Natural Language Processing (NLP)

Objectives

- Define NLP and describe several use cases
- Explain why NLP is hard
    - Or maybe easy to do (with sklearn) but hard to master
- Describe (and be able to execute) a “typical” text processing workflow
    - Use relevant NLP vocabulary
- Calculate, by hand and in-code, *tf matrices* and *tf-idf matrices*
- tf-idf in Sklearn
- Introduce nltk and spaCy libraries

What is NLP?

Natural-language processing (NLP) is an area of computer science and artificial intelligence concerned with the interactions between computers and human languages. In particular: how to program computers to *fruitfully* process large amounts of natural language data?

-Wikipedia

NLP history

1950: Alan Turing proposes Turing test.

1954: Automatic translation of 60 Russian sentences to English.

1966: 10 years of research in machine translation failed expectations.

1970's: Develop conceptual frameworks.

1980's: Up to this point, most NLP based on human-defined rules, but begin use of machine learning. For example, chatterbot Jabberwacky

To present: Transition from decision-tree algorithms (hard if-then rules) to statistical models which make soft, probabilistic decisions based on attaching weights to the features (words), to recent deep learning models.

And an anonymous version of the Turing Test has arguably been passed.

NLP use cases

Conversational Agents

- Siri, Cortana, Google Home, Alexa
- Talking to your car
- Communicating with robots

Machine Translation

- Google Translate

Speech Recognition, Speech Synthesis

Lexical Semantics, Sentiment Analysis

Dialogue Systems, Question Answering

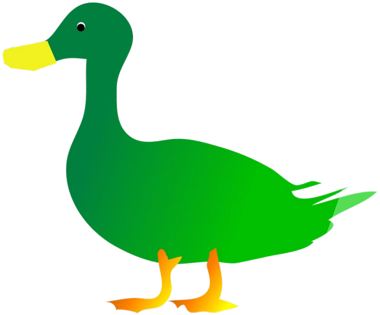
[Conversational Agent Lecture - Dan Jurafsky (Stanford)]

NLP is hard

What is the meaning of this sentence:

I made her duck.

NLP is hard



Ambiguity

"I made her duck"

- I cooked waterfowl for her
- I cooked waterfowl belonging to her
- I created the duck that she owns
- I caused her to quickly lower her head or body
- I waved my magic wand and turned her into undifferentiated waterfowl

This problem of determining which sense was meant by a specific word is formally known as *word sense disambiguation*.

Other examples:
"Court to try shooting defendant."
"Hospitals are sued by seven foot doctors."

NLP is hard

© 2014 Ted Goff
KDnuggets Cartoon

“The machine learning algorithm wants to know if we’d like a dozen wireless mice to feed the Python book we just bought.”

NLP is hard, and so we study it

Phonetics & Phonology (linguistic sounds)

Morphology (meaningful components of words)

Semantics (meaning)

Pragmatics (meaning with respect to goals and intentions)

Discourse (linguistic units larger than a single utterance)

For many machine-learning applications we want text & dialogue that makes sense (has semantic meaning), but many of our tools get more at syntax (syntactic meaning).

"*An ant ate an aunt.*" Syntactically correct! Semantically a disaster...

NLP vocabulary and concepts

corpus: A collection of documents. Usually each row in the corpus is a document. (So each row in your X matrix is usually a document).

stop-words: Domain-specific words that are so common that they are not expected to help differentiate documents. sklearn's stop-words list.

tokens: What documents are made of, and the columns in your X matrix. You'd think tokens are words (and you're right), but tokens are words that have been either stemmed or lemmatized to their root form: car, cars, car's, cars' ⇒ car

n-grams: How many tokens constitute a linguistic unit in the analysis?

- boy (n-gram = 1), little boy (n-gram=2), little boy blue (n-gram=3)

bag-of-words: A document is represented numerically by the set of its tokens, not preserving order and nearby tokens but preserving counts (frequency).

NLP text processing workflow

1. Lowercase all your text (unless for some words that are Part-Of-Speech (POS) the capitalization you decide is important.)
2. Strip out miscellaneous spacing and punctuation.
3. Remove stop words (careful they may be domain or use-case specific).
4. Stem or lemmatize the text into tokens.
5. Convert text to numbers using a bag-of-words model and a term-frequency, inverse document frequency matrix (more later).
6. Train / cluster your data in a machine learning model.

