WEB INTELLIGENCE

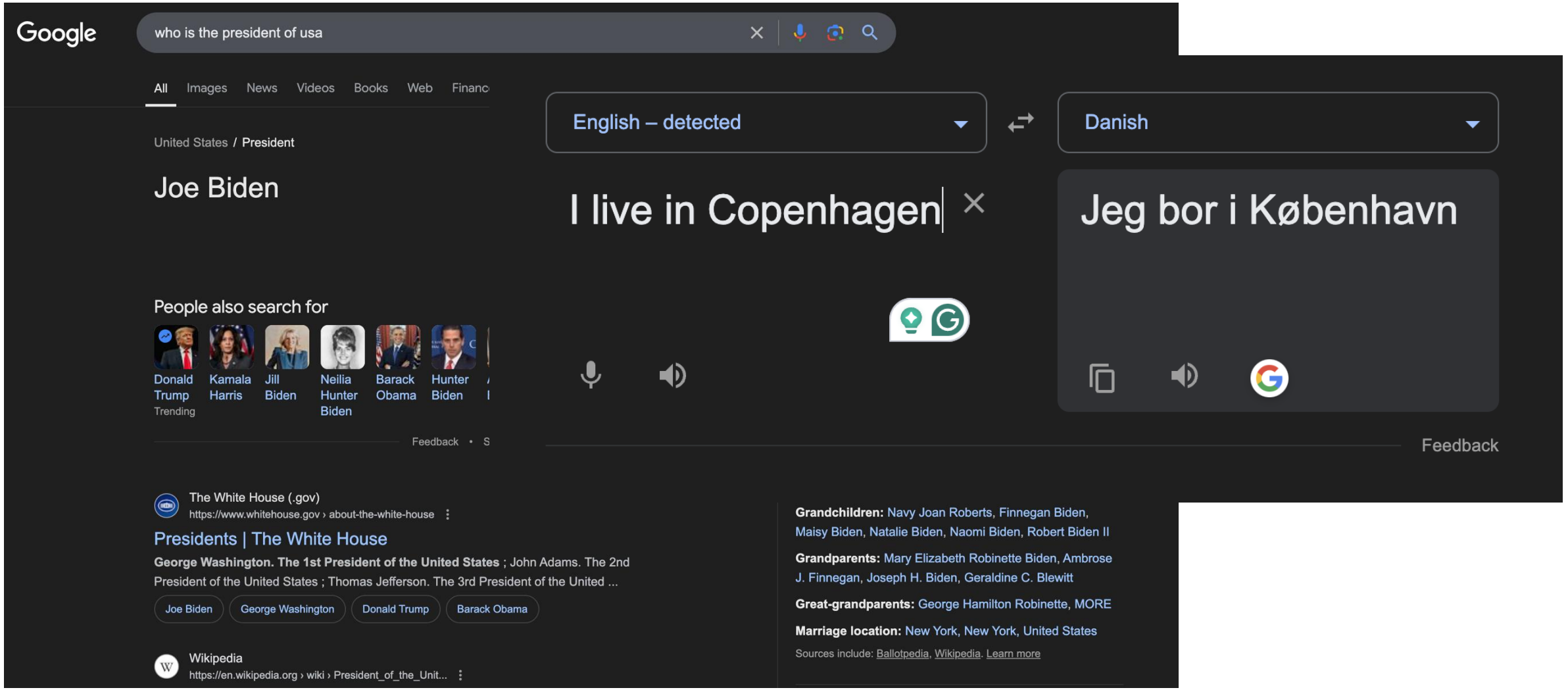
Lecture 2:
NLP Basics

Russa Biswas

September 09, 2025

AALBORG UNIVERSITY

What is Natural Language Processing?



What is Natural Language Processing?

Natural Language Processing is a sub field of Artificial Intelligence and Computational Linguistics, comprising of computational methods for understanding or generating Natural Languages.

Goal:

- Read human languages into machine understandable
- Decipher human languages
- Understand and makes sense of the text

Challenge: Making the meaning of the human language machine understandable.

