Representing and Comparing Texts

HSS 510 / DS 518: NLP for HSS

Taegyoon Kim

Mar 11, 2025

Agenda

Things to be covered

- Unit of analysis
- Tokenization (segmentation)
- Text normalization (= cleaning)
- BoW / vector space models
- Cosine similarity
- TF-IDF weighting
- Guided coding: segmentation, normalization, representation (Python)

Unit of Analysis

The main element that is being analyzed in a study

- "What" or "who" that is being studied
- Depends on the research question

Unit of Analysis

Typically, information about one unit is recorded as one row

- Think of a survey data set

	A	B	C	D	E	F	G	H	I	J	K
1	ID	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
2	10010101	Strongly Agree	Strongly Agree	Agree	Agree	Disagree	Disagree	Agree	Strongly Agree	Disagree	Disagree
3	10010102	Strongly Agree	Strongly Agree	Agree	Agree	Disagree	Disagree	Agree	Strongly Agree	Disagree	Disagree
4	10010103	Strongly Agree	Strongly Agree	Agree	Agree	Disagree	Disagree	Agree	Strongly Agree	Disagree	Disagree
5	10010104	Strongly Agree	Strongly Agree	Agree	Agree	Disagree	Disagree	Agree	Strongly Agree	Disagree	Disagree
6	10010105	Strongly Agree	Strongly Agree	Agree	Agree	Disagree	Disagree	Agree	Strongly Agree	Disagree	Disagree
7	10010106	Strongly Agree	Strongly Agree	Agree	Strongly Agree	Disagree	Disagree	Agree	Strongly Agree	Disagree	Disagree
8	10010107	Strongly Agree	Strongly Agree	Agree	Agree	Disagree	Disagree	Agree	Strongly Agree	Disagree	Disagree
9	10010108	Strongly Agree	Strongly Agree	Agree	Agree	Disagree	Disagree	Agree	Strongly Agree	Disagree	Disagree
10	10010109	Agree	Strongly Agree	Agree	Agree	Disagree	Disagree	Agree	Strongly Agree	Disagree	Disagree
11	10010110	Agree	Strongly Agree	Agree	Agree	Disagree	Disagree	Strongly Agree	Strongly Agree	Disagree	Disagree
12	10010111	Agree	Strongly Agree	Agree	Agree	Disagree	Disagree	Strongly Agree	Strongly Agree	Disagree	Disagree
13	10010112	Agree	Strongly Agree	Agree	Agree	Disagree	Disagree	Strongly Agree	Strongly Agree	Disagree	Stongly Disagree
14	10010113	Agree	Strongly Agree	Agree	Agree	Disagree	Disagree	Strongly Agree	Strongly Agree	Disagree	Stongly Disagree
15	10010114	Agree	Agree	Agree	Agree	Disagree	Disagree	Strongly Agree	Strongly Agree	Disagree	Stongly Disagree
16	10010115	Agree	Strongly Agree	Agree	Disagree	Strongly Disagre	Disagree	Strongly Agree	Strongly Agree	Disagree	Disagree
17	10010116	Agree	Strongly Agree	Agree	Agree	Strongly Disagre	Disagree	Strongly Agree	Strongly Agree	Disagree	Disagree
18	10010117	Agree	Strongly Agree	Agree	Agree	Strongly Disagre	Disagree	Strongly Agree	Strongly Agree	Disagree	Disagree
19	10010118	Agree	Strongly Agree	Agree	Agree	Strongly Disagre	Disagree	Strongly Agree	Strongly Agree	Disagree	Disagree
20	10010119	Strongly Agree	Strongly Agree	Agree	Agree	Strongly Disagre	Disagree	Strongly Agree	Strongly Agree	Disagree	Disagree
21	10010120	Strongly Agree	Strongly Agree	Agree	Agree	Strongly Disagre	Disagree	Strongly Agree	Agree	Strongly Disagree	Disagree
22	10010121	Strongly Agree	Disagree	Agree	Agree	Strongly Disagre	Disagree	Strongly Agree	Agree	Strongly Disagree	Disagree
23	10010122	Strongly Agree	Strongly Disagree	Agree	Strongly Agree	Disagree	Disagree	Strongly Agree	Agree	Strongly Disagree	Disagree
24	10010123	Strongly Agree	Strongly Disagree	Strongly Agrees	Strongly Agree	Disagree	Disagree	Strongly Agree	Agree	Strongly Disagree	Disagree
25	10010124	Disagree	Strongly Disagree	Strongly Agrees	Strongly Agree	Disagree	Disagree	Strongly Agree	Agree	Strongly Disagree	Disagree
26	10010125	Disagree	Strongly Agree	Strongly Agrees	Strongly Agree	Disagree	Disagree	Strongly Agree	Agree	Strongly Disagree	Disagree
27	10010126	Disagree	Strongly Agree	Agree	Strongly Agree	Disagree	Disagree	Strongly Agree	Agree	Strongly Disagree	Disagree
28	10010127	Disagree	Strongly Agree	Agree	Agree	Disagree	Disagree	Strongly Agree	Agree	Strongly Disagree	Agree
29	10010128	Disagree	Strongly Agree	Agree	Agree	Disagree	Disagree	Strongly Agree	Agree	Strongly Disagree	Disagree
30	10010129	Disagree	Strongly Agree	Agree	Agree	Disagree	Disagree	Strongly Agree	Agree	Strongly Disagree	Disagree
31	10010130	Disagree	Strongly Agree	Agree	Agree	Agree	Disagree	Strongly Agree	Agree	Strongly Disagree	Disagree
32	10010131	Disagree	Strongly Agree	Agree	Agree	Disagree	Disagree	Strongly Agree	Agree	Disagree	Disagree
33	10010132	Strongly Agree	Strongly Agree	Agree	Agree	Disagree	Disagree	Strongly Agree	Disagree	Disagree	Disagree
34	10010133	Strongly Agree	Strongly Agree	Agree	Agree	Disagree	Disagree	Strongly Agree	Agree	Disagree	Agree
35	10010134	Strongly Agree	Strongly Agree	Agree	Disagree	Disagree	Disagree	Strongly Agree	Agree	Disagree	Disagree