Other: Part-Of-Speech tagging, expand feature matrix with N-grams

Text processing workflow example

Original

Document	The corpus
0	Oh, the thinks you can think if you only try.
1	If you try, you can think up a guff going by.
2	And what would you do if you met a jaboo?

Text processing workflow example

Original

Document	The corpus
0	Oh, the thinks you can think if you only try.
1	If you try, you can think up a guff going by.
2	And what would you do if you met a jaboo?

Lowercase text and strip out punctuation.

Document	The corpus
0	oh the thinks you can think if you only try
1	if you try you can think up a guff going by
2	and what would you do if you met a jaboo

Text processing workflow example

Lowercase text and strip out punctuation

Document	The corpus
0	oh the thinks you can think if you only try
1	if you try you can think up a guff going by
2	and what would you do if you met a jaboo

Remove stop-words

Document	The corpus
0	oh thinks think try
1	try think guff
2	met jaboo

Text processing workflow example

Remove stop words

Document	The corpus
0	oh thinks think try
1	try think guff
2	met jaboo

Lemmatize words -> tokens

Document	The corpus
0	oh think think try
1	try think guff
2	meet jaboo

Text processing workflow example

Lemmatize words -> tokens

Document	The corpus
0	oh think think try
1	try think guff
2	meet jaboo

Work towards a tf-idf matrix (here showing simplest version of tf)

Document	guff (0)	jaboo (1)	meet (2)	think (3)	try (4)
0	0	0	0	2	1
1	1	0	0	1	1
2	0	1	1	0	0

Issues with just counting approach?

Document	guff (0)	jaboo (1)	meet (2)	think (3)	try (4)
0	0	0	0	2	1
1	1	0	0	1	1
2	0	1	1	0	0

Issues with just counting approach?

Document	guff (0)	jaboo (1)	meet (2)	think (3)	try (4)
0	0	0	0	2	1
1	1	0	0	1	1
2	0	1	1	0	0

1. What if documents are of different length? e.g. 10 words vs. 10,000 words.

2. What if some terms simply dominate across all documents?

Issues with just counting approach?

Document	guff (0)	jaboo (1)	meet (2)	think (3)	try (4)
0	0	0	0	2	1
1	1	0	0	1	1
2	0	1	1	0	0

1. What if documents are of different length? e.g. 10 words vs. 10,000 words. -> Term frequency normalized by L1 or L2 norm. (tf)
2. What if some terms simply dominate across all documents? -> use inverse document frequency (idf)

Term frequency (tf)

$$tf(t,d) = \frac{f_{t,d}}{\sqrt{\sum_{i \in V} (f_{i,d})^2}}$$

t is the term (token)
d is the document
$f_{t,d}$ is the count of term t in document d
$f_{i,d}$ is the count of term i in document d for all words in the vocabulary V

This is the L2 norm - useful because the magnitude of the tf vector associated with each document is 1 (using L2). L1 norm is common, too (especially when followed with the *idf*.)

There are other ways to define the *tf* as well...

Inverse document frequency (idf)

df = $$\frac{|\text{docs containing t}|}{|\text{docs}|}$$

t is the term (token)
df is the document frequency

idf = $$idf(t,D) = \log\left(\frac{|\text{docs}|}{|1 + \text{docs containing t}|}\right)$$

tf-idf = tf * idf for each token in each document

The log scale is used so terms that occur 10 times more than another are not 10 times more important. The 1 term on the bottom is known as a smoothing constant and is there to ensure that we don’t have a zero in the denominator.