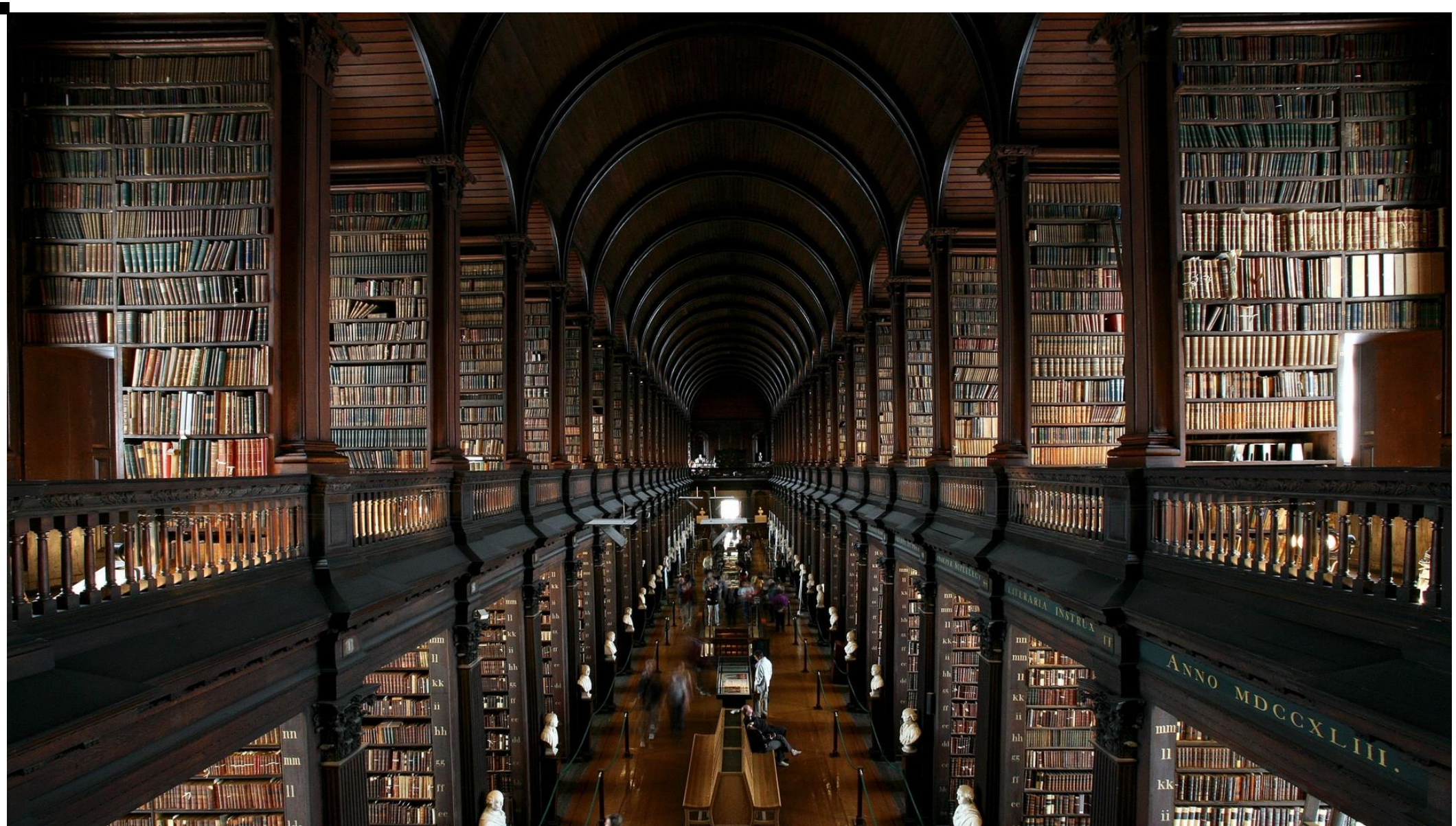
Natural Languages consists of

- Phonology: Sounds of the words
- Semantics: Meaning of the words
- Syntax: Grammatical rules according to which words are put together

Why is it difficult?

