Lecture 2: Tokenization and Morphology

Lecture 2: Overview

Today, we'll look at words:

- How do we identify words in text?
- Word frequencies and Zipf's Law — What is a word, really?
- What is the structure of words?
- How can we identify the structure of words?

Key Concepts

You should understand the distinctions between

- Word forms vs. lemmas
- Word tokens vs. word types
- Finite-state automata vs. finite-state transducers
- Inflectional vs. derivational morphology

And you should know the implications of Zipf's Law for NLP

Tokenization: Identifying word boundaries

Text is just a `sequence of characters`:

```
Of course he wants to take the advanced course too. He already took two beginners' courses.
```

How do we split this text into words and sentences?

```
[[Of, course, he, wants, to, take, the, advanced, course, too, .],
 [He, already, took, two, beginners', courses, .]]
```

How do we identify the words in a text?

For a language like English, this seems like a really easy problem:

```
A word is any sequence of alphabetical characters between whitespaces that’s not a punctuation mark?
```

That works to a first look, but...

- what about abbreviations like `D.C.`?
- what about complex names like `New York`?
- what about contractions like `doesn’t` or `couldn't've`?
- what about `New York-based`?
- what about names like `SARS-Cov-2`, or `R2-D2`?
- what about languages like Chinese that have no whitespace?

Words aren’t just defined by blanks

Problem 1: Compounding

```
“ice cream”, “website”, “web site”, “New York-based”
```

Problem 2: Other writing systems have no blanks

```
Chinese: 我开始写小说 = 我 开始 写 小说 (I start(ed) writing novel(s))
```

Problem 3: Contractions and Clitics

```
English: “doesn’t” , “I’m” ,
Italian: “dirglielo” = dir + gli(e) + lo (tell + him + it)
```

Tokenization Standards

Any actual NLP system will assume a particular tokenization standard.

- Because so much NLP is based on systems that are trained on particular corpora (text datasets) that everybody uses, these corpora often define a de facto standard.

Penn Treebank 3 standard:

- Input:
  `"The San Francisco-based restaurant," they said, "doesn’t charge $10".`
- Output:
  `“_ The _ San _ Francisco-based _ restaurant _ , _” _ they_ said* ,* "_ does _ n’t _ charge_ $_ 10 _ " _ . _`

What about sentence boundaries?

How can we identify that this is two sentences?

```
Mr. Smith went to D.C. Ms. Xu went to Chicago instead.
```

- Challenge: punctuation marks in abbreviations (Mr., D.C, Ms,...)

```
[It’s easy to handle a small number of known exceptions,  but much harder to identify these cases in general]
```

See also this headline from the NYT (08/26/20):

```
Anthony Martignetti (‘Anthony!’), Who Raced Home for Spaghetti, Dies at 63
```

What about sentence boundaries?

How many sentences are in this text?

```
"The San Francisco-based restaurant," they said, "doesn't charge $10".
```

- Answer: just one, even though "they said" appears in the middle of another sentence.

Similarly, we typically treat this also just as one sentence:

```
They said: "The San Francisco-based restaurant doesn't charge $10".
```

Spelling variants, typos, etc.

The same word can be written in different ways:

- with different `capitalizations`:
  - lowercase “cat” (in standard running text)
  - capitalized “Cat” (as first word in a sentence, or in titles/headlines),
  - all-caps “CAT” (e.g. in headlines)
- with different abbreviation or hyphenation styles:
  - US-based, US based, U.S.-based, U.S. based
  - US-EU relations, U.S./E.U. relations, …
- with spelling variants (e.g. regional variants of English):
  - labor vs labour, materialize vs materialise,
- with typos (teh)

Counting words: tokens vs types

When counting words in text, we distinguish between word types and word tokens:

- The vocabulary of a language is the set of (unique) word types:
  V = {a, aardvark, …., zyzzva}
- The tokens in a document include all occurrences of the word types in that document or corpus
  (this is what a standard word count tells you)
- The frequency of a word (type) in a document
  = the number of occurrences (tokens) of that type

How many different words are there in English?

How large is the vocabulary of English (or any other language)?

- Vocabulary size = the number of distinct word types
  Google N-gram corpus: 1 trillion tokens, 13 million word types that appear 40+ times

