n-grams and basic natural language processing
text data analysis

• Written text is often treated as a form of data for analysis

• Some types of analyses:
  • Measuring similarity between documents
  • Extracting topics from documents
  • Finding the most frequently occurring words
  • Quantifying the importance of phrases

• Most of these involve breaking up documents into words or “n-grams”
n-grams

- **n-grams** break up a sentence into overlapping **subsequences** of length \( n \).
  - \( n \) typically refers to words or characters (though it could be e.g., syllables).
  - Unigrams (\( n=1 \)), bigrams (\( n=2 \)), trigrams (\( n=3 \)), …

Consider the sentence “I saw a cat”

- Word-based 3-grams:
  - “I saw a”, “saw a cat”

- Character-based 3-grams:
bag-of-words

- The same $n$-gram can appear multiple times in a string
  - This indicates a higher frequency
  - Generally we only care about order within an $n$-gram, not between $n$-grams

- **Bag-of-words** model: Order between words (more generally, $n$-grams) in a document is not considered

- “wan can cup”

  wan : 1  an__ : 2  n__c : 2
  _ca : 1  can : 1  _cu : 1
  cup : 1

- Bag-of-words vector:

<table>
<thead>
<tr>
<th>Raw Text</th>
<th>Bag-of-words vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>it</td>
<td>2</td>
</tr>
<tr>
<td>they</td>
<td>0</td>
</tr>
<tr>
<td>puppy</td>
<td>1</td>
</tr>
<tr>
<td>and</td>
<td>1</td>
</tr>
<tr>
<td>cat</td>
<td>0</td>
</tr>
<tr>
<td>aardvark</td>
<td>0</td>
</tr>
<tr>
<td>cute</td>
<td>1</td>
</tr>
<tr>
<td>extremely</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
histograms of n-grams

• Can make a histogram of $n$-grams!

• Treat each $n$-gram in a document as a separate (categorical) bucket

• Different documents have different $n$-gram frequencies

• **But:** Documents from the same language have similar histograms!

  • Can use this for language classification
histograms of n-grams

• Can make a histogram of $n$-grams!

• Treat each $n$-gram in a document as a separate (categorical) bucket

• Different documents have different $n$-gram frequencies

• **But**: Documents from the same language have similar histograms!

• Can use this for language classification
histograms of n-grams

• Can make a histogram of $n$-grams!

• Treat each $n$-gram in a document as a separate (categorical) bucket

• Different documents have different $n$-gram frequencies

• **But:** Documents from the same language have similar histograms!

• Can use this for language classification
histograms of n-grams

• Can make a histogram of $n$-grams!

• Treat each $n$-gram in a document as a separate (categorical) bucket

• Different documents have different $n$-gram frequencies

• But: Documents from the same language have similar histograms!

• Can use this for language classification
n-gram importance

• How do we quantify the *importance* of an n-gram in a document?

• One possibility: Count its frequency
  • More frequently occurring should be more important
  • But what about common words like “a”, “as”, “is”, …?
  • These specific examples are *stopwords* (which we should probably remove from analysis anyway), but many cases are not stopwords

• Need to somehow measure how “unique” the n-gram is across documents
One algorithm which quantifies this intuition is **term frequency-inverse document frequency**, or **tf-idf**

- One of the most popular schemes used today

- Letting $t$ be a term (n-gram), $d$ be a document, and $D$ be a **corpus** (collection) of documents under consideration, the tf-idf score of $t$ in $d$ with respect to $D$ is

  $$ \text{tfidf}(t, d, D) = \text{tf}(t, d) \cdot \text{idf}(t, D) $$

- Many different methods for quantifying $\text{tf}$ and $\text{idf}$
tf-idf

- Term frequency $tf(t, d)$: Typically the fraction of terms in $d$ which are $t$
  - Letting $f_{t,d}$ be the number of occurrences of $t$ in $d$, it is
    $$ tf(t, d) = \frac{f_{t,d}}{\sum_{t'} f_{t',d}} $$

- Inverse document frequency $idf(t, D)$: A measure of how rare $t$ is across documents (i.e., how much information it provides)
  - Letting $N = |D|$ be the size of the corpus and $n_t$ be the number of documents where $t$ occurs, it is typically quantified as
    $$ idf(t, D) = \log_{10} \frac{N}{n_t} $$
    Why log?