- Ambiguity: same text can have very different meanings or different texts can have the same meaning
  - I saw a bat in the garden. (flying mammal or sports equipment) – Word level
  - The man saw the boy with the telescope. (Did the man use a telescope to see the boy, or did the boy have the telescope?)
  - The spy saw the man with the binoculars on the hill. (Who has the binoculars, and is the man or the spy on the hill?)
  - “She loves reading books.” “She enjoys reading novels.” (diff. text same meaning)
  - As the deadline approached, John was **burning the midnight oil** to finish the project. (multiword expression/idioms)
- Sociolinguistics, Cultural, Colloquial differences as well as informal languages (social media text)
- So many languages, different languages have different features
- Mixing different languages in the same sentence or same piece of text – Code /Language Switching
  - "I’ll meet you tomorrow at 5 बजे, ठीक है?”
  - Translation: "I’ll meet you tomorrow at 5 o'clock, is that fine?"

Unstructured Data



Understanding Languages

I love learning languages.

私は言語を学ぶことが大好きです。

Jeg elsker at lære sprog.

मुझे भाषाएँ सीखना पसंद है।

Me encanta aprender idiomas.

나는 언어를 배우는 것을 좋아합니다.

Tokenization

Tokenization involves segmenting text into smaller units that are analyzed individually.

I love learning languages. → [“I”, “love”, “learning”, “languages”, “.”]

Initial heuristics: Split on **whitespaces** or **punctuations** [for languages like English, Danish, German, Spanish, Hindi, etc.]

However, its language dependent.

For Korean and Japanese it’s a combination of characters and syllabic scripts, no white spaces considered.

私は言語を学ぶことが大好きです。 → ['私', 'は', '言語', 'を', '学ぶ', 'こと', 'が', '大好き', 'です', '。']

나는 언어를 배우는 것을 좋아합니다. → ['나는', '언어를', '배우는', '것을', '좋아합니다', '.'].

Tokenization

J.K. Rowling's book, Harry Potter and the Philosopher's Stone was published in 1997. It's still a bestseller and truly amazing how much people enjoy it!

Word level tokenization:

Tokenized words: ["J.K.", "Rowling's", "book", ",", "Harry", "Potter", "and", "the", "Philosopher's", "Stone", "was", "published", "in", "1997", ".", "It's", "still", "a", "bestseller", "and", "truly", "amazing", "how", "much", "people", "enjoy", "it", "!"]

Handling contractions:

The contraction "it's" needs to be split into two tokens: ['it', "'s"]. Some systems might tokenize this as two tokens

Punctuation handling:

Commas, apostrophes, and periods are separated from words, ensuring they are individual tokens

Named entity recognition (NER):

The book title and the author are multi-word entities, which ideally should be handled as a single token or entity for proper semantic analysis.

- *Harry Potter and the Philosopher's Stone*
- *J.K. Rowling*

Tokenization

- most tokenizers are rule-based
- several conventions:

	Penn Treebank	Moses
don't	do n't	don 't
aren't	are n't	aren 't
can't	ca n't	can 't
won't	wo n't	won 't

- important to be consistent across NLP systems, match tokenization of external tools/resources

- tokenization usually only *adds* whitespace
- might we also want to remove whitespace?

names:

New York → NewYork?

non-compositional compounds:

hot dog → hotdog?

The Penn Treebank is a human-annotated and partially `skeletally' parsed corpus consisting of over **4.5 million words of American English**.

Moses -> rule based

http://text-processing.com/demo/tokenize/

Tokenization in Encoding

Digital documents that are the input to an indexing process are typically bytes in a file; we need a sequence of characters

ACT

ASCII values (in decimal):

- A = 065
- C = 067
- T = 084

CAT

ASCII values (in decimal):

- C = 067
- A = 065
- T = 084

Tokenization in Encoding

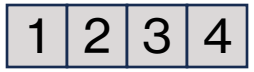
I live in Copenhagen = [“I”, ”live”, “in”, “Copenhagen”]

I = 1

live = 2

in = 3

Copenhagen = 4



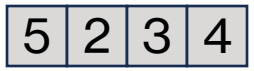
We live in Copenhagen = [“We”, ”live”, “in”, “Copenhagen”]

We = 5

live = 2

in = 3

Copenhagen = 4



Tokenizer model in LLMs

- With large text, many tokens are used only once or twice
- Can specify the no. of words you want the neural model to process
- Assigns tokens to words based on the popularity of the words, i.e., the most popular word will be at index 1 (rank 1)

Normalization

Transform distinct 'equivalent' tokens into one normalized form. E.g.:

- write all in lower case: `This` → `this`
- use non-hyphenated forms: `anti-discriminatory` → `antidiscriminatory`
- delete periods: `U.S.A` → `USA` (or `usa`)

Have to balance:

- more normalization:
  - smaller index
  - more permissive search: user searching for 'U.S.A' also receives results containing 'usa'
- less normalization:
  - supports more specific search: users searching for 'C.A.T.' don't receive results for 'cat'. (try Google vs. Bing on this one!)

