Words & Regexes

CS 485, Fall 2023
Applications of Natural Language Processing
https://people.cs.umass.edu/~brenocon/cs485_f23/

Brendan O'Connor
College of Information and Computer Sciences
University of Massachusetts Amherst
• Thanks for survey responses!
• Questions
  • Python version 3 (more to come)
  • Midterm: TBA
• Near-term roadmap
Survey bags-of-words

Why are you interested in this class? Why are you interested in NLP?
Survey bags-of-words

What natural languages do you speak or read?

English and pig latin
English, barely Spanish.
• Today: to do NLP you need to get started with text data

1. Text normalization: cleaning up text to reasonable sequences of words (tokens)

2. Regular expressions: a computational tool to help, and even do rule-based NLP
Text normalization

• Every NLP task needs text normalization

  • 1. Segment/tokenize words in running text

  • 2. Normalizing word formats

  • 3. Sentence segmentation and/or paragraphs/sections/chapters/etc.
Example preprocessing pipeline

Raw Text

Text cleaning and term extraction

Unigrams

<table>
<thead>
<tr>
<th>Word</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>candidate</td>
<td>215</td>
</tr>
<tr>
<td>section</td>
<td>158</td>
</tr>
<tr>
<td>Federal</td>
<td>154</td>
</tr>
<tr>
<td>election</td>
<td>140</td>
</tr>
<tr>
<td>committee</td>
<td>120</td>
</tr>
<tr>
<td>under</td>
<td>115</td>
</tr>
<tr>
<td>that</td>
<td>114</td>
</tr>
</tbody>
</table>

Downstream analysis

Text classification

Information extraction

...
Word statistics example
Corpus: news articles from late 1960s

FIGURE 10. Ratio of Term Frequencies in Articles About Protests Coded as Protester Nonviolent or Protester Violent
DISCHARGE CONDITION: The patient was able to oxygenate on room air at 93% at the time of discharge. She was profoundly weak, but was no longer tachycardic and had a normal blood pressure. Her respirations were much improved albeit with transmitted upper airway sounds.

DISCHARGE STATUS: The patient will be discharged to [**Hospital1 **] for both pulmonary and physical rehabilitation.

DISCHARGE MEDICATIONS:
1. Levothyroxine 75 mcg p.o. q.d.
2. Citalopram 10 mg p.o. q.d.
3. Aspirin 81 mg p.o. q.d.
4. Fluticasone 110 mcg two puffs inhaled b.i.d.
5. Salmeterol Diskus one inhalation b.i.d.
6. Acetaminophen 325-650 mg p.o. q.4-6h. prn.
DISCHARGE CONDITION: The patient was able to oxygenate on room air at 93% at the time of discharge. She was profoundly weak, but was no longer tachycardic and had a normal blood pressure. Her respirations were much improved albeit with transmitted upper airway sounds.

DISCHARGE STATUS: The patient will be discharged to [**Hospital1**] for both pulmonary and physical rehabilitation.

DISCHARGE MEDICATIONS:
1. Levothyroxine 75 mcg p.o. q.d.
2. Citalopram 10 mg p.o. q.d.
3. Aspirin 81 mg p.o. q.d.
4. Fluticasone 110 mcg two puffs inhaled b.i.d.
5. Salmeterol Diskus one inhalation b.i.d.
6. Acetaminophen 325-650 mg p.o. q.4-6h. prn.
DISCHARGE CONDITION: The patient was able to oxygenate on room air at 93% at the time of discharge. She was profoundly weak, but was no longer tachycardic and had a normal blood pressure. Her respirations were much improved albeit with transmitted upper airway sounds.

DISCHARGE STATUS: The patient will be discharged to [**Hospital1 **] for both pulmonary and physical rehabilitation.

DISCHARGE MEDICATIONS:
1. Levothyroxine 75 mcg p.o. q.d.
2. Citalopram 10 mg p.o. q.d.
3. Aspirin 81 mg p.o. q.d.
4. Fluticasone 110 mcg two puffs inhaled b.i.d.
5. Salmeterol Diskus one inhalation b.i.d.
6. Acetaminophen 325-650 mg p.o. q.4-6h. prn.

