Greens agreed to support Labor, they gave just two guarantees: confidence and supply.

Named Entity Recognition (NER)

A very important sub-task: find and **classify** names in text

The decision by the independent MP Andrew Wilkie to withdraw his support for the minority Labor government sounded dramatic but it should not further threaten its stability. When, after the 2010 election, Wilkie, Rob Oakeshott, Tony Windsor and the Greens agreed to support Labor, they gave just two guarantees: confidence and supply.

Person
Date
Location
Organization

Named Entity Recognition (NER)

The uses:

- Named entities can be indexed, linked, etc.
- Sentiment can be attributed to companies or products
- A lot of IE relations are associations between named entities
- For question answering, answers are often named entities

Named Entity Recognition (NER)

- Data $\{(c, d)\}$ of paired observations d and hidden classes c
- **Features** f are elementary pieces of evidence that link aspects of what we observe d with a category c that we want to predict

Named Entity Recognition (NER)

- Data $\{(c, d)\}$ of paired observations d and hidden classes c
- **Features** f are elementary pieces of evidence that link aspects of what we observe d with a category c that we want to predict

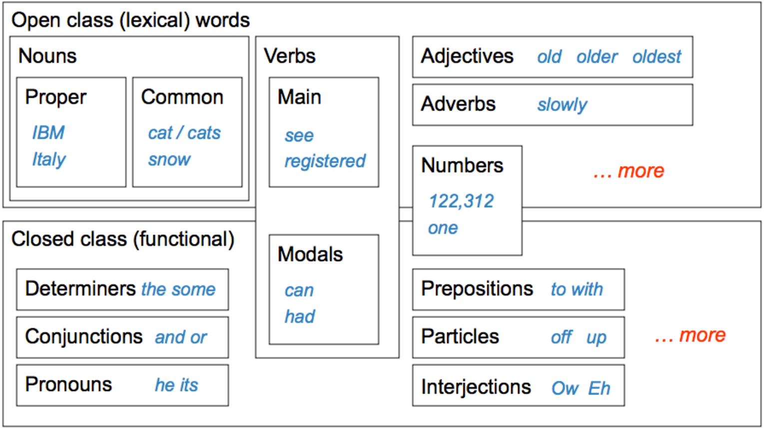
$$f_1(c, d) \equiv [c = \text{LOCATION} \wedge w_{-1} = \text{“in”} \wedge \text{isCapitalized}(w)]$$

$$f_2(c, d) \equiv [c = \text{LOCATION} \wedge \text{hasAccentedLatinChar}(w)]$$

$$f_3(c, d) \equiv [c = \text{DRUG} \wedge \text{ends}(w, \text{“c”})]$$

LOCATION	LOCATION	DRUG	PERSON
in Arcadia	*in Québec*	*taking Zantac*	*saw Sue*

Parts of Speech (POS)



POS Tagging

Words often have more than one POS:

- The *back* door
- On my *back*
- Win the voters *back*
- Promised to *back* the bill

POS Tagging

Words often have more than one POS:

- The *back* door
- On my *back*
- Win the voters *back*
- Promised to *back* the bill

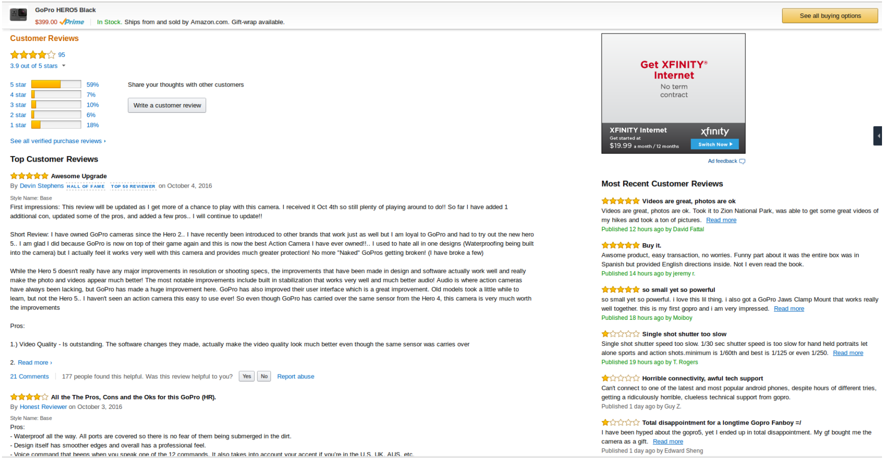
The **POS tagging problem** is to determine the POS tag for a particular instance of a word.