Stop word removal

Stop word removal

Stop words: very frequent words that are not semantically descriptive:

the, a, this, and, of, that, . . .

Stop list: list of stop words that are removed. E.g. containing 15-200 stop words.

Problem: stop words may become important as part of an *expression*:

"To be or not to be"

Stemming

Stemming

Similar to normalization: reduce different grammatical variants to their common underlying word "stem":

Inflectional forms:

learn, learns, learned $\rightarrow$ *learn*

Word types:

organize, organizer, organization $\rightarrow$ *organ*

but also:

organ, organizer, organic $\rightarrow$ *organ*

Terms

The strings that result from stemming are the **terms** that will be included in the index.

Porter's Stemming Algorithm

- The most common stemmer for English, introduced by Martin Porter in 1980
- A rule-based stemmer with rules for mostly suffix-stripping such as:
  - "ing" → "-" connecting → connect
  - "sses" → "ss" caresses → caress
  - "ies" → "i" ponies → poni
  - "s" → "-" cats → cat
- **[C](VC){m}[V]**

where C is the consonant, V is the vowel i.e., A,E,I,O,U and other than Y preceded by a consonant, m>0 is the number of times (VC) occurs.

  - m = 0, TREE, BY
  - m = 1, TROUBLE, OATS, TREES
  - m = 2, TROUBLES, OATEN, PRIVATE
- The rules for removing suffix: (condition) S1 -> S2

Porter's Stemming Algorithm

- [C](VC){m}[V]
- Stem the word "REPLACEMENT"
  - m >1, remove EMENT
  - Generates a stem "replac"

- Advantage:
  - It produces the best output as compared to other stemmers, and it has less error rate
- Disadvantage:
  - Morphological variants produced are not always real words (produces stems)

https://people.scs.carleton.ca/~armyunis/projects/KAPI/porter.pdf

Porter's Stemming Algorithm

Use the algorithm to stem the word

MULTIDIMENSIONAL

Stemming

- Language dependent,
  - For e.g., in Chinese stemming is not effective
- There are three criteria for evaluating stemmers:
  - Correctness
  - Efficiency of the task
  - Compression performance
- There are two ways in which stemming can be incorrect:
  - Over-stemming (too much of the term is removed)
    - Two or more words being reduced to the same wrong root
      - e.g., ‘centennial’, ‘century’, ‘center’ → **cent**
  - Under-stemming (too little of the term is removed)
    - Two or more words could be wrongly reduced to more than one root word
      - e.g., ‘acquire’, ‘acquiring’, ‘acquired’ → **acquir** ‘acquisition’ → **acquis**

Zipf's law

"The cat sat on the mat, and the dog sat on the mat."

Tokenize?

["The", "cat", "sat", "on", "the", "mat", ",", "and", "the", "dog", "sat", "on", "the", "mat", "."]

Type: a unique word

Token: an instance of a type in a corpus

Word Frequencies:

