Building a Robot Judge: Data Science for the Law

4. N-Gram Models

Elliott Ash

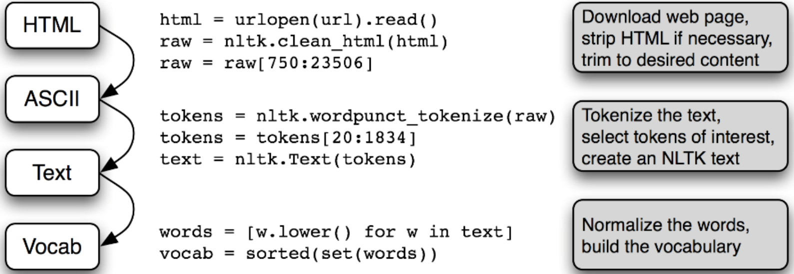
Overview

- These slides describe the process of transforming a corpus into numerical data that can be used in statistical analysis.
- Input:
  - A set of documents (e.g. text files), D.
- Output:
  - A matrix, X, containing statistics about phrase frequencies in those documents.

Goals of Featurization

- To summarize: A major goal of featurization is to produce features that are
  - **predictive** in the learning task
  - **interpretable** by human investigators
  - **tractable** enough to be easy to work with

The NLP Pipeline



Source: NLTK Book, Chapter 3.

Split into paragraphs/sentences

- Many tasks should be done on sentences, rather than corpora as a whole.
  - NLTK and spaCy do a good (but not perfect) job of splitting sentences, while accounting for periods on abbreviations, etc.
  - spaCy is slower but significantly better.
- There isn't a grammar-based paragraph tokenizer.
  - most corpora have new paragraphs annotated.
  - or use line breaks.

Pre-processing

- An important piece of the "art" of text analysis is deciding what data to throw out.
  - Uninformative data add noise and reduce statistical precision.
  - They are also computationally costly.
- Pre-processing choices can affect down-stream results, especially in unsupervised learning tasks (Denny and Spirling 2017).
  - consider trying to interpret your model: compare "judge has" and "has discretion" to "judge has discretion".

Capitalization/Punctuation

God or god?

Let's eat grandpa.
Let's eat, grandpa.

correct punctuation can
save a person`s life.

Source: Chris Bail text data slides.

Word Tokens

- After removing punctuation, getting word tokens is as simple as splitting on white space.

Numbers

1871

1949

1990

- can drop numbers, or replace with special characters; can encode magnitude for example.

Source: Chris Bail text data slides.

Drop Stopwords?

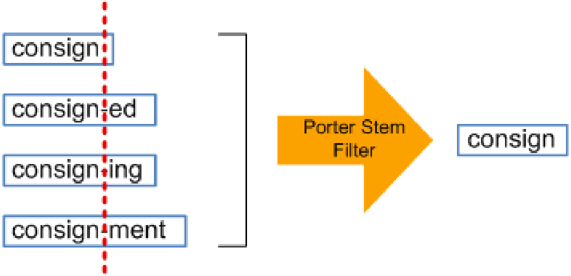
a	an	and	are	as	at	be	by	for	from
has	he	in	is	it	its	of	on	that	the
to	was	were	will	with					

- But legal "memes" often contain stopwords:
  - "beyond a reasonable doubt"
  - "with all deliberate speed"
- An alternative is to filter out words and phrases using part-of-speech tags (later).

Stopwords lists (Kelly et al 2018)

http://www.ranks.nl/stopwords
https://dev.mysql.com/doc/refman/5.1/en/fulltext-stopwords.html
https://code.google.com/p/stop-words/
http://www.lextek.com/manuals/onix/stopwords1.html
http://www.lextek.com/manuals/onix/stopwords2.html
http://analytics101.com/2014/10/all-about-stop-words-for-text-mining.html
http://www.nlm.nih.gov/bsd/disted/pubmedtutorial/020_170.html
https://pypi.python.org/pypi/stop-words
https://msdn.microsof,t.com/zh-cn/library/bb164590
http://www.nltk.org/book/ch02.html

Stemming/lemmatizing



- Porter Stemmer is more aggressive than Snowball Stemmer.
- Lemmatizer produces real words, but N-grams won't make grammatical sense
  - e.g., "judges have been ruling" would become "judge have is rule"

Bag-of-words representation

- Recall the goal of this lecture:
  - Convert a corpus D to a matrix X
- In the "bag-of-words" representation, a row of X is just the frequency distribution over words in the document corresponding to that row.