Text data
(MIMIC III EHR)
1. Easy to structure: write hard-coded, custom string processor
2. Harder: develop more complex processor
3. Hardest: full natural language

Semi-structured, regular ordering
MEDICINE_NAME NUMBER UNITS MODIFIERS

(1) and (2): regular expressions are often useful!
Regular expressions

• A formal language for specifying text strings
• How can we search for any of these?
  • woodchuck
  • woodchucks
  • Woodchuck
  • Woodchucks
Regular Expressions: Disjunctions

• Letters inside square brackets []

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>[wW]oodchuck</td>
<td>Woodchuck, woodchuck</td>
</tr>
<tr>
<td>[1234567890]</td>
<td>Any digit</td>
</tr>
</tbody>
</table>

• Ranges [A–Z]

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>[A–Z]</td>
<td>An upper case letter</td>
</tr>
<tr>
<td></td>
<td>Drenched Blossoms</td>
</tr>
<tr>
<td>[a–z]</td>
<td>A lower case letter</td>
</tr>
<tr>
<td></td>
<td>my beans were impatient</td>
</tr>
<tr>
<td>[0–9]</td>
<td>A single digit</td>
</tr>
<tr>
<td></td>
<td>Chapter 1: Down the Rabbit Hole</td>
</tr>
</tbody>
</table>
Regular Expressions: Negation in Disjunction

- Negations \[^Ss\]
  - Carat means negation only when first in []

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Matches</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>[^A-Z]</td>
<td>Not an upper case letter</td>
<td>Oyfn pripeitchik</td>
</tr>
<tr>
<td>^Ss</td>
<td>Neither ‘S’ nor ‘s’</td>
<td>I have no exquisite reason”</td>
</tr>
<tr>
<td>^e^</td>
<td>Neither e nor ^</td>
<td>Look here</td>
</tr>
<tr>
<td>a^b</td>
<td>The pattern a carat b</td>
<td>Look up a^b now</td>
</tr>
</tbody>
</table>
Regular Expressions: More Disjunction

- Woodchucks is another name for groundhog!
- The pipe | for disjunction

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>groundhog</td>
<td>woodchuck</td>
</tr>
<tr>
<td>yours</td>
<td>mine</td>
</tr>
<tr>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>[gG]roundhog</td>
<td>[Ww]oodchuck</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>colou?r</td>
<td>Optional previous char</td>
</tr>
<tr>
<td></td>
<td>color, colour</td>
</tr>
<tr>
<td>oo*h!</td>
<td>0 or more of previous char</td>
</tr>
<tr>
<td></td>
<td>oh!, ooh!, oooh!, ooooh!</td>
</tr>
<tr>
<td>o+h!</td>
<td>1 or more of previous char</td>
</tr>
<tr>
<td></td>
<td>oh!, ooh!, oooh!, ooooh!</td>
</tr>
<tr>
<td>baa+</td>
<td></td>
</tr>
<tr>
<td></td>
<td>baa, baaa, baaaa, baaaaa</td>
</tr>
<tr>
<td>beg.n</td>
<td></td>
</tr>
<tr>
<td></td>
<td>begin, begun, begun, begun, begun, beg3n</td>
</tr>
</tbody>
</table>

Stephen C Kleene
Kleene *, Kleene +
Demo: regexes in grep
grep -Po '#[^ ]*livesmatter'
grep -Po '#[a-z0-9]*livesmatter'
Example

• Find me all instances of the word “the” in a text.

  the

  Misses capitalized examples

  \[ tT \]he

  Incorrectly returns other or theology

  \[ ^a-zA-Z][tT]he[^a-zA-Z] \]

Errors

• The process we just went through was based on fixing two kinds of errors

  • Matching strings that we should not have matched (there, then, other)
    • False positives (Type I)
  • Not matching things that we should have matched (The)
    • False negatives (Type II)
Errors cont.

• In NLP we are always dealing with these kinds of errors.