the: 3

sat: 2

on: 2

mat: 2

cat: 1

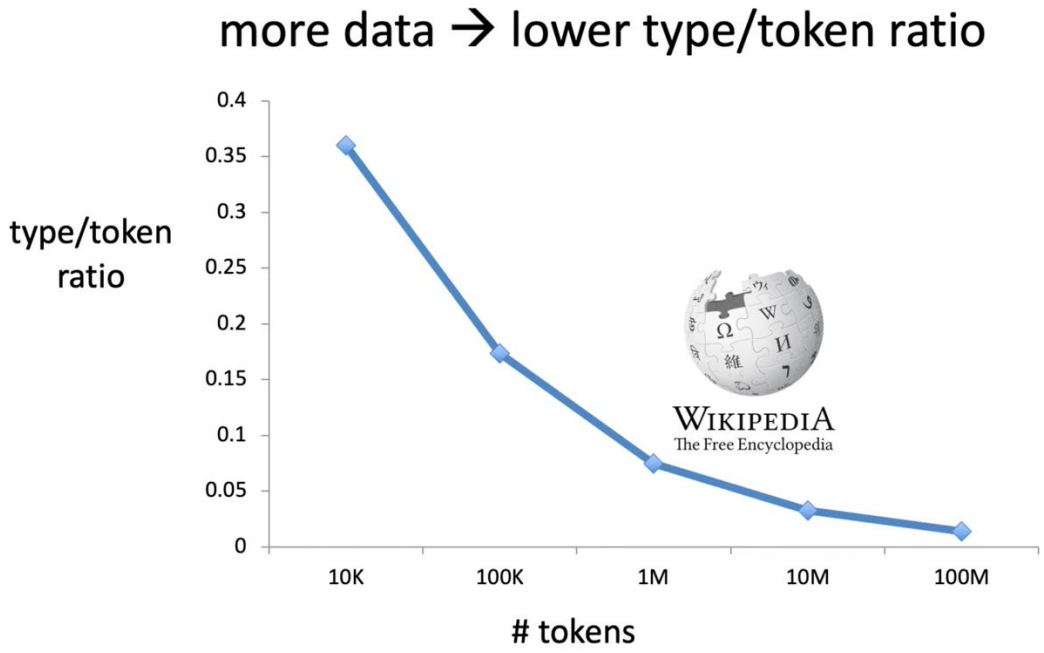
dog: 1

and: 1

Type: 7

Token: 13

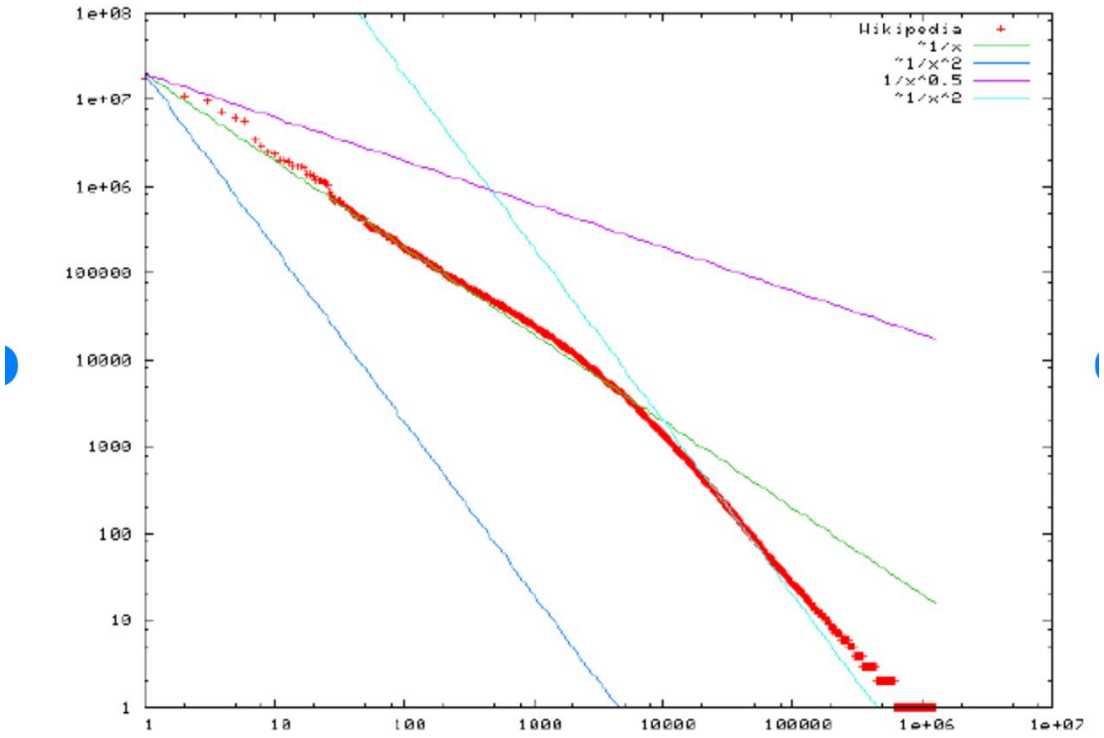
Type/Token Ratio: 7/13



Zipf's law

- **Zipf's law: frequency of a word is inversely proportional to its rank in the word frequency list**

