PRACTICAL TOOLS FOR WORKING WITH TEXT

CS/INFO 4300

MARCH 24, 2016

WHERE YOU START

1: DATA MASSAGE

FROM APIS

APIs are awesome!

…but they are rate limited.

A QUICK NOTE ON SCRAPING:

1. Read the privacy policy and robots.txt.
2. Ask permission if it makes sense.
3. Be nice.

Python libraries: urllib2, lxml, requests

For more: http://docs.python-guide.org/en/latest/scenarios/scrape/

EXTRACTING TEXT

Formats:

- CSV
- HTML
- XML
- JSON
- Etc.

EXTRACTING TEXT

Libraries:

- CSV: csv
- HTML: BeautifulSoup
- XML: BeautifulSoup
- JSON: json/simplejson
- Etc. : Have fun.

TOKENS

Use nltk and sklearn. Think about stopwords and text normalization (stemming or lemmatizing).

2: MAKING FEATURES

You now have words.

Now what?

Hint:

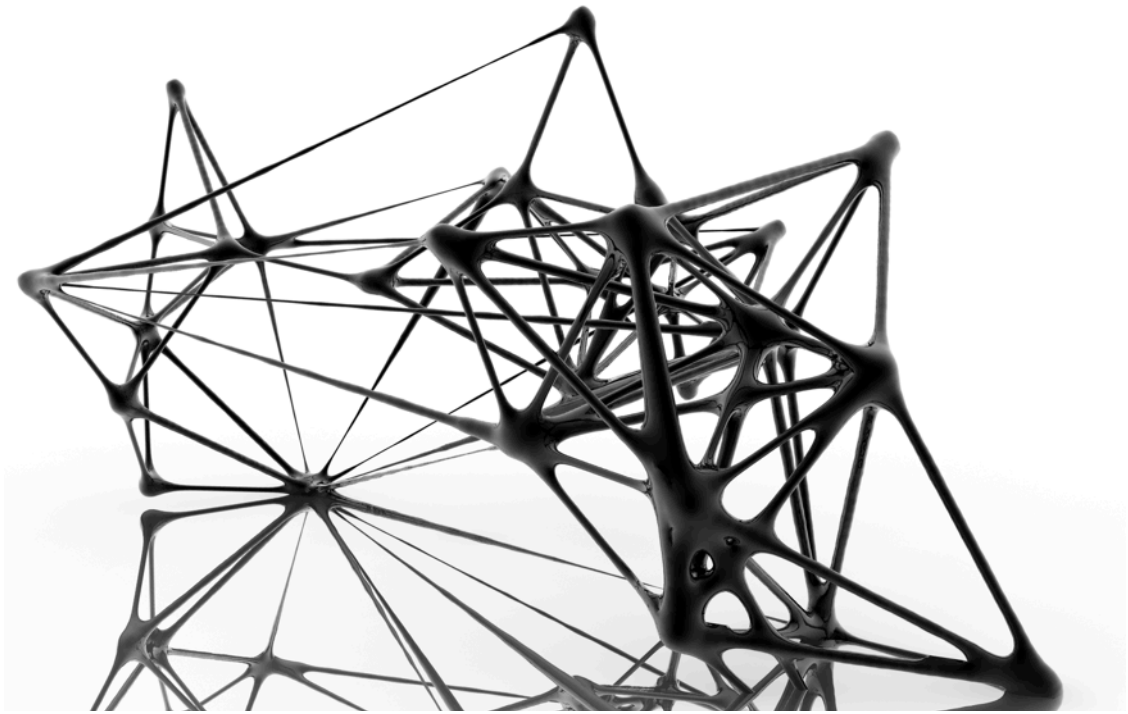
COUNT!

- Words (or TF-IDF)
- N-grams
- Letter n-grams

Caveat: lots of features

STRUCTURAL FEATURES

- Lengths of
  - words
  - sentences
  - utterances
  - documents
- Type/Token ratio
- ???



LEXICONS

For word list *w* and document *d*:

- does *d* contain words from *w*?
- how many?
- what proportion of *d* is words from *w*?

