

Text Data in Business and Economics

Basel University - Autumn 2024

3. Tokenization

Tokenization: Overview

Pre-Processing Text

Counts and Frequencies

N-Grams

Parts of Speech

Tokenization: Introduction

- ▶ Input:
 - ▶ A set of documents (e.g. text files), D

Tokenization: Introduction

- ▶ Input:
 - ▶ A set of documents (e.g. text files), D
- ▶ Output (tokens):
 - ▶ A sequence, W_i , containing a list of tokens in document i – words, word pieces, or n-grams for use in natural language processing
 - ▶ A document-term matrix, X , containing statistics about word/phrase frequencies in those documents

Goals of Tokenization

To summarize: A major goal of tokenization is to produce features that are

- ▶ **predictive** in the learning task
- ▶ **interpretable** by human investigators
- ▶ **tractable** enough to be easy to work with

Goals of Tokenization

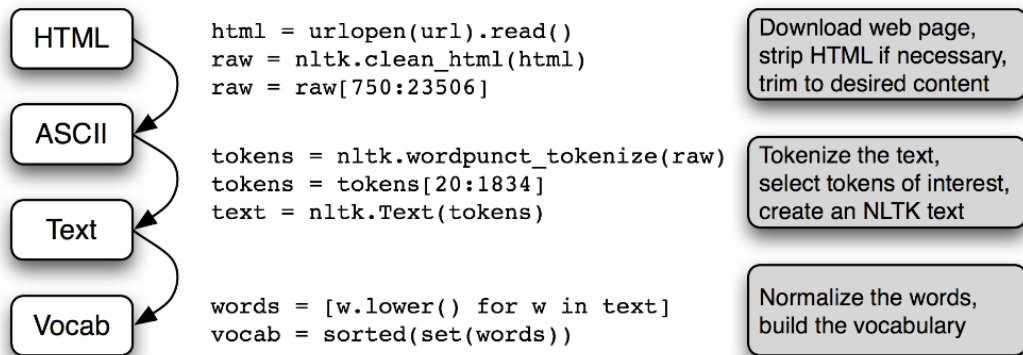
To summarize: A major goal of tokenization is to produce features that are

- ▶ **predictive** in the learning task
- ▶ **interpretable** by human investigators
- ▶ **tractable** enough to be easy to work with

Two broad approaches:

1. convert documents to vectors, usually frequency distributions over pre-processed n-grams
2. convert documents to sequences of tokens, for inputs to sequential models

A Standard Tokenization Pipeline



Source: NLTK Book, Chapter 3.

Tokenization: Overview

Pre-Processing Text

Counts and Frequencies

N-Grams

Parts of Speech

Segmenting paragraphs/sentences

- ▶ Many tasks should be done on sentences, rather than corpora as a whole
 - ▶ spaCy does a good (but not perfect) job of splitting sentences, while accounting for periods on abbreviations, etc.
- ▶ 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)
 - ▶ some features are more interpretable: “judge has” / “has discretion” vs “judge has discretion”

Capitalization

- ▶ Removing capitalization is a standard corpus normalization technique
 - ▶ usually the capitalized/non-capitalized version of a word are equivalent – e.g. words showing up capitalized at beginning of sentence
 - ▶ → capitalization not informative

Capitalization

- ▶ Removing capitalization is a standard corpus normalization technique
 - ▶ usually the capitalized/non-capitalized version of a word are equivalent – e.g. words showing up capitalized at beginning of sentence
 - ▶ → capitalization not informative
- ▶ Also: what about “the first amendment” versus “the First Amendment”?
 - ▶ Compromise: include capitalized version of words not at beginning of sentence

Capitalization

- ▶ Removing capitalization is a standard corpus normalization technique
 - ▶ usually the capitalized/non-capitalized version of a word are equivalent – e.g. words showing up capitalized at beginning of sentence
 - ▶ → capitalization not informative
- ▶ Also: what about “the first amendment” versus “the First Amendment”?
 - ▶ Compromise: include capitalized version of words not at beginning of sentence
- ▶ For some tasks, capitalization is important
 - ▶ needed for sentence splitting, part-of-speech tagging, named entity recognition, syntactic/semantic parsing