Unit of Analysis

"How have the dominant themes in a corpus of 19th-century British literature changed?"

- Data: literary works (novels, poems, etc.)
- Unit of analysis: whole pieces, chapters, paragraphs, etc.

Unit of Analysis

“What are the key scientific topics debated within the scientific community between 2000–2020?”

- Data: scientific publication databases (e.g., Dimension, Web of Science, etc.)
- Unit of analysis: titles, abstracts, introductions, full texts, etc.

Unit of Analysis

The key consideration is our research question

Unit of Analysis

E.g., Barbera et al. (2019)

- Investigate whether politicians respond to people's policy interests, focused on Twitter (2013–2014)
- Run topic models (**L**atent **D**irichlet **A**llocation) on tweets from ordinary users and 500+ legislators in the U.S.
- Examine if the topics in the former at **t** predicts the latter at **t+1**

Unit of Analysis

E.g., Barbera et al. (2019)

- *“Our definition of “document” is the aggregated total of tweets sent by members of Congress each day”*
- *“Our conceptualization of each day’s tweets as the political agenda that each party within each legislative chamber is trying to push for that specific day”*
- *“Conducting an analysis at the tweet level is complex, given its very limited length”*

Unit of Analysis

E.g., Hammer et al. (2019)

THREAT: A Large Annotated Corpus for Detection of Violent Threats

1st Hugo L. Hammer
Department of Computer Science
OsloMet – Oslo Metropolitan University
Oslo, Norway
hugo.hammer@oslomet.no

2nd Michael A. Riegler
Simula Metropolitan Center for Digital Engineering
Oslo, Norway

3rd Lilja Øvrelid
4th Erik Velldal
Department of Informatics
University of Oslo
Oslo, Norway

***Abstract*—Understanding, detecting, moderating and in extreme cases deleting hateful comments in online discussions and social media are well-known challenges. In this paper we present a dataset consisting of a total of around 30 000 sentences from around 10 000 YouTube comments. Each sentence is manually annotated as either being a violent threat or not. Violent threats is the most extreme form of hateful communication and is of particular importance from an online radicalization and national security perspective. This is the first publicly available dataset with such an annotation. The dataset can further be useful to develop automatic moderation tools or may even be useful from a social science perspective for analyzing the characteristics of online threats and how hateful discussions evolve.**

***Index Terms*—national security, publicly available dataset, social media, threat detection, violent threats**

commenting [1], [2], [5], [11], [12], [15], [16], [26]. The methods are mainly based on machine learning and thus require annotated text to learn to separate abominable from harmless online behaviour. Unfortunately, neither of these studies have made the accompanied datasets publicly available. In fact, we are not aware of any publicly available datasets that can be used to develop automatic threat detection.