$$word\ frequency \propto \frac{1}{word\ rank}$$



A plot of word frequency in Wikipedia (November 27, 2006). The plot is in log-log coordinates. x is rank of a word in the frequency table; y is the total number of the word's occurences. Most popular words are "the", "of" and "and", as expected. Zipf's law corresponds to the upper linear portion of the curve, roughly following the green (1/x) line (Victor Grishchenko)

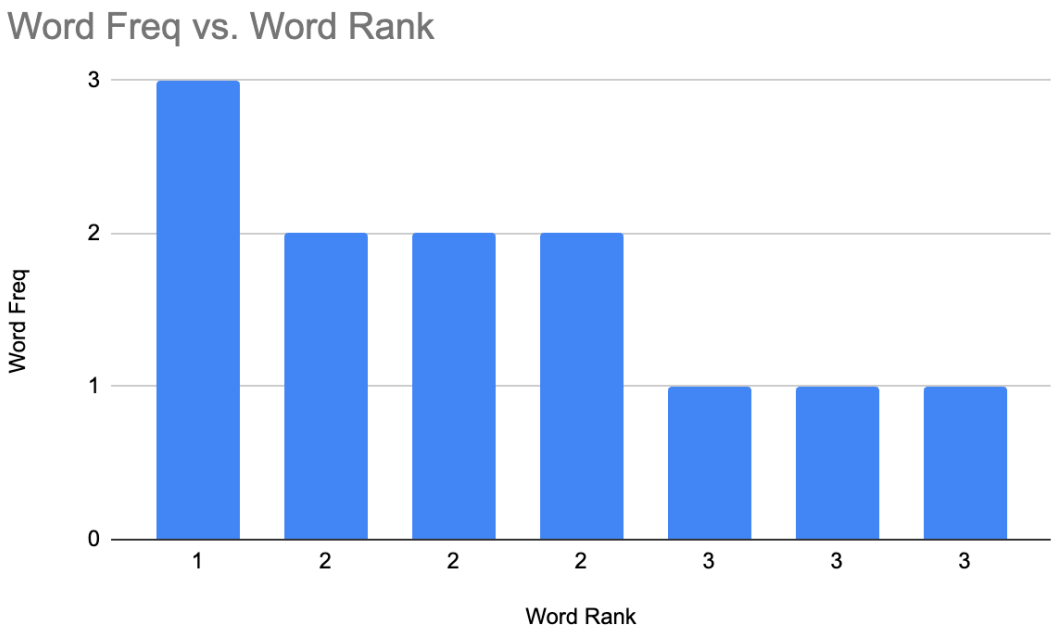
Zipf’s law

“The cat sat on the mat, and the dog sat on the mat.”

Tokenize?

[“The”, “cat”, “sat”, “on”, “the”, “mat”, “,”, “and”, “the”, “dog”, “sat”, “on”, “the”, “mat”, “.”]

Rank-Ordered Frequencies:
1."the" (frequency: 3) – Rank 1
2."sat" (frequency: 2) – Rank 2
3."on" (frequency: 2) – Rank 2 (tied with "sat")
4."mat" (frequency: 2) – Rank 2 (tied with "sat" and "on")
5."cat" (frequency: 1) – Rank 3
6."dog" (frequency: 1) – Rank 3 (tied with "cat")
7."and" (frequency: 1) – Rank 3 (tied with "cat" and "dog")



Zipf's law - Implications

Positive Implications

- Any document/text will contain a number of words that are very common.
- Help us understand the structure (and possibly meaning) of the text

Negative Implications

- Any document/text will have a large number of rare words known as Out-of-vocabulary (OOV) words

Handling OOV words:

- Replace the set of OOV words in the training data with an unknown word token (-UNK)
- Use of statistical models
- Use sub-string based representations such as Byte-Pair Encoding

Byte-Pair Encoding: learn which character sequences are common in the vocabulary of the language, and treat those common sequences as atomic units of the vocabulary

Preprocessing

- Tokenization
- Normalization
- Stop word removal
- Stemming

"The quick brown foxes are jumping over the lazy dogs."