• Reducing the error rate for an application often involves two antagonistic efforts:
  • Increasing accuracy or precision (minimizing false positives)
  • Increasing coverage or recall (minimizing false negatives).
Simple Tokenization in UNIX

• (Inspired by Ken Church’s UNIX for Poets.)
• Given a text file, output the word tokens and their frequencies
  
  \texttt{tr -sc 'A-Za-z' ' \n' < shakes.txt} \hspace{1cm} \text{Change all non-alpha to newlines}
  \hspace{0.5cm} | \hspace{1cm} \texttt{sort} \hspace{1cm} \text{Sort in alphabetical order}
  \hspace{0.5cm} | \hspace{1cm} \texttt{uniq -c} \hspace{1cm} \text{Merge and count each type}

1945 A 25 Aaron
72 AARON 6 Abate
19 ABBESS 1 Abates
5 ABBOT 5 Abbess
... ... 6 Abbey
3 Abbot
... ...
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?
- m.p.h., PhD. → ??
Tokenization: language issues

- French
  - *L'ensemble* → one token or two?
    - Want *l’ensemble* to match with *un ensemble*

- German noun compounds are not segmented
  - *Lebensversicherungsgesellschaftsangestellter*
  - ‘life insurance company employee’
  - German information retrieval needs *compound splitter*
Summary

- Regular expressions play a surprisingly large role
  - Sophisticated sequences of regular expressions are often the first model for any text processing text
- For many hard tasks, we use machine learning classifiers
  - But regular expressions are used as features in the classifiers
  - Can be very useful in capturing generalizations
The patient was able to oxygenate on room air at 93% at the time of discharge. She was profoundly weak, but was no longer tachycardic and had a normal blood pressure. Her respirations were much improved albeit with transmitted upper airway sounds.

DISCHARGE STATUS: The patient will be discharged to [**Hospital1 **] for both pulmonary and physical rehabilitation.

DISCHARGE MEDICATIONS:
1. Levothyroxine 75 mcg p.o. q.d.
2. Citalopram 10 mg p.o. q.d.
3. Aspirin 81 mg p.o. q.d.
4. Fluticasone 110 mcg two puffs inhaled b.i.d.
5. Salmeterol Diskus one inhalation b.i.d.
6. Acetaminophen 325-650 mg p.o. q.4-6h. prn.

The patient was able to oxygenate on room air at 93% at the time of discharge. She was profoundly weak, but was no longer tachycardic and had a normal blood pressure. Her respirations were much improved albeit with transmitted upper airway sounds.

**This step is usually specific to your dataset**
The patient was able to oxygenate on room air at 93% at the time of discharge. She was profoundly weak, but was no longer tachycardic and had a normal blood pressure. Her respirations were much improved albeit with transmitted upper airway sounds.

There are good off-the-shelf tokenizers (NLTK, SpaCy, CoreNLP, Twokenizer)

- Words are (usually) the basic units of analysis in NLP.
- In English, words are delineated as tokens via space and punctuation conventions, recognizable via moderately simple rules.
- Tokenization: from text string to sequence of word strings.
- Sentence splitting: harder but sometimes done too.
Preprocessing: Normalization

• Often:
  • Lowercase words ("She" -> "she")

• Sometimes:
  • Remove numbers ("93" -> "NUMBER_NN")
  • Correct misspellings / alternate spellings ("color" -> "colour")

• Problem specific:
  • Resolve synonyms / aliases (if you know them already)
  • Remove “stopwords”
    • Punctuation and grammatical function words ("if", "the", "by"), and
    • Very common words in your domain that don’t add much meaning
How many words?

\[ N = \text{number of tokens} \]
\[ V = \text{vocabulary} = \text{set of types} \]
\[ |V| \text{ is the size of the vocabulary} \]

Church and Gale (1990): \[ |V| > O(N^{\frac{1}{2}}) \]

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

<table>
<thead>
<tr>
<th>Word</th>
<th>Frequency (f)</th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td>1629</td>
</tr>
<tr>
<td>and</td>
<td>844</td>
</tr>
<tr>
<td>to</td>
<td>721</td>
</tr>
<tr>
<td>a</td>
<td>627</td>
</tr>
<tr>
<td>she</td>
<td>537</td>
</tr>
<tr>
<td>it</td>
<td>526</td>
</tr>
<tr>
<td>of</td>
<td>508</td>
</tr>
<tr>
<td>said</td>
<td>462</td>
</tr>
<tr>
<td>i</td>
<td>400</td>
</tr>
<tr>
<td>alice</td>
<td>385</td>
</tr>
</tbody>
</table>