As a contribution to solve these challenges and to make it possible to perform open and important research on making cyberspace more secure for people we present a large dataset of YouTube comments, where each sentence (manually segmented) is annotated as either being a threat of violence or not.

Unit of Analysis

E.g., Hammer et al. (2019)

- Use supervised learning to detect threatening speech on YouTube comments (i.e., text classification)
- Comments on YouTube videos are split into individual sentences

Tokenization

Breaking up a text into discrete components

- Tokenization is a form of segmentation (= word segmentation)
- Token: each individual component in the document
  - Possibly including numbers, punctuation, or other symbols

Tokenization

“To be or not to be, that is the question”

$\longrightarrow$ “To”, “be”, “or”, “not”, “to”, “be”, “that”, “is”, “the”, “question”

Tokenization

Types

- Each token is of a particular “type”
- The set of types is the vocabulary (often denoted as $|V|$)
- “To be or not to be, that is the question”

  $\longrightarrow$ “to” “be” “or” “not” “that” “is” “the”, “question” ($|V| = 8$)

Tokenization

`“Let us explore tokenization.”`

- Word-level: [“Let”, “us”, “explore”, “tokenization.”]
- Subword-level: [“Let”, “us”, “explore”, “token”, “ization.”]
- Character-level: [“L”, “e”, “t”, “u”, “s”, “e”, “x”, “p”, “l”, “o”, “r”, “e”, “t”, “o”, “k”, “e”, “n”, “i”, “z”, “a”, “t”, “i”, “o”, “n”, “.”]