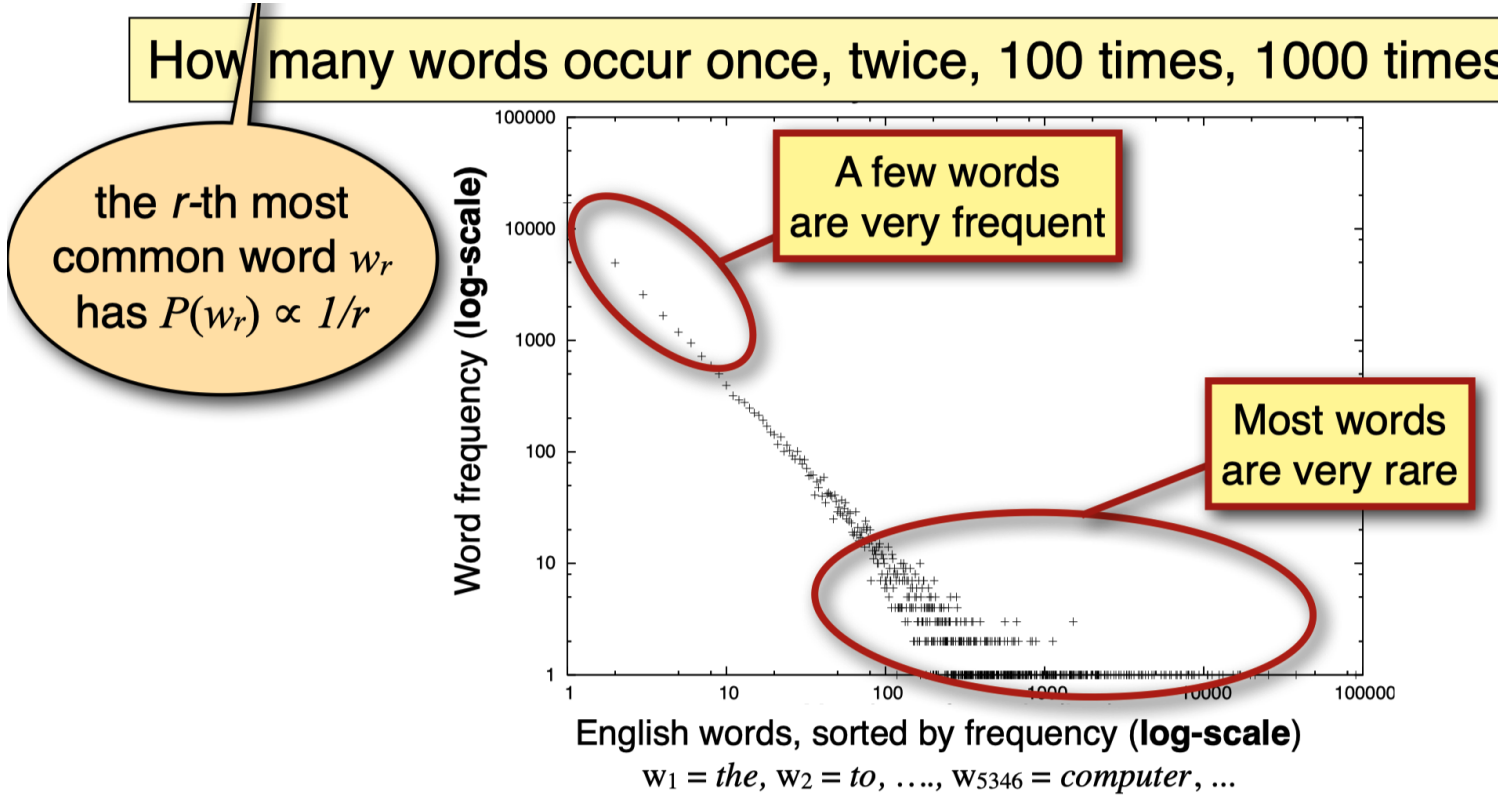
If you count words in text, you will find that...

- a few words (mostly closed-class) are very frequent (the, be, to, of, and, a, in, that,...)
- most words (all open class) are very rare.
- even if you've read a lot of text,
  you will keep finding words you haven't seen before. Word frequency: the number of occurrences of a word type in a text (or in a collection of texts)

Zipf's law: the long tail

In natural language:

- A small number of events (e.g. words) occur with high frequency
- A large number of events occur with very low frequency



Implications of Zipf's Law for NLP

The good:

```
Any text will contain a number of words that are very common.
We have seen these words often enough that we know (almost) everything about them.
These words will help us get at the structure (and possibly meaning) of this text.
```

The bad:

```
Any text will contain a number of words that are rare.
We know something about these words, but haven't seen them often enough to know everything about them.
They may occur with a meaning or a part of speech we haven't seen before.
```

The ugly:

```
Any text will contain a number of words that are unknown to us.
We have never seen them before, but we still need to get at the structure (and meaning) of these texts.
```

Dealing with the bad and the ugly

Our systems need to be able to generalize from what they have seen to unseen events.

There are two (complementary) approaches to generalization:

- Linguistics provides us with insights about the rules and structures in language that we can exploit in the (symbolic) representations we use
  ex.) a finite set of grammar rules is enough to describe an infinite language
- Machine Learning/Statistics allows us to learn models (and/or representations) from real data that often work well empirically on unseen data
  ex.) most statistical or neural NLP

How do we represent words?

Option 1: Words are atomic symbols

- Each (surface) word form is its own symbol
- Add some generalization by mapping different forms of a word to the same symbol
  - `Normalization`: map all variants of the same word (form) to the same canonical variant (e.g. lowercase everything, normalize spellings, perhaps spell-check)
  - `Lemmatization`: map each word to its lemma (esp. in English, the lemma is still a word in the language, but lemmatized text is no longer grammatical)
  - `Stemming`: remove endings that differ among word forms (no guarantee that the resulting symbol is an actual word)

How do we represent words?