Capitalization

- ▶ Removing capitalization is a standard corpus normalization technique
 - ▶ usually the capitalized/non-capitalized version of a word are equivalent – e.g. words showing up capitalized at beginning of sentence
 - ▶ → capitalization not informative
- ▶ Also: what about “the first amendment” versus “the First Amendment”?
 - ▶ Compromise: include capitalized version of words not at beginning of sentence
- ▶ For some tasks, capitalization is important
 - ▶ needed for sentence splitting, part-of-speech tagging, named entity recognition, syntactic/semantic parsing
 - ▶ For sequence data, e.g. language modeling. To generate believable text, need to keep everything

Punctuation

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

**correct punctuation can
save a person`s life.**

Source: Chris Bail text data slides.

Inclusion of punctuation depends on your task:

Punctuation

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

**correct punctuation can
save a person`s life.**

Source: Chris Bail text data slides.

Inclusion of punctuation depends on your task:

- ▶ if you are vectorizing the document as a bag of words or bag of n-grams, punctuation won't be needed

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

**correct punctuation can
save a person`s life.**

Source: Chris Bail text data slides.

Inclusion of punctuation depends on your task:

- ▶ if you are vectorizing the document as a bag of words or bag of n-grams, punctuation won't be needed
- ▶ like capitalization, punctuation is needed for annotations (sentence splitting, parts of speech, syntax, roles, etc) or for text generators

Numbers

- ▶ for bag of words/phrases:
 - ▶ drop numbers, or replace with a special character (e.g. #)
- ▶ for language models:
 - ▶ just treat them like letters

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 | | | | | |

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

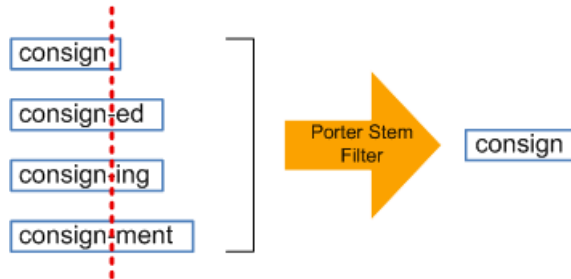
- ▶ What about “not guilty”?
- ▶ Legal “memes” often contain stopwords:
 - ▶ “beyond a reasonable doubt”
 - ▶ “with all deliberate speed”

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

- ▶ What about “not guilty”?
- ▶ Legal “memes” often contain stopwords:
 - ▶ “beyond a reasonable doubt”
 - ▶ “with all deliberate speed”
- ▶ can drop stopwords by themselves, but keep them as part of phrases
- ▶ can filter out words and phrases using part-of-speech tags (later)

Stemming/lemmatizing



- ▶ Effective dimension reduction with little loss of information
- ▶ Lemmatizer produces real words, but N-grams won't make grammatical sense
 - ▶ e.g., "judges have been ruling" would become "judge have be rule"

Tokenization: Overview

Pre-Processing Text

Counts and Frequencies

N-Grams

Parts of Speech

Tokens

The most basic unit of representation in a text

- ▶ characters: documents as sequence of individual letters $\{h, e, l, l, o, , w, o, r, l, d\}$

Tokens

The most basic unit of representation in a text

- ▶ characters: documents as sequence of individual letters {h,e,l,l,o, ,w,o,r,l,d}
- ▶ words: split on white space {hello, world}

Tokens

The most basic unit of representation in a text

- ▶ characters: documents as sequence of individual letters {h,e,l,l,o, ,w,o,r,l,d}
- ▶ words: split on white space {hello, world}
- ▶ n-grams: learn a vocabulary of phrases and tokenize those: “Basel University → basel_university”

Bag-of-words representation

Say we want to 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
- ▶ more generally, “bag of terms” representation refers to counts over any informative features – e.g. n-grams, syntax features, etc.