Alice’s Adventures in Wonderland, by Lewis Carroll
Zipf’s Law

- When word types are ranked by frequency, then \textit{frequency (f) * rank (r)} is roughly equal to some \textit{constant (k)}

\[ f \times r = k \]
<table>
<thead>
<tr>
<th>Rank ($r$)</th>
<th>Word</th>
<th>Frequency ($f$)</th>
<th>$r \cdot f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>the</td>
<td>1629</td>
<td>1629</td>
</tr>
<tr>
<td>2</td>
<td>and</td>
<td>844</td>
<td>1688</td>
</tr>
<tr>
<td>3</td>
<td>to</td>
<td>721</td>
<td>2163</td>
</tr>
<tr>
<td>4</td>
<td>a</td>
<td>627</td>
<td>2508</td>
</tr>
<tr>
<td>5</td>
<td>she</td>
<td>537</td>
<td>2685</td>
</tr>
<tr>
<td>6</td>
<td>it</td>
<td>526</td>
<td>3156</td>
</tr>
<tr>
<td>7</td>
<td>of</td>
<td>508</td>
<td>3556</td>
</tr>
<tr>
<td>8</td>
<td>said</td>
<td>462</td>
<td>3696</td>
</tr>
<tr>
<td>9</td>
<td>i</td>
<td>400</td>
<td>3600</td>
</tr>
<tr>
<td>10</td>
<td>alice</td>
<td>385</td>
<td>3850</td>
</tr>
<tr>
<td>20</td>
<td>all</td>
<td>179</td>
<td>3580</td>
</tr>
<tr>
<td>30</td>
<td>little</td>
<td>128</td>
<td>3840</td>
</tr>
<tr>
<td>40</td>
<td>about</td>
<td>94</td>
<td>3760</td>
</tr>
<tr>
<td>50</td>
<td>again</td>
<td>82</td>
<td>4100</td>
</tr>
<tr>
<td>60</td>
<td>queen</td>
<td>68</td>
<td>4080</td>
</tr>
<tr>
<td>70</td>
<td>don’t</td>
<td>60</td>
<td>4200</td>
</tr>
<tr>
<td>80</td>
<td>quite</td>
<td>55</td>
<td>4400</td>
</tr>
<tr>
<td>90</td>
<td>just</td>
<td>51</td>
<td>4590</td>
</tr>
<tr>
<td>100</td>
<td>voice</td>
<td>47</td>
<td>4700</td>
</tr>
<tr>
<td>200</td>
<td>hand</td>
<td>20</td>
<td>4000</td>
</tr>
<tr>
<td>300</td>
<td>turning</td>
<td>12</td>
<td>3600</td>
</tr>
<tr>
<td>400</td>
<td>hall</td>
<td>9</td>
<td>3600</td>
</tr>
<tr>
<td>500</td>
<td>kind</td>
<td>7</td>
<td>3500</td>
</tr>
</tbody>
</table>

Plot: log frequencies
recall: $f^* = k \log f + \log r = \log k$
Plot: log frequencies

Recall:
\[ f^* r = k \]
\[ \log f + \log r = \log k \]
Normalization

• Need to “normalize” terms
  • Information Retrieval: indexed text & query terms must have same form.
    • We want to match *U.S.A.* and *USA*

• We implicitly define equivalence classes of terms
  • e.g., deleting periods in a term
Case folding

- Applications like IR: reduce all letters to lower case
  - Since users tend to use lower case
  - Possible exception: upper case in mid-sentence?
    - e.g., *General Motors*
    - *Fed* vs. *fed*
    - *SAIL* vs. *sail*
- For sentiment analysis, MT, Information extraction
  - Case is helpful (*US* versus *us* is important)
Lemmatization

• Reduce inflections or variant forms to base form
  • *am, are, is* $\rightarrow$ *be*
  • *car, cars, car's, cars'* $\rightarrow$ *car*
• *the boy's cars are different colors* $\rightarrow$ *the boy car be different color*
• Lemmatization: have to find correct dictionary headword form
• Machine translation
  • Spanish *quiero* (‘I want’), *quieres* (‘you want’) same lemma as *querer* ‘want’
Morphology

• **Morphemes:**
  • The small meaningful units that make up words
  • **Stems:** The core meaning-bearing units
  • **Affixes:** Bits and pieces that adhere to stems
    • Often with grammatical functions
Stemming

- Reduce terms to their stems in information retrieval
- *Stemming* is crude chopping of affixes
  - language dependent
  - e.g., *automate(s), automatic, automation* all reduced to *automat*.
Porter’s algorithm
The 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*

Consider the IR query matching problem. What are the precision/recall tradeoffs of the Porter stemmer?