Option 2: Represent the structure of each word

```
"books" => "book N pl" (or "book V 3rd sg")
```

- This requires a morphological analyzer
- The output is often a lemma (“book”) plus morphological information (“N pl” i.e. plural noun)
- This is particularly useful for highly inflected languages, e.g. Czech, Finnish, Turkish, etc. (less so for English or Chinese):

How do we represent unknown words?

Many NLP systems assume a fixed vocabulary, but still have to handle out-of-vocabulary (OOV) words.

- Option 1: `the UNK token`
  Replace all rare words (with a frequency at or below a given threshold, e.g. 2, 3, or 5) in your training data with an UNK token (UNK = “Unknown word”). Replace all unknown words that you come across after training (including rare training words) with the same UNK token
- Option 2: `substring-based representations`
  [often used in neural models]
  Represent (rare and unknown) words [“Champaign”] as sequences of characters [‘C’, ‘h’, ‘a’,…,’g’, ’n'] or substrings [“Ch”, “amp”, “ai”, “gn”]
  `Byte Pair Encoding (BPE)` : learn which character sequences are common in the vocabulary of your language, and treat those common sequences as atomic units of your vocabulary

How many different words are there in English?

How large is the vocabulary of English (or any other language)?

- Vocabulary size = the number of distinct word types

  Google N-gram corpus: 1 trillion tokens, 13 million word types that appear 40+ times

  `[here, we're treating inflected forms (took, taking) as distinct]`
- You may have heard statements such as

  `adults know about 30,000 words`

  `you need to know at least 5,000 words to be fluent`

  Such statements do not refer to inflected word forms (take/takes/taking/take/takes/took) but to lemmas or dictionary forms (take), and assume if you know a lemma, you know all its inflected forms too.

Which words appear in this text?

```
Of course he wants to take the advanced course too. He already took two beginners' courses.
```

Actual text doesn't consist of dictionary entries:

- wants is a form of want
- took is a form of take
- courses is a form of course

Linguists distinguish between

- **the (surface) forms that occur in text**: want, wants, beginners', took,...
- **and the lemmas that are the uninflected forms of these words**: want, beginner, take, ...

In NLP, we sometimes map words to lemmas (or simpler "stems"), but the raw data always consists of surface forms

How many different words are there?

`Inflection` creates different forms of the same word:

- Verbs: to be, being, I am, you are, he is, I was,
- Nouns: one book, two books

`Derivation` creates different words from the same lemma:

- grace ⇒ disgrace ⇒ disgraceful ⇒ disgracefully

`Compounding` combines two words into a new word:

- cream ⇒ ice cream ⇒ ice cream cone ⇒ ice cream cone bakery

Word formation is productive:

- New words are subject to all of these processes:
  Google ⇒ Googler, to google, to ungoogle, to misgoogle, googlification, ungooglification, googlified, Google Maps, Google Maps service,...

Inflectional morphology in English

Verbs:

- Infinitive/present tense: walk, go
- 3rd person singular present tense (s-form): walks, goes
- Simple past: walked, went
- Past participle (ed-form): walked, gone
- Present participle (ing-form): walking, going

Nouns:

- Common nouns inflect for number:
  singular (book) vs. plural (books)
- Personal pronouns inflect for person, number, gender, case:
  I saw him; he saw me; you saw her; we saw them; they saw us.

Derivational morphology in English

Nominalization:

- V + -ation: computerization
- V+ -er: killer
- Adj + -ness: fuzziness

Negation:

- un-: undo, unseen, ...
- mis-: mistake,...

Adjectivization:

- V+ -able: doable
- N + -al: national

Morphemes: stems, affixes

```
  dis-grace-ful-ly
prefix-stem-suffix-suffix
```

Many word forms consist of a `stem` plus a number of `affixes (prefixes or suffixes)`

- Exceptions: Infixes are inserted inside the stem Circumfixes (German gesehen) surround the stem

`Morphemes` : the smallest (meaningful/grammatical) parts of words.

- Stems (grace) are often `free morphemes` .
  Free morphemes can occur by themselves as words.
- Affixes (dis-, -ful, -ly) are usually `bound morphemes` .
  Bound morphemes have to combine with others to form words.

Morphemes and morphs

The same information (plural, past tense, ...) is often expressed in different ways in the same language.

- One way may be more common than others, and exceptions may depend on specific words:
  - Most plural nouns: add **-s** to singular: book-book**s**, but: box-box**es**, fly-fl**ies**, child-child**ren**
  - Most past tense verbs add **-ed** to infinitive: walk-walk**ed**, but: like-like**d**, leap-leap**t**
    Such exceptions are called `irregular word forms`

Linguists say that there is `one underlying morpheme` (e.g. for plural nouns) that is "realized" as different "surface" forms (morphs) (e.g. -s/-es/-ren)

- Allomorphs: two different realizations (-s/-es/-ren) of the same underlying morpheme (plural)

“Surface”?