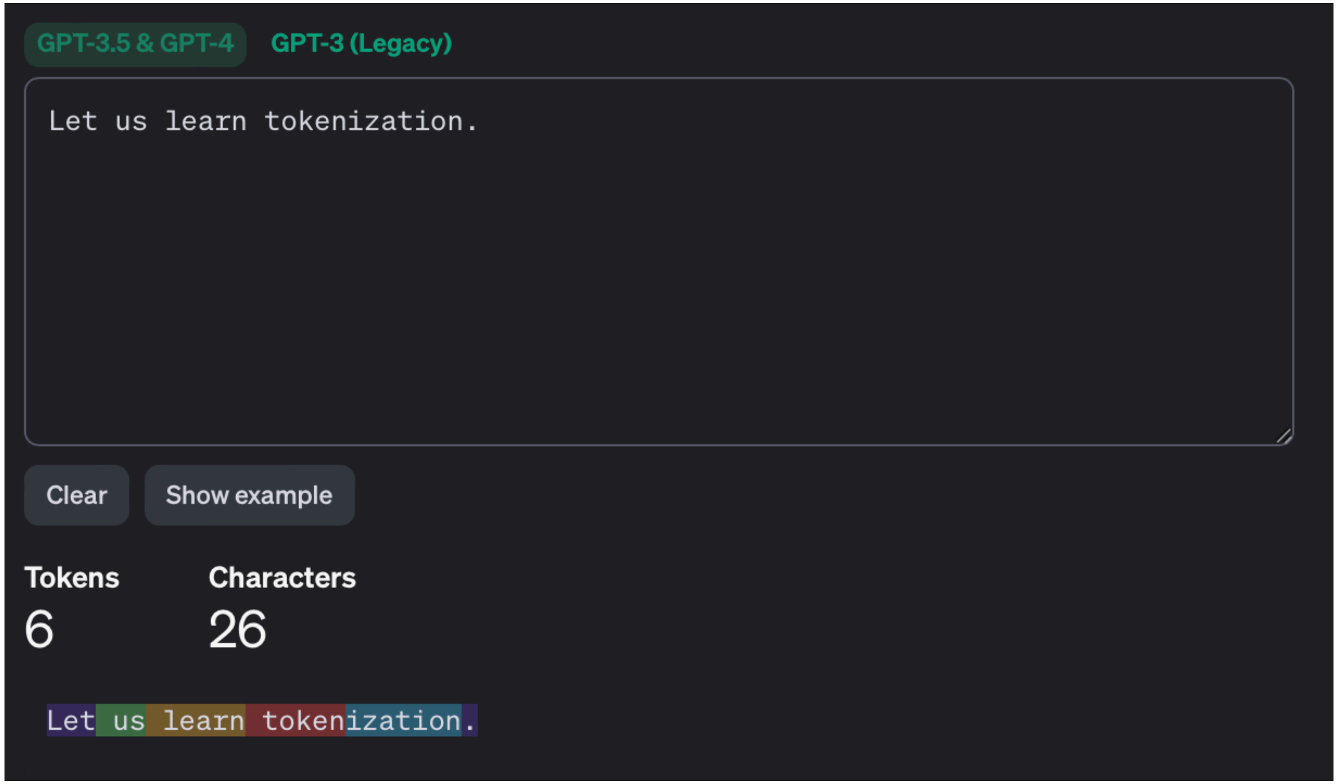
Tokenization

Levels of tokenization

- Words: most common pre-LLM
- Subwords: now prevalent in neural NLP / LLM
  - Handling of OOV (out-of-vocabulary) words
    - E.g, if the model has `vi` (as in virus) and `rologist` (as in neurologist, urologist, etc.), it can handle `virologist`
  - Reduced vocabulary / more efficiency (again, consider "tokenization")
  - Common apporaches include Byte Pair Encoding (BPE) for GPT, WordPiece for BERT
- Character: no meaning (although computationally very efficient)
  - E.g., what would be the vocabulary size?
- Sentences: too many types

Tokenization

Subword tokenization in GPTs



Tokenization

Tokenization at the word level

Tokenization

Tokenization at the sub-word level

- Very common in LLMs
- Tokenizers are built (trained) as a separate process before model training
- After a tokenizer is initialized and trained, it is then used in the training process of its associated (L)LMs
  \(\rightarrow\) The model is “locked” to its tokenizer
  \(\rightarrow\) (L)LMs are linked with their tokenizers

Tokenization

Tokenization at the sub-word level (cont'd)

- Methods: while there are various methods, they all aim to optimize an efficient set of tokens to represent a texts data set
  - E.g., Byte Pair Encoding or WordPiece
- Special tokens: used to indicate specific roles or structures
  - E.g., `[CLS]` (BERT) or `<s>` (GPT) to mark the beginning of input/output
  - E.g., `[SEP]` (BERT) or `</s>` (GPT) to separate sentences or mark the end of input/output
- Vocabulary size: it should be decided how many tokens to keep in the tokenizer's vocablary

Tokenization

n-grams

- A sequence of *n* adjacent tokens
- Unigrams, bigrams, trigrams, etc.
- Why would we need multi-grams?
  - E.g., “White House”, “look after”, “take care of”, etc.

Tokenization

n-grams

- Be aware of the computational cost
  - Consider the number of all consecutive sets of two words in the corpus
- Alternatively, we can compile a list of particular bi-grams or tri-grams

Tokenization

n-grams

- With modern (L)LMs, we no longer need to explicitly generate n-grams most of the time
- However, the idea behind n-grams remains relevant
- The purpose of *n*-grams is often achieved implicitly through subword tokenization and attention mechanisms in modern transformer-based models
  - This inclues capturing local word patterns, collocations, and context,
  - E.g., subword tokenization involves common patterns (e.g., `artificial intelligence` or `New York`)

Segmenting Sentences/paragraphs

Sentence segmentation

- Useful cues: periods, question marks, or exclamation marks
- Prone to errors (the example of `"."`)
  - Abbreviations and initials: “Ph.D.”, “J.K. Rowling”, etc.
  - Decimal numbers: “3.14”
  - Websites and email addresses: “www.kaist.ac.kr”) and email addresses
  - Quotations within a sentence: “He said, ‘Stop.’ Then he left.”
- Rule-based/deterministic or ML-based approaches (part of `NLTK` and `spaCy`)