Counts and frequencies

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

$$\text{Term Frequency of } w \text{ in document } d = \frac{\text{Count of } w \text{ in document } d}{\text{Total tokens in document } d}$$

Application: Ranking Partisan language

Monroe et al (2009), "Fightin' Words"

- ▶ This paper systematically explores a number of methods for identifying words that are distinctive of groups of speakers
 - ▶ in this case, whether U.S. congressmen are Republicans or Democrats

Application: Ranking Partisan language

Monroe et al (2009), "Fightin' Words"

- ▶ This paper systematically explores a number of methods for identifying words that are distinctive of groups of speakers
 - ▶ in this case, whether U.S. congressmen are Republicans or Democrats
- ▶ First, they separate speeches by topic using latent dirichlet allocation (next lectures)
 - ▶ they then test a number of methods for ranking partisanship of words

Relative Frequency of Words

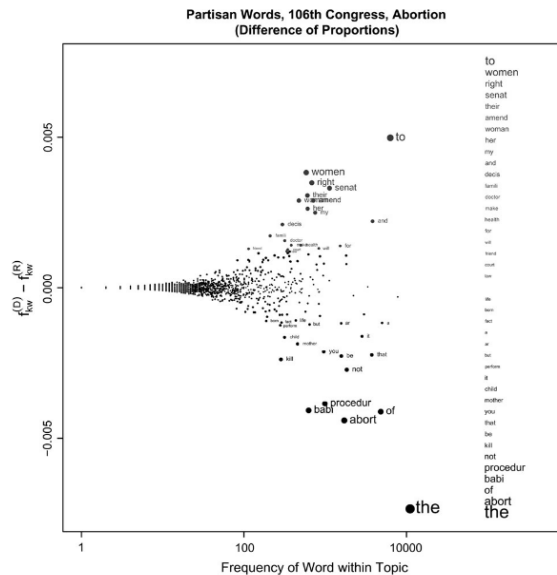


Fig. 1 Feature evaluation and selection using $f_{kw}^{(D)} - f_{kw}^{(R)}$. Plot size is proportional to evaluation weight, $|f_{kw}^{(D)} - f_{kw}^{(R)}|$. The top 20 Democratic and Republican words are labeled and listed in rank order to the right. The results are almost identical for two other measures discussed in the text: unlogged $tf.idf$ and frequency-weighted WordScores.

Questions

- ▶ drop stopwords?
- ▶ try n-grams?
- ▶ How robust across topics?
- ▶ Is this useful for anything besides description?

Others?

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

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

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 :

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

TF-IDF Weighting

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

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

- ▶ 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
 - ▶ 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.

scikit-learn's TfidfVectorizer

https://scikit-learn.org/stable/modules/feature_extraction.html#text-feature-extraction

https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

```
>>> from sklearn.feature_extraction.text import TfidfVectorizer
>>> vectorizer = TfidfVectorizer()
>>> vectorizer.fit_transform(corpus)
<4x9 sparse matrix of type '<... 'numpy.float64'>'
  with 19 stored elements in Compressed Sparse ... format>
```

scikit-learn's TfidfVectorizer

https://scikit-learn.org/stable/modules/feature_extraction.html#text-feature-extraction

https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

```
>>> from sklearn.feature_extraction.text import TfidfVectorizer
>>> vectorizer = TfidfVectorizer()
>>> vectorizer.fit_transform(corpus)
<4x9 sparse matrix of type '<... 'numpy.float64'>'
  with 19 stored elements in Compressed Sparse ... format>
```

- ▶ **corpus** is a sequence of strings, e.g. pandas data-frame columns
- ▶ pre-processing options: strip accents, lowercase, drop stopwords
- ▶ n-grams: can produce phrases up to length n (words or characters)
- ▶ vocab options: min/max frequency, vocab size
- ▶ post-processing: binary, l2 norm, (smoothed) idf weighting, etc.

Other Transformations?

- ▶ e.g., Kelly et al (2019) suggest that including indicators for whether a phrase appears in a document (rather than the count) is often independently predictive

Other Transformations?

- ▶ e.g., Kelly et al (2019) suggest that including indicators for whether a phrase appears in a document (rather than the count) is often independently predictive
- ▶ Could add log counts, quadratics in counts, etc.
- ▶ Could also add pairwise interactions between word counts/frequencies

Other Transformations?