Counts and frequencies

- **Document counts**: number of documents where a token appears.
- **Term counts**: number of total appearances of a token in corpus.
- **Relative frequency**:

$$\text{Relative Frequency in document } k = \frac{\text{Term count in document } k}{\text{Total tokens in document } k}$$

Building a vocabulary

- An important featurization step is to build a vocabulary of words:
  - Compute document frequencies for all words
  - Inspect low-frequency words and determine a minimum document threshold.
    - e.g., 10 documents, or .25% of documents.
- Can also impose more complex thresholds, e.g.:
  - appears twice in at least 20 documents
  - appears in at least 3 documents in at least 5 years
- Assign numerical identifiers to tokens to increase speed and reduce disk usage.

TF-IDF Weighting

- TF/IDF: "Term-Frequency / Inverse-Document-Frequency."
- The formula for word w in document k:

$$\underbrace{\frac{\text{Count of } w \text{ in } k}{\text{Total word count of } k}}_{\text{Term Frequency}} \times \underbrace{\log(\frac{\text{Number of documents in } D}{\text{Count of documents containing } w})}_{\text{Inverse Document Frequency}}$$

- Example:
  - A document contains 100 words, and the word appears 3 times in the document. The TF is .03. The corpus has 100 documents, and the word appears in 10 documents. the IDF is $\log(100/10) \approx 2.3$, so the TF-IDF for this document is $.03 \times 2.3 = .07$. Say the word appears in 90 out of 100 documents: Then the IDF is 0.105, with TF-IDF for this document equal to .003.
- The formula up-weights relatively rare words that do not appear in all documents.
  - These words are probably more distinctive of topics or differences between documents.

Choosing a vocabulary using TF-IDF

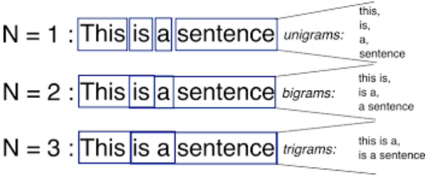
- For each word w, compute average frequency across documents, and compute inverse document frequency.
- Then a word can be "scored" as the product:

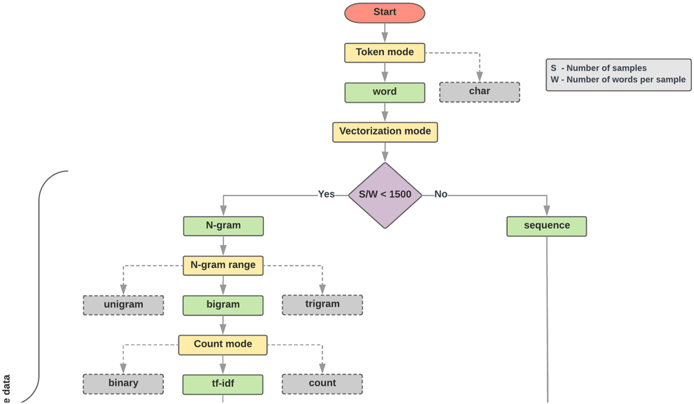
$$\text{average frequency of } w \times \text{IDF of } w$$

- Can rank words or phrases on this metric and choose the top 50,000, for example.
- What counts as a document?

What are N-grams

- N-grams are phrases, sequences of words up to length N.
  - bigrams, trigrams, quadgrams, etc.





- Google Developers recommend **tf-idf-weighted bigrams** as a baseline specification for text classification tasks.
  - ideal for fewer, longer documents.
- With more numerous, shorter documents (rows / doclength > 1500), better to use an embedded sequence.
  - To be described later in the course.

N-grams and high dimensionality

- N-grams will blow up your feature space:
  - filtering out uninformative n-grams is necessary.
- Google Developers say that a feature space with $P = 20,000$ will work well for descriptive and prediction tasks.
  - For supervised learning tasks, a decent baseline is to build a vocabulary of 60K, then use feature selection to get down to 20K.