Segmenting Sentences/paragraphs

Paragraph segmentation

- Not as commonly addressed
- There are useful cues
  - Newline characters (`\n`) or double newline characters (`\n\n`)
  - Indentations (e.g., `\t`)
  - With HTLM documents, we could potentially use tags (e.g., `<p>`) to parse different parts of the document (not necessarily paragraphs though)
- Fewer specialized libraries or algorithms in Python

Text Normalization

A set of approaches to reducing complexity in text (a.k.a. pre-processing)

- The output from tokenization will contain too many words
- With normalization, vocabulary size can be reduced (computationally more efficient)
- It can enhance many downstream tasks
  - E.g., topic modeling (player, players, playing, etc.)
  - E.g., information retrieval: finding a pattern in a corpus (e.g., Penny, Pennies, penny, pennies, etc.)

Text Normalization

We will discuss five approaches

- Lowercasing
- Removing punctuation
- Removing stop words
- Lemmatization/stemming
- Filtering by frequency

Text Normalization

Lowercasing

- We often replace all capital letter with lowercase letters
- It is assumed that there is no (semantic) difference
- Is it?

Text Normalization

Lowercasing

- Compare “NOW” and “now” in terms of sentiment
- Capital letters also signal the start of of a sentence
- Proper nouns (May vs. may. US vs. us)

Text Normalization

Removing punctuation

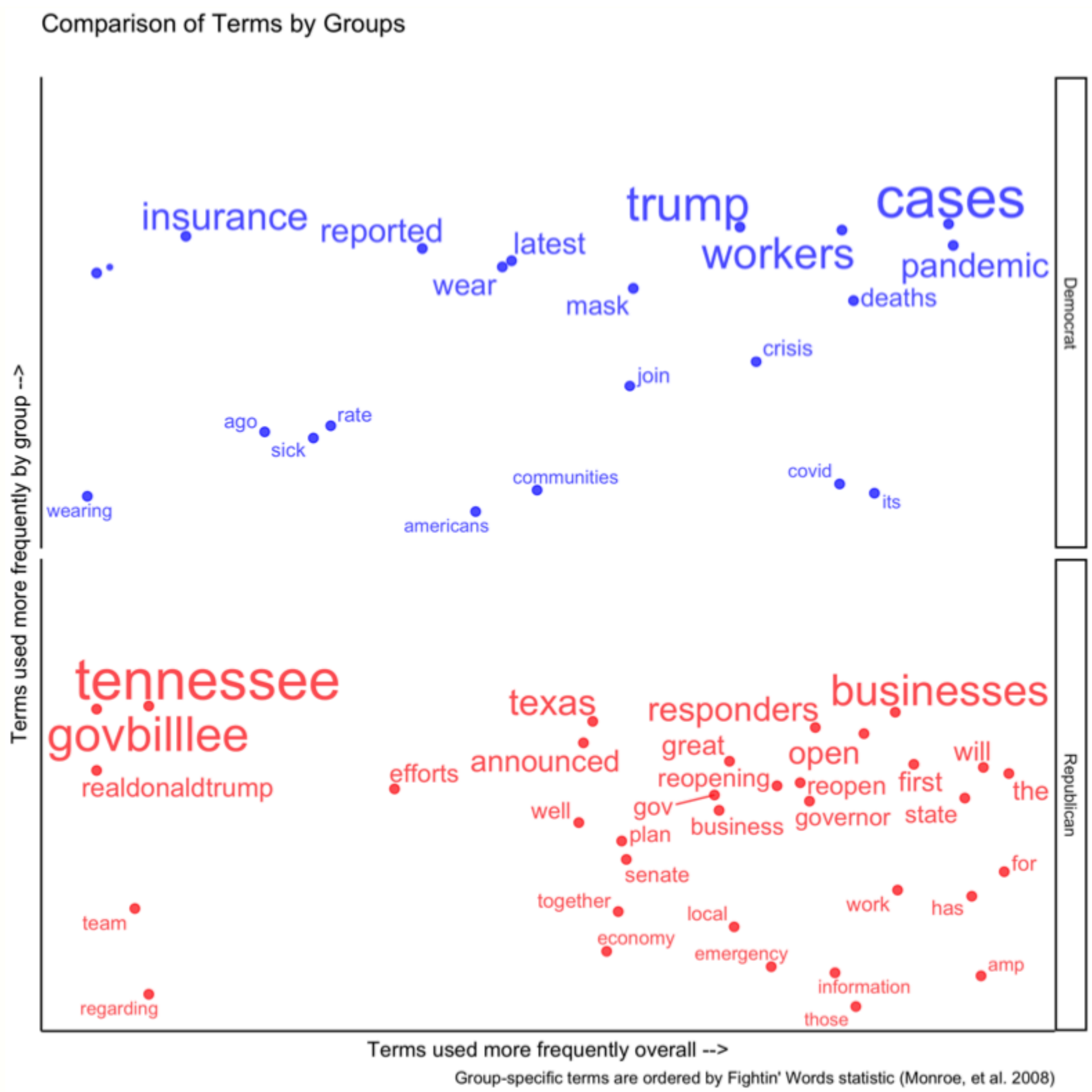
- Period (`.`), comma (`,`), apostrophe (`'`), quotation (`""`), question (`?`), exclamation (`!`), dash (`–`), ellipsis (`...`), colon (`:`), semicolon (`;`), etc.
- In many cases, these are (considered) unimportant
- Are they?