- ▶ e.g., Kelly et al (2019) suggest that including indicators for whether a phrase appears in a document (rather than the count) is often independently predictive
- ▶ Could add log counts, quadratics in counts, etc.
- ▶ Could also add pairwise interactions between word counts/frequencies
- ▶ These often are not done much because of the dimensionality problem
 - ▶ could use feature selection or principal components to help deal with that

Tokenization: Overview

Pre-Processing Text

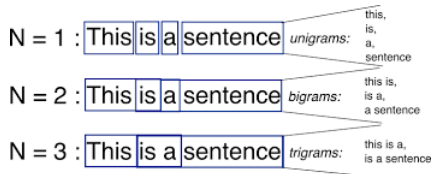
Counts and Frequencies

N-Grams

Parts of Speech

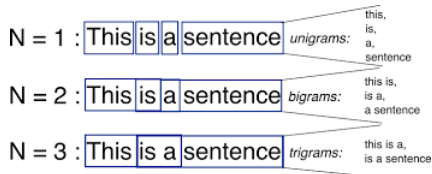
What are N-grams

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



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.

N-grams and high dimensionality

- ▶ N-grams will blow up your feature space:
 - ▶ filtering out uninformative n-grams is necessary

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
 - ▶ I have gotten good performance with 10K or even 2K features
 - ▶ For supervised learning tasks, a decent baseline is to build a vocabulary of 60K, then use feature selection to get down to 10K

De-Confounded Feature Selection

- ▶ What if a feature is important due to a confounding correlation?
 - ▶ e.g. in “Fightin Words” paper: say there are more republicans in congress over time, and the word “kill” coincidentally becomes more popular over time
 - ▶ then the republican-”kill” relationship is a spurious correlation and does not say anything about partisan language

De-Confounded Feature Selection

- ▶ What if a feature is important due to a confounding correlation?
 - ▶ e.g. in “Fightin Words” paper: say there are more republicans in congress over time, and the word “kill” coincidentally becomes more popular over time
 - ▶ then the republican-“kill” relationship is a spurious correlation and does not say anything about partisan language
- ▶ Solution: de-mean the predictors (word frequencies) by year – that way, partisanship is predicted using only within-year variation
 - ▶ can be done with other groups as well – e.g., compare legislators from the same state
 - ▶ can also de-mean the outcome

De-Confounded Feature Selection

- ▶ What if a feature is important due to a confounding correlation?
 - ▶ e.g. in “Fightin Words” paper: say there are more republicans in congress over time, and the word “kill” coincidentally becomes more popular over time
 - ▶ then the republican-“kill” relationship is a spurious correlation and does not say anything about partisan language
- ▶ Solution: de-mean the predictors (word frequencies) by year – that way, partisanship is predicted using only within-year variation
 - ▶ can be done with other groups as well – e.g., compare legislators from the same state
 - ▶ can also de-mean the outcome
- ▶ What if you want to de-mean by both year and state?
 - ▶ → take residuals from linear regression of each variable (outcome and predictor) on the category dummies
 - ▶ That is:
 - ▶ regress $Y_i = FE_1 + FE_2 + \epsilon_i$ and $x_i^w = FE_1 + FE_2 + \epsilon_i, \forall w$,
 - ▶ take residuals $\tilde{Y}_i = Y_i - \hat{Y}_i$ and $\tilde{x}_i^w = x_i^w - \hat{x}_i^w$
 - ▶ Then use residuals as variables, in feature selection step or in machine learning task

Collocations are Familiar N-grams

- ▶ 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 \neq "kick the bucket")

Collocations are Familiar N-grams

- ▶ 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 \neq "kick the bucket")
 - ▶ Non-substitutable: cannot substitute components with synonyms ("fast food" \neq "quick food")

Collocations are Familiar N-grams

- ▶ 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 \neq "kick the bucket")
 - ▶ Non-substitutable: cannot substitute components with synonyms ("fast food" \neq "quick food")
 - ▶ Non-modifiable: cannot modify with additional words or grammar: (e.g., "kick around the bucket", "kick the buckets")

Collocations are Familiar N-grams

- ▶ 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 \neq "kick the bucket")
 - ▶ Non-substitutable: cannot substitute components with synonyms ("fast food" \neq "quick food")
 - ▶ Non-modifiable: cannot modify with additional words or grammar: (e.g., "kick around the bucket", "kick the buckets")
- ▶ But the reduction methods so far do not help identify collocations