This terminology comes from Chomskyan Transformational Grammar.

- Dominant early approach in theoretical linguistics, superseded by other approaches (“minimalism”).
- Not computational, but has some historical influence on computational linguistics (e.g. Penn Treebank)

`“Surface”` = standard English (Chinese, Hindi, etc.).

- `“Surface string”` = a written sequence of characters or words

vs. `“Deep”/“Underlying”` structure/representation:

- A more abstract representation.
- Might be the same for different sentences/words with the same meaning.

Tokenization

Input:

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

Output (tokens):

- A sequence, W , containing a list of tokens – words or word pieces for use in natural language processing

Output (n-grams):

- A matrix, X, containing statistics about word/phrase frequencies in those documents.

Goals of Tokenization

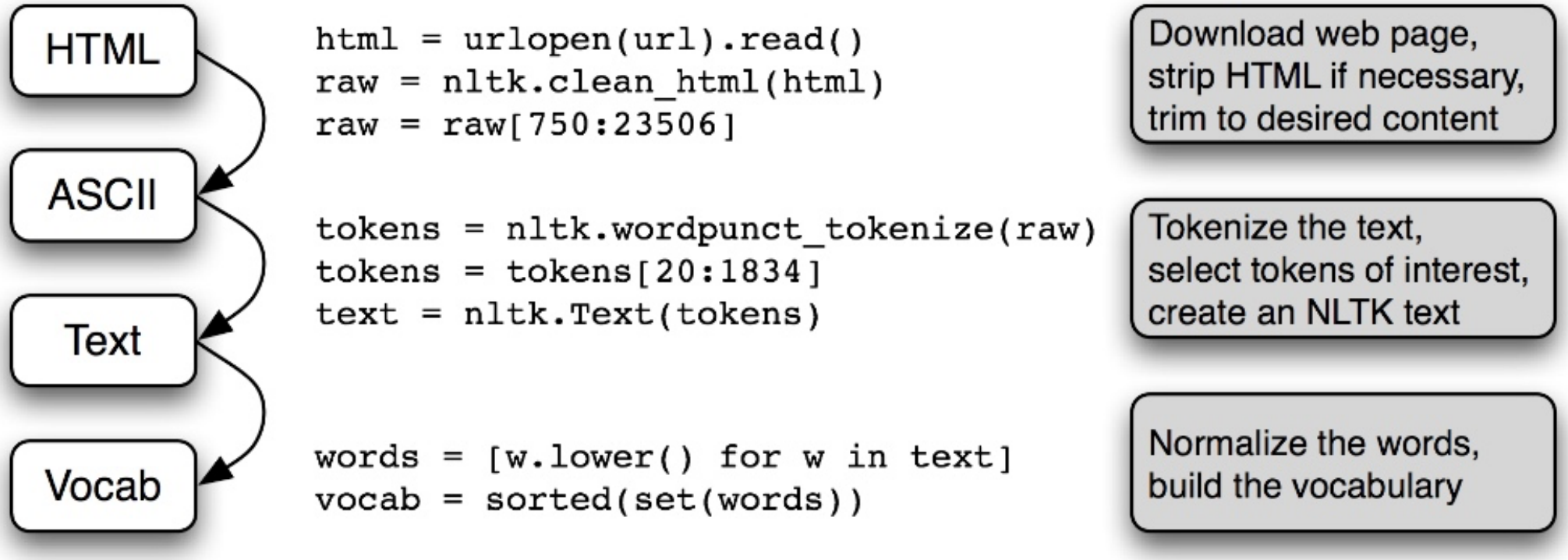
To summarize: A major goal of tokenization is to produce features that are

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

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



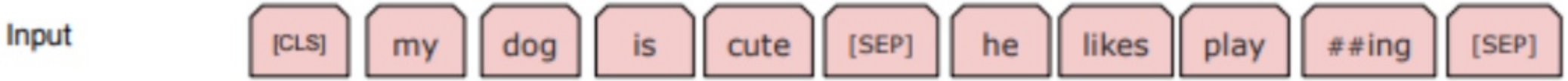
Source: NLTK Book, Chapter 3.

Subword Tokenization for Sequence Models

Modern transformer models (e.g. BERT, GPT) use subword tokenization:

- construct character-level n-grams
- whitespace treated the same as letters
- all letters to lowercase, but add a special character for the next letter being capitalized.

e.g., BERT's SentencePiece tokenizer:



- character-level byte-pair encoder, learns character n-grams to breaks words like "playing" into "play" and "##ing".
- have to fix a vocabulary size: e.g. BERT uses 30K.

Segmenting paragraphs/sentences

Many tasks should be done on sentences, rather than corpora as a whole.

- spaCy is a good (but not perfect) job of splitting sentences, while accounting for periods on abbreviations, etc.
- pySBD is a better option for splitting sentences.