Text Normalization

Removing punctuation

- Punctuation carries important information
  - Exclamation mark (`!!!`), hashtags (`#metoo`), emojis (`:)`), etc.
- Punctuation itself can be of interest (studying writing styles)

Text Normalization



Text Normalization

Removing stop words

- Common words used across documents that do not give much information
- E.g., “and”, “the”, or “that”

Text Normalization

Removing stop words can spare much computational power

- C.f., Heaps' Law
- However, under what circumstances are these words *not* stop words? For instance, consider the case of `the`.

Text Normalization

Lemmatization

- Lemma: the base form
  - E.g., “run”
- Wordform: various forms derived the lemma
  - E.g., “runs”, “ran”, “running”
- Lemmatizatoin is the process of mapping words to their lemma

Text Normalization

Lemmatization

- Not always straightforward
  - Irregular variations E.g., “see-saw-seen”
  - Same token but different lemmas
    - E.g., he is “writing” an email vs. a nice piece of “writing”
- Necessitates a dictionary and POS (**p**art **o**f **s**peech) tagging

Text Normalization

Stemming is a popular approximation to lemmatization

- Simply discards the end of a word
  - E.g., family: famili
- Errors
  - E.g., “leav” for both “leaves” (as in “He leaves the room”) and “leaves” (as in parts of a plant)
- Various algorithms: *Porter*, *Lancaster*, etc.

Text Normalization

Filtering by frequency

- Too (in)frequent words across documents
  - E.g., stop words
- Can be filtered by the minimum/maximum document frequencies
  - Word that appear in fewer/more than *n*% of documents
- The rationale
  - Discriminatory power
  - Computational savings

Text Normalization

(How) Should we normalize?

- Difficult to know its consequences *a priori*
- Before analysis: carefully think about the pros and cos in each of the steps
- After analysis: conduct robustness check

Text Normalization

Normalization with LLMs

- With LLMs, traditional normalization steps are less relevant, especially due to the widespread adoption of subword-level tokenization (BPE, WordPiece, etc.)
- However, they are still relevant depending on the context