Point-wise mutual information

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

$$\begin{aligned}\text{PMI}(w_1, w_2) &= \frac{\text{Pr}(w_1_w_2)}{\text{Pr}(w_1)\text{Pr}(w_2)} \\ &= \frac{\text{Prob. of collocation, actual}}{\text{Prob. of collocation, if independent}}\end{aligned}$$

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

Point-wise mutual information

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

$$\begin{aligned}\text{PMI}(w_1, w_2) &= \frac{\Pr(w_1_w_2)}{\Pr(w_1)\Pr(w_2)} \\ &= \frac{\text{Prob. of collocation, actual}}{\text{Prob. of collocation, if independent}}\end{aligned}$$

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
- ▶ Generalizes to longer phrases (length N) as the geometric mean of the probabilities:

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

- ▶ E.g., for trigrams:

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

Point-wise mutual information

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

$$\begin{aligned}\text{PMI}(w_1, w_2) &= \frac{\Pr(w_1_w_2)}{\Pr(w_1)\Pr(w_2)} \\ &= \frac{\text{Prob. of collocation, actual}}{\text{Prob. of collocation, if independent}}\end{aligned}$$

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
- ▶ Generalizes to longer phrases (length N) as the geometric mean of the probabilities:

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

- ▶ E.g., for trigrams:

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

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

Tokenization: Overview

Pre-Processing Text

Counts and Frequencies

N-Grams

Parts of Speech

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 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
- ▶ In particular, noun phrases are often informative features – spaCy can do fast noun phrase chunking

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
- ▶ In particular, noun phrases are often informative features – spaCy can do fast noun phrase chunking
- ▶ Can count parts of speech tags as features – e.g., using more adjectives, or using more passive verbs

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
- ▶ In particular, noun phrases are often informative features – spaCy can do fast noun phrase chunking
- ▶ Can count parts of speech tags as features – e.g., using more adjectives, or using more passive verbs
- ▶ POS n-gram frequencies (e.g. NN, NV, VN, ...), like function words, are good stylistic features for authorship detection
 - ▶ not biased by topics/content

Named Entity Recognition

- ▶ refers to the task of identifying named entities such as “Basel University” and “Marie Curie”, which can be used as tokens.

[PER John Smith] , president of [ORG McCormik Industries] visited his niece [PER Paris]
in [LOC Milan], reporters say .

Named Entity Recognition

- refers to the task of identifying named entities such as “Basel University” and “Marie Curie”, which can be used as tokens.

[**PER** John Smith] , president of [**ORG** McCormik Industries] visited his niece [**PER** Paris] in [**LOC** Milan] , reporters say .

- detecting the type requires a trained model (e.g. spaCy):

| Type | Tag | Sample Categories | Example sentences |
|----------------------|------------|------------------------------|--|
| People | PER | people, characters | Turing is a giant of computer science. |
| Organization | ORG | companies, sports teams | The IPCC warned about the cyclone. |
| Location | LOC | regions, mountains, seas | The Mt. Sanitas loop is in Sunshine Canyon . |
| Geo-Political Entity | GPE | countries, states, provinces | Palo Alto is raising the fees for parking. |
| Facility | FAC | bridges, buildings, airports | Consider the Golden Gate Bridge . |
| Vehicles | VEH | planes, trains, automobiles | It was a classic Ford Falcon . |

Figure 18.1 A list of generic named entity types with the kinds of entities they refer to.

What to do with out-of-vocab words

- ▶ Typically, have to choose a vocabulary for featurizing a document
 - ▶ e.g., top 10K words by frequency
- ▶ what to do with the words that get dropped out?

What to do with out-of-vocab words

- ▶ Typically, have to choose a vocabulary for featurizing a document
 - ▶ e.g., top 10K words by frequency
- ▶ what to do with the words that get dropped out?
 - ▶ drop them
 - ▶ replace with “unknown” token
 - ▶ replace with part-of-speech tag
 - ▶ replace with in-vocab hypernym (from WordNet)