<table>
<thead>
<tr>
<th>Word</th>
<th>TF</th>
<th>IDF</th>
<th>TF*IDF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>The</td>
<td>1/7</td>
<td>1/7</td>
<td>log(2/2) = 0</td>
</tr>
<tr>
<td>Car</td>
<td>1/7</td>
<td>0</td>
<td>log(2/1) = 0.3</td>
</tr>
<tr>
<td>Truck</td>
<td>0</td>
<td>1/7</td>
<td>log(2/1) = 0.3</td>
</tr>
<tr>
<td>Is</td>
<td>1/7</td>
<td>1/7</td>
<td>log(2/2) = 0</td>
</tr>
<tr>
<td>Driven</td>
<td>1/7</td>
<td>1/7</td>
<td>log(2/2) = 0</td>
</tr>
<tr>
<td>On</td>
<td>1/7</td>
<td>1/7</td>
<td>log(2/2) = 0</td>
</tr>
<tr>
<td>The</td>
<td>1/7</td>
<td>1/7</td>
<td>log(2/2) = 0</td>
</tr>
<tr>
<td>Road</td>
<td>1/7</td>
<td>0</td>
<td>log(2/1) = 0.3</td>
</tr>
<tr>
<td>Highway</td>
<td>0</td>
<td>1/7</td>
<td>log(2/1) = 0.3</td>
</tr>
</tbody>
</table>
Consider the following four documents: “The sky is blue.”, “The sun is bright today.”, “The sun in the sky is bright.”, “We can see the shining sun, the bright sun.”

Documents are typically much longer than single sentences.

TF: Find doc-word matrix, normalize rows to sum to 1.

\[
f_{t,d}
\]

\[
\text{tf}(t, d)
\]
example

• IDF: Find number of documents each word occurs in, compute formula

\[ f_{t,d} \]

\[
\begin{array}{cccccccc}
\text{blue} & \text{bright} & \text{can} & \text{see} & \text{shining} & \text{sky} & \text{sun} & \text{today} \\
1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
2 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\
3 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\
4 & 0 & 1 & 1 & 1 & 1 & 0 & 2 & 0 \\
\end{array}
\]

\[ \text{idf}(t,D) \]

\[
\begin{array}{cccccccc}
\text{blue} & \text{bright} & \text{can} & \text{see} & \text{shining} & \text{sky} & \text{sun} & \text{today} \\
0.602 & 0.125 & 0.602 & 0.602 & 0.602 & 0.301 & 0.125 & 0.602 \\
\end{array}
\]

\[
\log_{10} \frac{4}{1} = 0.602
\]

• TF-IDF: Multiply TF and IDF scores, use to rank importance of words within documents

\[
\begin{array}{cccccccc}
\text{blue} & \text{bright} & \text{can} & \text{see} & \text{shining} & \text{sky} & \text{sun} & \text{today} \\
1 & 0.301 & 0 & 0 & 0 & 0 & 0.151 & 0 & 0 \\
2 & 0 & 0.0417 & 0 & 0 & 0 & 0 & 0.0417 & 0.201 \\
3 & 0 & 0.0417 & 0 & 0 & 0 & 0 & 0.100 & 0.0417 & 0 \\
4 & 0 & 0.0209 & 0.100 & 0.100 & 0.100 & 0 & 0 & 0.0417 & 0 \\
\end{array}
\]
text preprocessing

• Typically apply a series of preprocessing steps to each document prior to analysis

  • Mostly using Python’s nltk library

• Tokenization

  • Remove non-word characters (e.g., html tags, punctuation)

  • Break text into tokens, e.g., n-grams (text.split())

• Stopword removal

  • Remove common word tokens, e.g., “a, I, as, …” that are not useful for analysis

  • from nltk.corpus import stopwords

    sr = stopwords.words('english')
text preprocessing

• Stemming / Lemmatizing

• Stemming reduces inflected words to their word stem (e.g., studies, studying -> studi)

• Lemmatization maps words to their dictionary form, representing them as words (e.g., studies, studying —> study)

• Lemmatization is more complex, but typically preferred

• from nltk.stem import PorterStemmer, WordNetLemmatizer

stemmer = PorterStemmer()

lemmatized = WordNetLemmatizer()
natural language processing

- What we have been studying are specific methods in natural language processing, or NLP

- NLP is concerned with how to automatically analyze large corpuses of text

- Two main classes of NLP: rules-based and statistical
  - tf-idf is a simple (yet widely used) statistical technique
  - Today’s innovations are largely in the statistical category, leveraging machine learning

- Key is building knowledge representations
natural language processing

• Some common functions of NLP

• **Machine translation**: Translating between languages (e.g., Google translate)

• **Speech recognition**: Determine the textual representation of an audio track (e.g., Siri)

• **Document summarization**: Determine an effective summary of a document (e.g., Watson)

• All of these are constantly being innovated with new NLP algorithms