Text Normalization

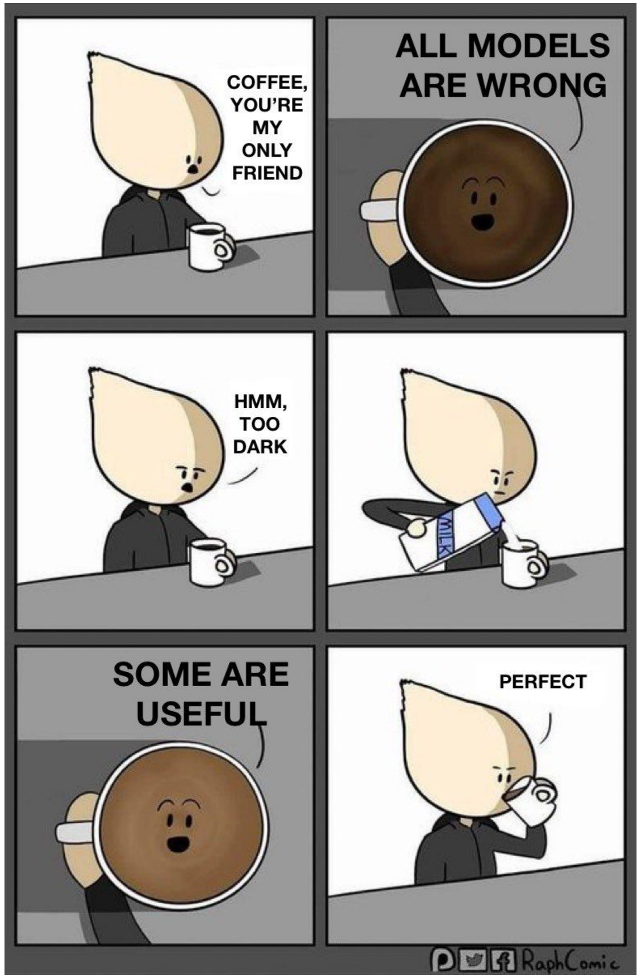
Normalization with LLMs (cont’d)

- Lower-casing: different models deal with lower-casing differently
  - BERT-base-uncased vs. BERT-cased, GPT models, etc.
- Punctuation and stopwords: LLMs can handle them effectivrly by treating them as meaningful tokens
- Lemmatization/stemming: subword tokenization already handles word variations
  - See [AG] pp. 47–55 for various approaches
- White spaces: white space handling can matter where they represent structure
  - E.g., four white spaces as a single token representing an indentation should work better for code (see CodeBERT)

Text Representation

Text representation as a model

- Text representation abstracts away from reality (actual texts)
- It serves as a simplified model of language and meaning
- This applies to LLMs too
- “All models are wrong, some are useful” (George Box, 1976)



Text Representation

Levels of text representation

- Word representations
  - Static embeddings: Word2Vec, GloVe, FastText (**Week 6**)
    - The same embeddings for the `bank` in `river bank` and `bank account`
  - Contextual embeddings: neural embeddings from transformer models (BERT, GPT) (**Week 12**)
    - The embedding for the `bank` differs across contexts

Text Representation

Levels of text representation (cont’d)

- Representations beyond the word-level (sentences, paragraphs, or documents)
  - Lexcial approaches
    - Bag of Words (BoW): basic word frequency representation
    - TF-IDF (Term-Frequency Inverse Document Frequency): weighting of words based on their importance to documents
  - Transformer-based neural embeddings can be used for representing entire sentences, paragraphs, or documents
    - There are models specifically tailored for sentence-level representations too (e.g., S-BERT, Universal Sentence Encoder)

The Bag of Words (BoW) model

The most basic text representation model

- A text is represented as a set of words that appear in it

The Bag of Words (BoW) model

Document-Feature Matrix (or Document-Term Matrix)

- Columns record features/terms (all types or $|V|$)
- Rows record documents
- Cells can be binary vectors or count vectors

The Bag of Words (BoW) model

An example corpus

- Doc 1: “The clever fox cleverly jumps over the lazy dog, showcasing its cleverness.”
- Doc 2: “Magic and mysteries mingle in the wizard’s daily musings, revealing mysteries unknown.”
- Doc 3: “Sunny days bring sunshine and sunsets, making sunny parks the best for sunny strolls.”

The Bag of Words (BoW) model

An example DFM

Document	clever	jumps	lazy	dog	magic	mysteries	…
Doc 1	3	1	1	1	0	0	…
Doc 2	0	0	0	0	1	2	…
Doc 3	0	0	0	0	0	0	…

The Bag of Words (BoW) model

An example DFM

Document	`clever`	jumps	lazy	dog	magic	mysteries	...
Doc 1	3	1	1	1	0	0	...
Doc 2	0	0	0	0	1	2	...
Doc 3	0	0	0	0	0	0	...