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: “govenor has” / “has discretion” vs “govenor 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.

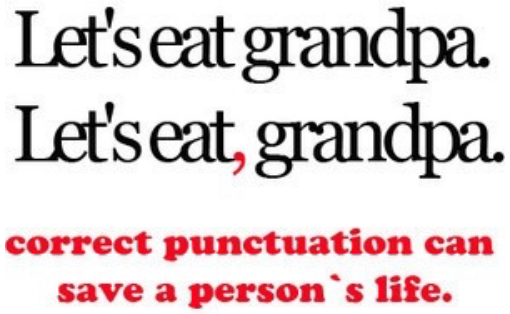
Also: what about “the first amendment” versus “the First Amendment”?

- 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, syntactic parsing, and semantic role labeling.
- For sequence data, e.g. language modeling – huggingface tokenizer takes out capitalization but then add a special “capitalized” token before the word.

Punctuation



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)
    - also needed for language models.

Numbers

for classification using bag of words:

- can drop numbers, or replace with special characters

for language models:

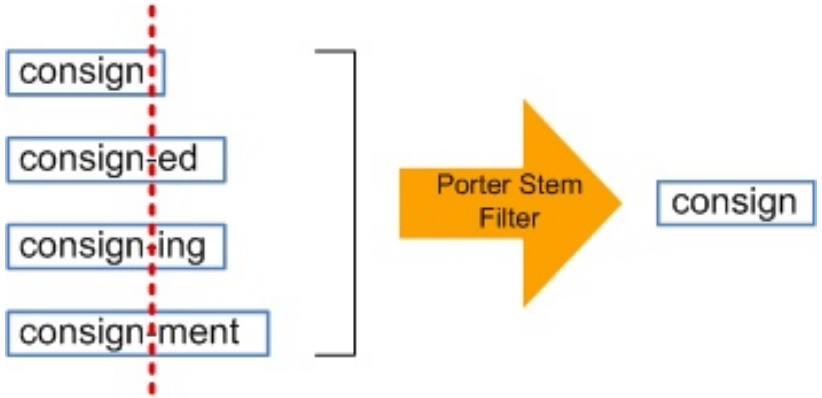
- just treat them like letters.
- GPT-3 can solve math problems (but not well, this is an area of research)

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"

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:
  `“hellow world → hellow_world”`
- what else?

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.

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` :

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

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:

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

- 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 0log(100/10) ≈ 2.3, so the TF-IDF for this document is .03×2.3 = .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>
```

- corpus is a sequence of strings, e.g. pandas data-frame columns.
- l 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

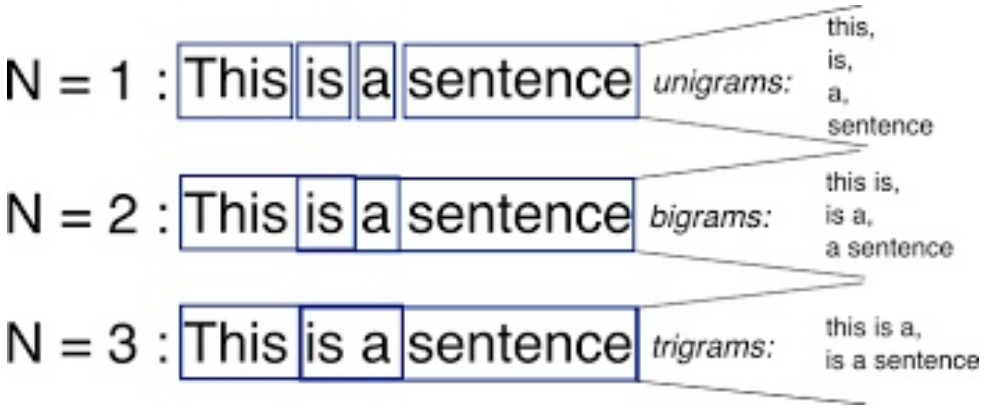
Other Transformations?

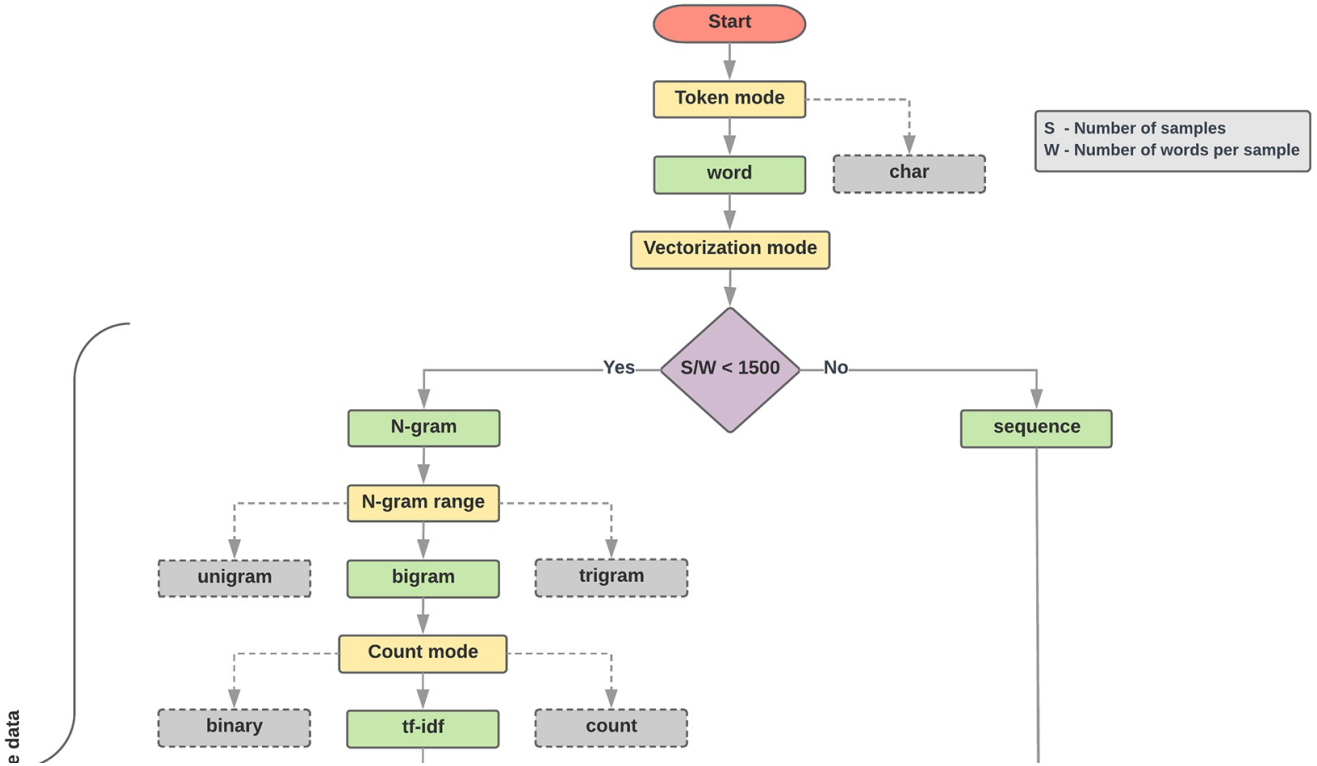
- e.g., Kelly et al (2019) suggest that including indicators for whether a phrase appears in a document (rather than the count) is often independently predictive.
- Could add log counts, quadratics in counts, etc.
- Could also add pairwise interactions between word counts/frequencies.
- These often are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

What are N-grams

N-grams are phrases, sequences of words up to length N.

- bigrams, trigrams, quadgrams, etc.





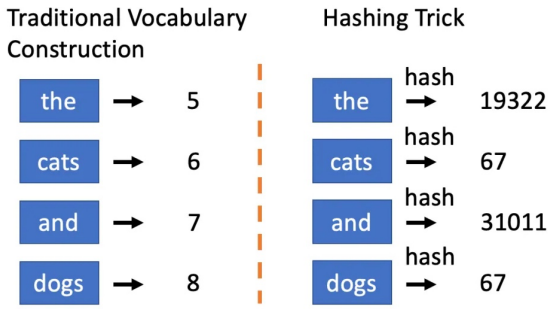
- Google Developers recommend tf-idf-weighted bigrams as a baseline specification for text classification tasks.
    - ideal for fewer, longer documents.
- With more numerous, shorter documents (rows / doclength > 1500), better to use an embedded sequence (starting Week 5).

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.

Hashing Vectorizer

Rather than make a one-to-one lookup for each n-gram, put n-grams through a hashing function that takes an arbitrary string and outputs an integer in some range (e.g. 1 to 10,000).




```
>>> from sklearn.feature_extraction.text import HashingVectorizer
>>> hv = HashingVectorizer(n_features=10)
>>> hv.transform(corpus)
<4x10 sparse matrix of type '<... 'numpy.float64'>'
    with 16 stored elements in Compressed Sparse ... format>
```

Hashing Vectorizer

- Pros:
  - can have arbitrarilly small feature space
  - handles out-of-vocabulary words – any word or n-gram gets assigned to an arbitrary integer based on the hash function.
- Cons:
  - harder to interpret features, at least not directly – but the eli5 implementation keeps track of the mapping
  - collisions – n-grams will randomly be paired with each other in the feature map.
    - usually innocuous, but could sum outputs of two hashing functions to minimize this.

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")

Point-wise mutual information

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

$$PMI(w_1, w_2) = \frac{P(w_1\_w_2)}{P(w_1)P(w_2)}$$
$$= \frac{\text{Prob. of collocation, actual}}{\text{Prob. of collocation, if independent}}$$

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.

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

$$\frac{P(w*1, \ldots w_2)}{\prod *i = 1^n \sqrt[n]{P(w_i)}}$$

Warning: Rare words that appear together once or twice will have high PMI.

Address this with minimum frequency thresholds.

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.

Parts of speech tags

- 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-gam frequencies (e.g. NN, NV, VN, ...), like function words, are good stylistic features for authorship detection.
    - not biased by topics/content

What do do with out-of-vocab words

- unless using a hashing vectorizer, 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
  - run (auxiliary) hashing vectorizer on them replace
  - with in-vocab hypernym (from WordNet) others?

Parts of Speech Predict Loan Repayment

Netzer, Lemaire, and Herzenstein (2019), "When Words Sweat"

Imagine you consider lending $2,000 to one of two borrowers on a crowdfunding website. The borrowers are identical in terms of demographic and financial characteristics. However, the text they provided when applying for a loan differs:

