Preprocessing Data

for Topic Models and Recommendation Systems

Allison J.B. Chaney

Preprocessing Data

Topics:

- Tokenizing raw text into terms
- Curating a vocabulary
- Thresholding for recommendation systems and social networks

Tokenizing raw text into terms

For clean text, this is pretty easy:

```
doc = doc.lower() # make everything lower case
doc = re.sub(r'-',' ', doc) # turn hyphens into spaces
doc = re.sub(r'[^a-z ]','', doc) # get rid of all punctuation
doc = re.sub(r' +',' ', doc) # turn multiple spaces into only one
words = doc.split() # split by spaces
```

Hyphens are tricky

- Some people include and some don't. It really depends on your corpus.
- What would you want to do for each of these?

logisitic-normal	mother-in-law	post-Aristotelian	obser-vations
x-ray	pre-eminent	pre-1900	hel-met
mean-field	dis-abled	50-year-old	… go---I will…
non-sequitur	re-cover	20-30 people	two-thirds
camera-ready	co-op	two- or threefold	ex-wife
mid-July	wind-proof	semi-invalid	mayor-elect

External options

- If you want to just let someone else do the leg work:
  - Stanford Tokenizer
  - Apache Open NLP
  - NLTK (python; interface to Stanford + others)
- IBM article The Art of Tokenization compares some of these options (and is generally a good resource)

Messier raw text

- Beautiful Soup for XML

```
from bs4 import BeautifulSoup

# open the document
xml = open(docfilename, 'r')
soup = BeautifulSoup(xml)
xml.close()

# find all the text
fulltext = soup.find("block", {"class":"full_text"})
paras = fulltext.findAll("p")
doc = ' '.join([p.contents[0] for p in paras])
```

Messier raw text

- Misspellings
  - easiest option is to ignore them
  - fix them using a spelling corrector; this might introduce different kinds of problems
- FTFY for encoding issues

Messier raw text

- Named entities
  - Many permutations of the same name: “Joe Smith,” “J. Smith,” “Joseph Smith,” “Joseph F. Smith”, just “Smith.”
  - fuzzywuzzy to help find fuzzy matches
  - Stanford Named Entity Recognizer (available via NLTK)

```
>>> from nltk.tag import StanfordNERTagger
>>> st = StanfordNERTagger('english.all.3class.distsim.crf.ser.gz')
>>> st.tag('Rami Eid is studying at Stony Brook University in NY'.split())
[('Rami', 'PERSON'), ('Eid', 'PERSON'), ('is', 'O'), ('studying', 'O'),
 ('at', 'O'), ('Stony', 'ORGANIZATION'), ('Brook', 'ORGANIZATION'),
 ('University', 'ORGANIZATION'), ('in', 'O'), ('NY', 'LOCATION')]
```

Curating a vocabulary

- exclude short words (generally < ~3 characters)
- minimum # of documents a word must be in
- maximum % of documents a word can be in
- general vocab curation script

Curating a vocabulary

- TF-IDF
  - top N words (e.g., 1000), by TF-IDF
  - pick a threshold manually

$$\text{tfidf}(w) = (\text{total \# of times } w \text{ occurs}) \times \log \frac{\text{total \# of documents}}{\text{\# of docs in which } w \text{ occurs}}$$

Curating a vocabulary

- External stop / common words lists
  - Ranks NL (multiple languages)
  - Word frequency (need to set a threshold)
- Can exclude by part-of-speech (e.g., no adverbs)
  - the NLTK POS tagger is good for this

```
>>> from nltk.tag import pos_tag
>>> from nltk.tokenize import word_tokenize
>>> pos_tag(word_tokenize("John's big idea isn't all that bad."))
[('John', 'NNP'), ("'s", 'POS'), ('big', 'JJ'), ('idea', 'NN'), ('is',
'VBZ'), ("n't", 'RB'), ('all', 'DT'), ('that', 'DT'), ('bad', 'JJ'),
('.', '.')]
```

Curating a vocabulary

- Stemming
  - via NLTK
  - faster but less interpretable
- Lemmatization
  - code snippet (uses NLTK/WordNet)
  - slower, more interpretable

original	**stem**	**lemma**
arguing	argu	argue
taller	tall	tall
better	bet	good
provision	provide	provide
cement	cem	cement
maximum	maxim	maximum

Recommendation data

General thresholds (e.g., 10 items per user)

- Can't threshold *both* by items-per-user and by users-per-item without having one soft threshold

Social Networks

A large network can be both noisy and unwieldy, so sometimes we can speed up inference if we cull it.

Social Recommendation

- Run data through a social network filter to cull some of the connections.
    - Given user-item matrix A, we can compute A * A^T, which gives us a user-user matrix ("social network"). We can take the intersection of that and the observed social network, which gives us a smaller, more relevant network.
- Threshold to only include users who share a % of items in common with their friends (soft threshold)

Social Networks

- threshold by degree (indegree/outdegree)
- exponential random graph models (ERGMs) to test for properties of data
    - e.g., density, centrality, or assortativity
    - Ask yourself: what are the properties of the data and do we think it is suitable for our task?
    - We can try this on different cuts of the data to understand it (e.g., for social recommendation: just scifi books, only jazz music)