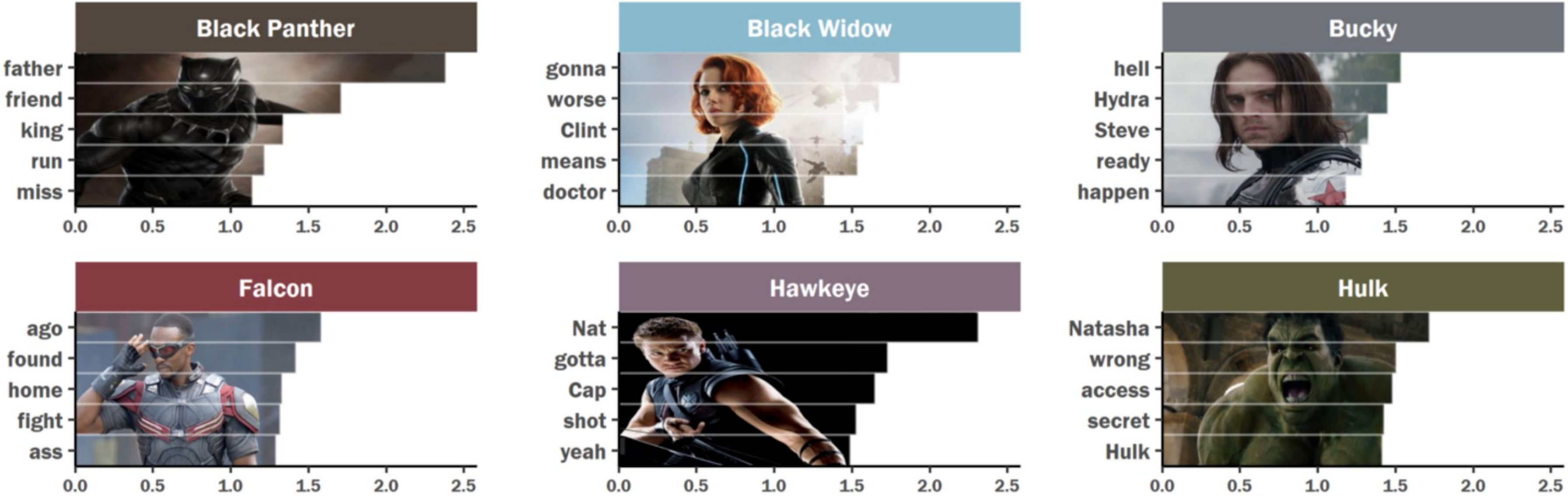
Breakout (~ 10 minutes)

What's the tf-idf of this count matrix?

1. Calculate normalized tf (L2).
2. Calculate the idf for each word.
3. Multiply the tf by the idf to get tf-idf for each word in each document.

Document	guff (0)	jaboo (1)	meet (2)	think (3)	try (4)
0	0	0	0	2	1
1	1	0	0	1	1
2	0	1	1	0	0

Fun application of tf-idf



source

Sklearn example

working_with_text_walktrough.ipynb

Libraries

nltk (good for learning)

spaCy (better for production and your projects)

Reference

Stanford Deep Learning NLP class

Objectives

- Define NLP and describe several use cases
- Explain why NLP is hard
  - Or maybe easy to do (with sklearn) but hard to master
- Describe (and be able to execute) a “typical” text processing workflow
  - Use relevant NLP vocabulary
- Calculate, by hand and in-code, *tf matrices* and *tf-idf matrices*
- tf-idf in Sklearn
- Introduce nltk and spaCy libraries

Appendix - Breakout solution

What's the tf-idf of this count matrix?

Document	guff (0)	jaboo (1)	meet (2)	think (3)	try (4)
0	0	0	0	2	1
1	1	0	0	1	1
2	0	1	1	0	0

1) **tf** (using L2 norm)

Document	guff (0)	jaboo (1)	meet (2)	think (3)	try (4)
0	**0**	**0**	**0**	$2/(2^2+1^2)^{0.5}$ = **0.89**	$1/(2^2+1^2)^{0.5}$ = **0.45**
1	$1/(1^2+1^2+1^2)^{0.5}$ = **0.58**	**0**	**0**	$1/(1^2+1^2+1^2)^{0.5}$ = **0.58**	$1/(1^2+1^2+1^2)^{0.5}$ = **0.58**
2	**0**	$1/(1^2+1^2)^{0.5}$ = **0.71**	$1/(1^2+1^2)^{0.5}$ = **0.71**	**0**	**0**

Appendix - Breakout solution

2) **idf**

Document	guff (0)	jaboo (1)	meet (2)	think (3)	try (4)
0	0	0	0	2	1
1	1	0	0	1	1
2	0	1	1	0	0
	ln(3/2) = **0.41**	ln(3/2) = **0.41**	ln(3/2) = **0.41**	ln(3/3) = **0**	ln(3/3) = **0**

3) **tf-idf** (before another L2 normalization)

Document	guff (0)	jaboo (1)	meet (2)	think (3)	try (4)
0	0 * 0.41 = **0**	0 * 0.41 = **0**	0 * 0.41 = **0**	0 .89 * 0 = **0**	0 .45 * 0 = **0**
1	0.58 * 0.41 = **0.24**	0 * 0.41 = **0**	0 * 0.41 = **0**	0.58 * 0 = **0**	0.58 * 0 = **0**
2	0 * 0.41 = **0**	0.71 * 0.41 = **0.29**	0.71 * 0.41 = **0.29**	0 * 0 = **0**	0 * 0 = **0**