SAMPLE LEXICONS

- nltk.corpus.names – first names by gender
- Subjectivity Lexicon – list of English words and how subjective they are (http://mpqa.cs.pitt.edu/#subj_lexicon)
- Opinion Lexicon – list of 6800 positive and negative English opinions words (https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html#lexicon)
- LIWC – so many lexicons. (http://liwc.wpengine.com/)

And many more…http://sentiment.christopherpotts.net/lexicons.html

	Strength	Length	Word	Part-of-speech	Stemmed	Polarity
1.	type=weaksubj	len=1	word1=abandoned	pos1=adj	stemmed1=n	priorpolarity=negative
2.	type=weaksubj	len=1	word1=abandonment	pos1=noun	stemmed1=n	priorpolarity=negative
3.	type=weaksubj	len=1	word1=abandon	pos1=verb	stemmed1=y	priorpolarity=negative
4.	type=strongsubj	len=1	word1=abase	pos1=verb	stemmed1=y	priorpolarity=negative

SENTIMENT ANALYSIS

Finding the sentiment polarity of a word, sentence, or document (usually based on a lexicon)

HARD HITTERS

Part-of-speech tagging, parsing, sentiment labeling, and named entity recognition:

NLTK

textblob

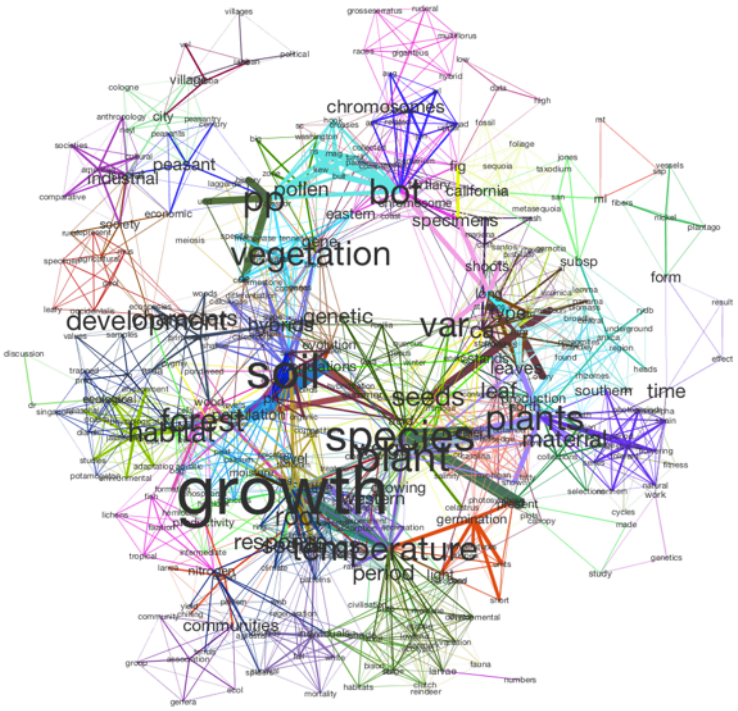
Stanford CoreNLP

An ontology of words:

WordNet

3: SEMANTIC MODELS

This is where it gets weird.



WHAT'S A SEMANTIC MODEL?

semantic: referring to meaning in language or logic.

model: a description or analogy used to help visualize something (as an atom) that cannot be directly observed

WHAT'S A SEMANTIC MODEL?

semantic model: a representation of meaning of words or chunks of text that takes less space than full-scale analysis of meaning.

(we get to ignore structure and parts of speech)

WHERE WE START

Documents →

↓ Words

0	1	0	2	0	0	0	1	5	0	3	1	0	1	0	0	0	1	0	0
0	2	0	3	0	1	1	0	2	1	1	0	2	1	1	0	6	1	2	0
1	3	0	2	0	0	2	0	2	9	0	0	1	0	0	0	1	0	0	1
0	0	0	9	0	1	1	5	0	1	0	0	1	0	2	0	3	0	0	2
1	2	1	2	0	1	0	0	8	0	2	1	0	0	1	0	0	1	1	1

...

⋮

WHAT WE WANT

Representations of words and documents such that

- It’s easy to make them
- We can figure out which words are similar
- Similar documents have “close” representations
- Representations are small

METHOD 1: TOPIC MODELS

LINGUISTS HATE HIM!

This one weird trick will model the meaning in your text without having to keep track of word counts!

THE (GENERATIVE) STORY

You want to write speeches for this guy.

THE (GENERATIVE) STORY

Specifically, you want to write

M speeches $d_1 \ldots d_M$

with N_m words in speech m

about K topics

WHAT'S A TOPIC?