Collocations

- Conceptually, the goal of including n-grams is to featurize **collocations**:
  - Non-compositional: the meaning is not the sum of the parts (kick+the+bucket≠"kick the bucket")
  - Non-substitutable: cannot substitute components with synonyms ("fast food"≠"quick food")
  - Non-modifiable: cannot modify with additional words or grammar: (e.g., "kick around the bucket", "kick the buckets")

Point-wise mutual information

- A metric for identifying collocations is point-wise mutual information:

$$\mathrm{PMI}(w_1, w_2) = \frac{\Pr(w_1, w_2)}{\Pr(w_1)\Pr(w_2)}$$
$$= \frac{\text{Prob. of collocation, actual}}{\text{Prob. of collocation, if independent}}$$

  where w_1 and w_2 are words in the vocabulary, and w_1, w_2 is the N-gram $w_1\_w_2$.
  - ranks words by how often they collocate, relative to how often they occur apart.

- Warning: Rare words that appear together once or twice will have high PMI.
  - Address this with minimum frequency thresholds.

Geometric Mean: Normalized PMI for $N \geq 2$

- PMI can be generalized to arbitrary N as the geometric mean of the probabilities:

$$\frac{\Pr(w_1, ..., w_N)}{\prod_{i=1}^{n} \sqrt[n]{\Pr(w_i)}}$$

- E.g., for trigrams:

$$\frac{\Pr(w_1, w_2, w_3)}{\sqrt[3]{\Pr(w_1)\Pr(w_2)\Pr(w_3)}}$$

- The *n*-root normalizer is not necessary (it does not change the ranking), but makes scores for bigrams/trigrams/quadgrams/etc. more comparable.

Computing Geometric Mean with N-gram Counts

- Probability of a token is the frequency in the corpus:

$$\Pr(w_1) = \frac{\text{Count}(w_1)}{\sum_{i=1}^{P} \text{Count}(w_i)}$$

  where P is vocabulary size.

- Let $f_i = \text{Count}(w_i)$ and $F = \sum_{i=1}^{P} f_i$. Then we have

$$\text{PMI}(w_1, w_2) = \frac{\Pr(w_1, w_2)}{\Pr(w_1)\Pr(w_2)} = \frac{\frac{f_{12}}{F}}{\frac{f_1}{F} \cdot \frac{f_2}{F}} = \frac{1}{F} \frac{f_{12}}{f_1 f_2}$$

- Note that the leading $\frac{1}{F}$ does not affect the ranking of bigrams, and cancels out with the geometric mean formula:

$$\text{gmean}(w_1, w_2) = \frac{\Pr(w_1, w_2)}{\sqrt{\Pr(w_1)\Pr(w_2)}} = \frac{\frac{f_{12}}{F}}{\sqrt{\frac{f_1}{F} \cdot \frac{f_2}{F}}} = \frac{f_{12}}{\sqrt{f_1 f_2}}$$

- Similarly, it cancels out for $N > 2$.
- Therefore PMI can be computed directly from term counts (rather than frequencies).

Parts of speech tags

- Parts of speech (POS) tags provide useful word categories corresponding to their functions in sentences:
  - Eight main parts of speech: verb (VB), noun (NN), pronoun (PR), adjective (JJ), adverb (RB), determinant (DT), preposition (IN), conjunction (CC).
  - The Penn TreeBank POS tag set (used in many applications) has 36 tags: `https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html`
- Parts of speech vary in their informativeness for various functions:
  - For categorizing topics, nouns are usually most important
  - For sentiment, adjectives are usually most important.

Constructing Legal Memes with POS

- A: Adjective, N: Noun, V: Verb, P: Preposition, D: Determinant, C: Conjunction.
- 2-grams: AN, NN, VN, VV, NV, VP.
  - tax credit, magistrate judge
- 3-grams: NNN, AAN, ANN, NAN, NPN, VAN, VNN, AVN, VVN, VPN, ANV, NVV, VDN, VVV, NNV, VVP, VAV, VVN, NCN, VCV, ACA, PAN.
  - armed and dangerous, stating the obvious
- 4-grams: NCVN, ANNN, NNNN, NPNN, AANN, ANNN, ANPN, NNPN, NPAN, ACAN, NCNN, NNCN, ANCN, NCAN, PDAN, PNPN, VDNN, VDAN, VVDN.
  - `Beyond a reasonable doubt (preposition, article, adjective, noun)`
  - `Earned income tax credit (adjective, noun, noun, noun)`