```
Borrower #1:

"I am a hard working person, married for 25 years, and have two wonderful boys.
Please let me explain why I need help.
I would use the $2,000 loan to fix our roof.
Thank you, god bless you, and I promise to pay you back."

Borrower #2:

"While the past year in our new place has been more than great,
the roof is now leaking and I need to borrow $2,000 to cover the cost of the repair.
I pay all bills (e.g., car loans, cable, utilities) on time."
```

Parts of Speech Predict Loan Repayment

Which borrower is more likely to default?

“Loan requests written by defaulting borrowers are more likely to include words (or themes) related to the borrower’s family, financial and general hardship, mentions of god, and the near future, as well as pleading lenders for help, and using verbs in present and future tenses.”

Loan Application Words Predicting Repayment (Netzer, Lemaire, and Herzenstein 2019)

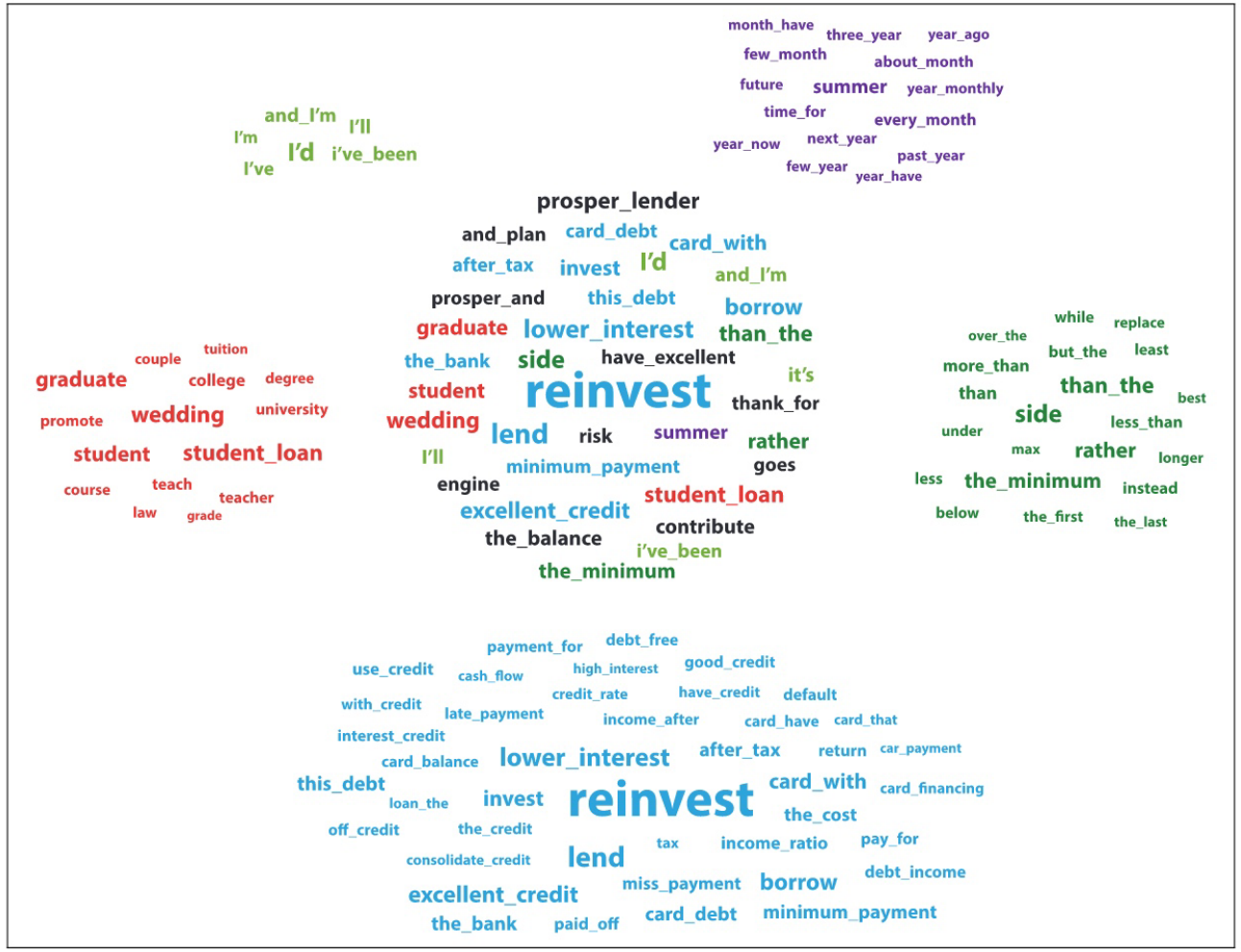


Figure 2. Words indicative of loan repayment.

Notes: The most common words appear in the middle cloud (cutoff = 1:1.5) and are then organized by themes. Starting on the right and moving clockwise: relative words, financial literacy words, words related to a brighter financial future, "I" words, and time-related words.

Loan Application Words Predicting Default (Netzer, Lemaire, and Herzenstein 2019)

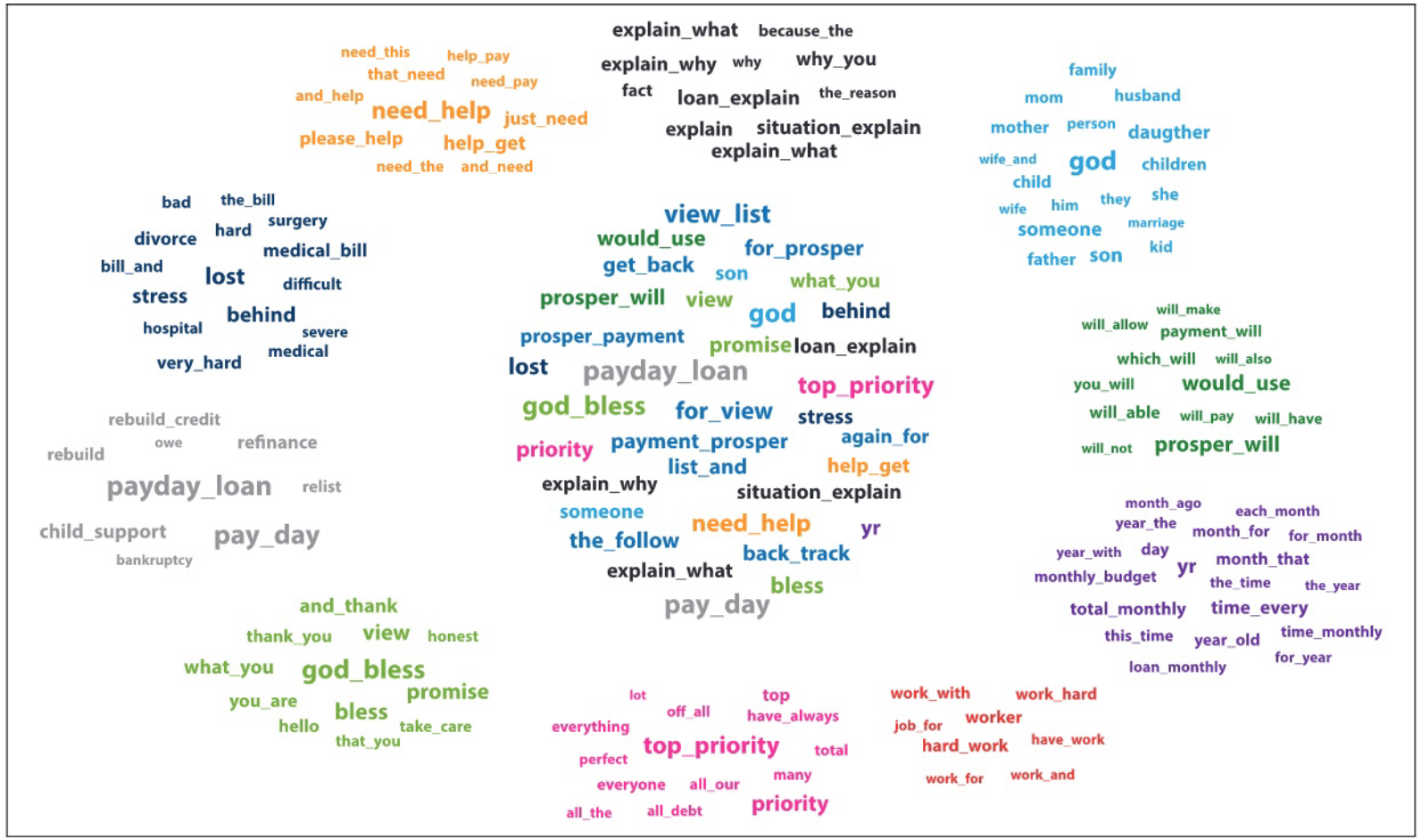


Figure 3. Words indicative of loan default.

Notes: The most common words appear in the middle cloud (cutoff = 1:1.5) and are then organized by themes. Starting on the top and moving clockwise: words related to explanations, external influence words and others, future-tense words, time-related words, work-related words, extremity words, words appealing to lenders, words relating to financial hardship, words relating to general hardship, and desperation/plea words.