POS Tagging

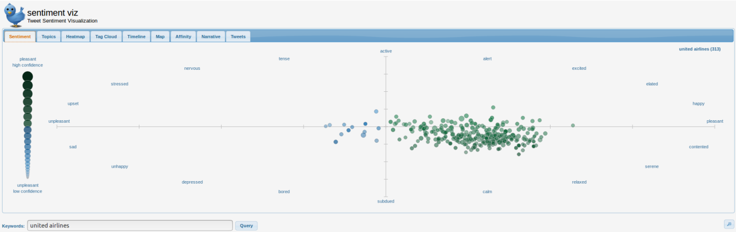
- **Input**: Plays well with others
- **Ambiguity**: NNS/VBZ UH/JJ/NN/RB IN NNS
- **Output**: Plays/VBZ well/RB with/IN others/NNS

Penn Treebank Tag-set

Sentiment Analysis



Sentiment Analysis



- https://www.nltk.org/howto/sentiment.html
- https://nlp.stanford.edu/sentiment/
- https://textblob.readthedocs.io/en/dev/

Sentiment analysis has many other names

- Opinion extraction
- Opinion mining
- Sentiment mining
- Subjectivity analysis

Sentiment Analysis

Sentiment analysis is the detection of **attitudes**

- "enduring, affectively colored beliefs, dispositions towards objects or persons"

Attitudes

- **Holder** (source) of attitude
- **Target** (aspect) of attitude

Attitudes

- **Holder** (source) of attitude
- **Target** (aspect) of attitude
- **Type** of attitude
  - From a set of types:
  Like, love, hate, value, desire, etc.
  - Or (more commonly) simple weighted polarity:
  positive, negative, neutral, together with strength

Attitudes

- **Holder** (source) of attitude
- **Target** (aspect) of attitude
- **Type** of attitude
  - From a set of types:
  Like, love, hate, value, desire, etc.
  - Or (more commonly) simple weighted polarity:
  positive, negative, neutral, together with strength
- **Text** containing the attitude
  - Sentence or entire document

Sentiment analysis

- **Simplest task**:
  Is the attitude of this text positive or negative?

Sentiment analysis

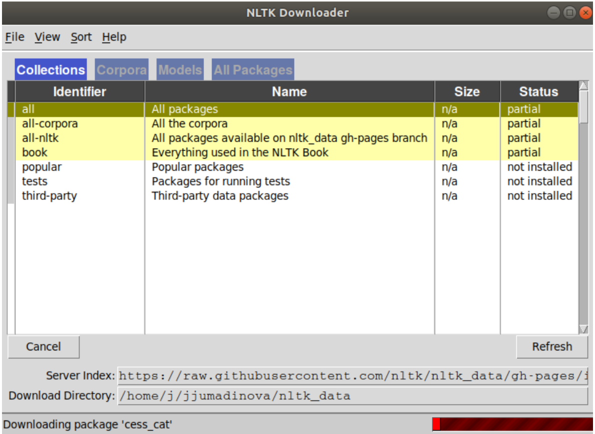
- **Simplest task**:
  Is the attitude of this text positive or negative?
- **More complex**:
  Rank the attitude of this text from 1 to 5

Sentiment analysis

- **Simplest task**:
  Is the attitude of this text positive or negative?
- **More complex**:
  Rank the attitude of this text from 1 to 5
- **Advanced**:
  Detect the target, source, or complex attitude types

NLTK

```
$ python3
$ import nltk
$ nltk.download()
```



NLTK Basic Pre-Processing

Tokenize using Python

1. `urllin module` to crawl the webpage
2. `BeautifulSoup` to clean the text with html tags
3. convert text into tokens using `split()` function

NLTK Basic Pre-Processing

Tokenize using Python

1. `urllin module` to crawl the webpage
2. `BeautifulSoup` to clean the text with html tags
3. convert text into tokens using `split()` function

Remove Stop Words

1. get english stop words from nltk
2. remove stop words before plotting

NLTK Basic Pre-Processing

Tokenize using Python

1. `urllin module` to crawl the webpage
2. `BeautifulSoup` to clean the text with html tags
3. convert text into tokens using `split()` function

Remove Stop Words

1. get english stop words from nltk
2. remove stop words before plotting

Frequency Analysis

1. nltk's `FreqDist` to calculate the frequency distribution
2. `plot` function to produce a graph