For us: history, foreign policy, economics…

For our model: a probability distribution over words in a vocabulary *V*

e.g.

Topic 1: P("china") = 0.2, P("mexico") = 0.1, P("wall") = 0.1…

Topic 2: P("i") = 0.5, P("great") = 0.1, P("sorry") = 0.0001…

HOW WE WRITE

Before we start writing:

Pick our *K* topics over our vocab *V*

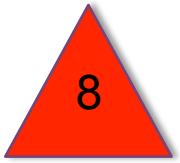
HOW WE WRITE

For each document *m*:

Pick a distribution over topics *K*

For each word in 1…N_m:

Pick a topic:



Pick a word from that topic:

8

→ “huge”

REPEAT!

1: 8 → “huge”

2: 2 → “the”

3: 2 → “the”

4: 12 → “face”

5: 1 → “friends”

6: 19 → “russia”

7: 2 → “has”

…

N_m: 12 → “how”

NOW THE QUESTION:

Which topics would generate the documents you already have?

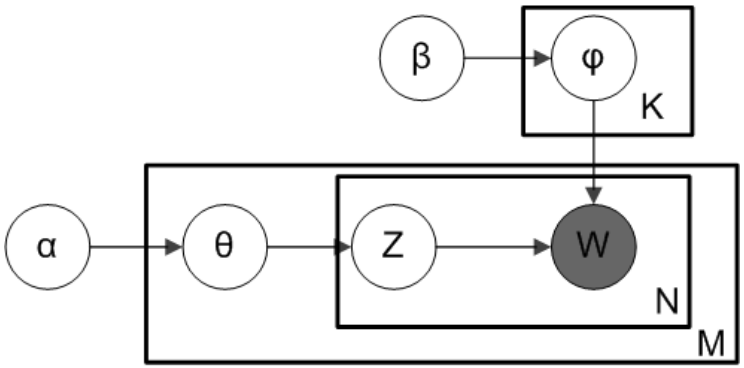
WHICH BRINGS US TO LDA

Latent Dirichlet Allocation (Blei et. al., 2003)

Latent: we can't directly observe topics

Dirichlet: the kind of distribution we get our urns and dice from

Allocation: we allocate words by generating them from document and topic distributions



DEMO

jsLDA

from David Mimno

PROS AND CONS

PROS

Already implemented

Models documents and words

Intuitive dimensions and representation meanings

Easy to-understand output

CONS

Can be slow

Nondeterministic

Awkward on really short documents

WHAT YOU NEED TO KNOW

Python tools exist to do this for you:

- **gensim**
- **lda**

Non-Python tools also exist:

- **Mallet**
- **jsLDA**
- **Stanford TMT**

LDA RECOMMENDATIONS

1. Have lots of decent-sized documents
2. Use 20-100 topics to start, then try other values
3. Remove stop words unless you really need them
4. If you have hyperparameter options, you want
   A. “Asymmetric” alpha (for document distribution over topics)
   B. “Symmetric” beta (for topic distribution over words)
   C. Adaptive hyperparameters if available; ~0.01 alpha, ~0.1 beta if not

METHOD 2: WORD EMBEDDINGS

WHAT ARE WORD EMBEDDINGS?

Maps from words to points in a *k*-dimensional Euclidean space such that

- points close together in space are similar
- vectors can represent analogies:

e.g. *v*(king) – *v*(man) + *v*(woman) = *v*(queen)

POPULAR SYSTEMS

- LSA (1988)
- LDA (2003)
- Turian embeddings (2009)
- SENNA (2010)
- word2vec (2013)
  - Skip-gram
  - CBOW
- GloVe (2014)

PROS AND CONS

PROS

Really good at modeling similarity

Easily clusterable

Pre-trained

Totally trendy

CONS

No obvious document representations

Slow to train

Requires LOTS of data

EMBEDDINGS IN PRACTICE

- Probably just use word2vec skip-gram.
- The original **word2vec** is ugly but fast; **gensim** is cool but slightly slower
- Unless you have O(millions-billions) tokens, just use pretrained embeddings

If you make new embeddings and want to see them:

- www.wordvectors.org
- t-SNE projections: see **scikit-learn**, **tsne**

EXERCISE CAUTION

QUESTIONS?

If you have questions about how to use these things,

PUT THEM ON PIAZZA!