Preprocessing

- **Tokenized**: ['The', 'quick', 'brown', 'foxes', 'are', 'jumping', 'over', 'the', 'lazy', 'dogs', '.']
- **Normalized**: ['the', 'quick', 'brown', 'foxes', 'are', 'jumping', 'over', 'the', 'lazy', 'dogs']
- **After Stop Word Removal**: ['quick', 'brown', 'foxes', 'jumping', 'lazy', 'dogs']
- **After Stemming**: ['quick', 'brown', 'fox', 'jump', 'lazy', 'dog']

Corpus: collection of documents (e.g., all web-pages that have been crawled)

Raw corpora have only minimal (or no) processing:
Sentence boundaries may or may not be identified.
There may or may not be metadata.
Typos (written text) or disfluencies (spoken language) may or may not be corrected.

Annotated corpora contain some labels (e.g. POS tags, sentiment labels), or linguistic structures (e.g syntax trees, semantic interpretations), etc.

Vocabulary: all terms that appear in the corpus

Part of Speech (POS) tagging

The quick brown fox jumps over the lazy dog.

The – Determiner (DT)

quick – Adjective (JJ)

brown – Adjective (JJ)

fox – Noun (NN)

jumps – Verb (VBZ)

over – Preposition (IN)

the – Determiner (DT)

lazy – Adjective (JJ)

dog – Noun (NN)

Part-of-speech tagging, or POS tagging, is a task that entails classifying words in a text according to their grammatical categories (such as noun, verb, and adjective).

Advantages:

- Helps in identifying the syntactic structure
- POS tagging often helps disambiguate the meaning of such words based on their role in the sentence.
- Aids Named Entity Recognition

https://cogcomp.seas.upenn.edu/page/demo_view/pos

String Similarity

Hamming distance: between two equal-length strings of symbols is the number of positions at which the corresponding symbols are different.

Binary Strings:

- Strings: 11001 and 10110
- Hamming Distance: 4
- **Text Strings:**
- Strings: Base and Case
- Hamming Distance: 1 (differ at position 1)

String Similarity

Levenshtein distance measures the minimum number of single-character edits (insertions, deletions, or substitutions) required to transform one string into another.

Strings: kitten and sitting

kitten → sitten (substitute k with s)

sitten → sittin (substitute e with i)

sittin → sitting (insert g at the end)

Levenshtein Distance: 3

Strings: flaw and lawn

flaw → law (delete f)

law → lawn (insert n at the end)

Levenshtein Distance: 2

String Similarity

Jaccard Similarity is a statistic to determine the similarity or the overlap between the two sets

$$\text{Jaccard Similarity: } \frac{A \cap B}{A \cup B}$$

where A and B are two sets, $A \cap B$ is the intersection of two sets (common items), and $A \cup B$ is the union of two sets (all unique elements)

Example:

String 1: "Cats are wonderful pets”

Set A: {Cats, are, wonderful, pets}

String 2: "Dogs are wonderful companions”

Set B: {Dogs, are, wonderful, companions}

$A \cap B$ = {are, wonderful}

$A \cup B$ = {Cats, are, wonderful, pets, Dogs, companions}

Jaccard Similarity: $\frac{A \cap B}{A \cup B}$ = 2/6 = 0.33

Jaccard Distance: dissimilarity between the two strings

Jaccard Distance = 1 - $\frac{A \cap B}{A \cup B}$

= 1 – 0.33 = 0.67

Interpretation:
A Jaccard Distance of 0.6667 means the two sets are moderately dissimilar, with some overlap in the words "are" and "wonderful."

Literature

- https://people.scs.carleton.ca/~armyunis/projects/KAPI/porter.pdf
- Adamic, L.A. and Huberman, B.A., 2002. Zipf's law and the Internet. *Glottometrics*, *3*(1), pp.143-150.
- Introduction to information retrieval Book by Christopher D. Manning, Hinrich Schütze, and Prabhakar Raghavan
- https://caiml.org/summerschool2023/program/krenn-petrak-slides.pdf