The Bag of Words (BoW) model

An example corpus

- Doc 1: “The `clever` fox `cleverly` jumps over the lazy dog, showcasing its `cleverness`.”
- Doc 2: “Magic and mysteries mingle in the wizard’s daily musings, revealing mysteries unknown.”
- Doc 3: “Sunny days bring sunshine and sunsets, making sunny parks the best for sunny strolls.”

The Vector Space Model

What is the vector space model?

- Each row (representing a text) in a DFM is a vector (an array of numbers) in a high-dimensional space
- The size of the dimension (the number of columns) is $|V|$
- Originates from IR (**I**nformation **R**etrieval)
  - See Turney and Pantel (2010) for details

Comparing Texts

With some form of DFM, we are ready to compare different documents

- "Similar" can mean different things
  - Sentiments, stances, themes, etc.
- There is no "correct" notion of similarity
- Yet there are metrics that are more or less effective across contexts

Cosine Similarity

We have two vectors (representing two documents), $\vec{A}$ and $\vec{B}$:

$$\vec{A} = [a_1, a_2, \ldots, a_n]$$

$$\vec{B} = [b_1, b_2, \ldots, b_n]$$

The inner product:

$$\vec{A} \cdot \vec{B} = (a_1 \times b_1) + (a_2 \times b_2) + \ldots + (a_n \times b_n)$$

Cosine Similarity

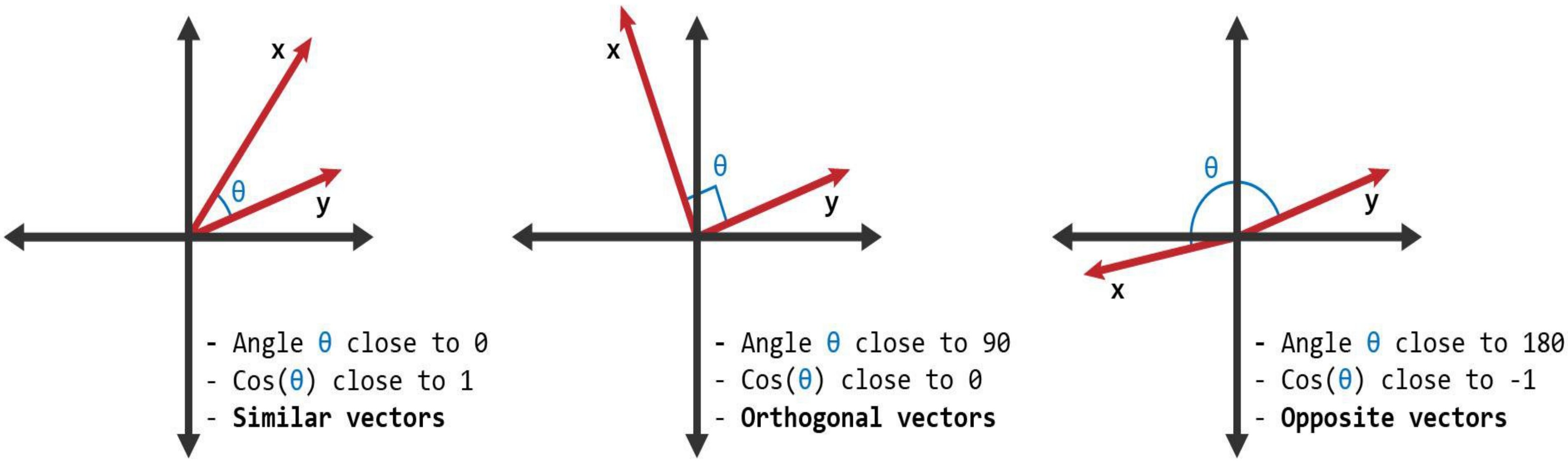
Cosine similarity between vectors $\vec{A}$ and $\vec{B}$ is given by:

$$\text{Cosine Similarity} (\vec{A}, \vec{B}) = \frac{\vec{A} \cdot \vec{B}}{\|\vec{A}\| \|\vec{B}\|}$$

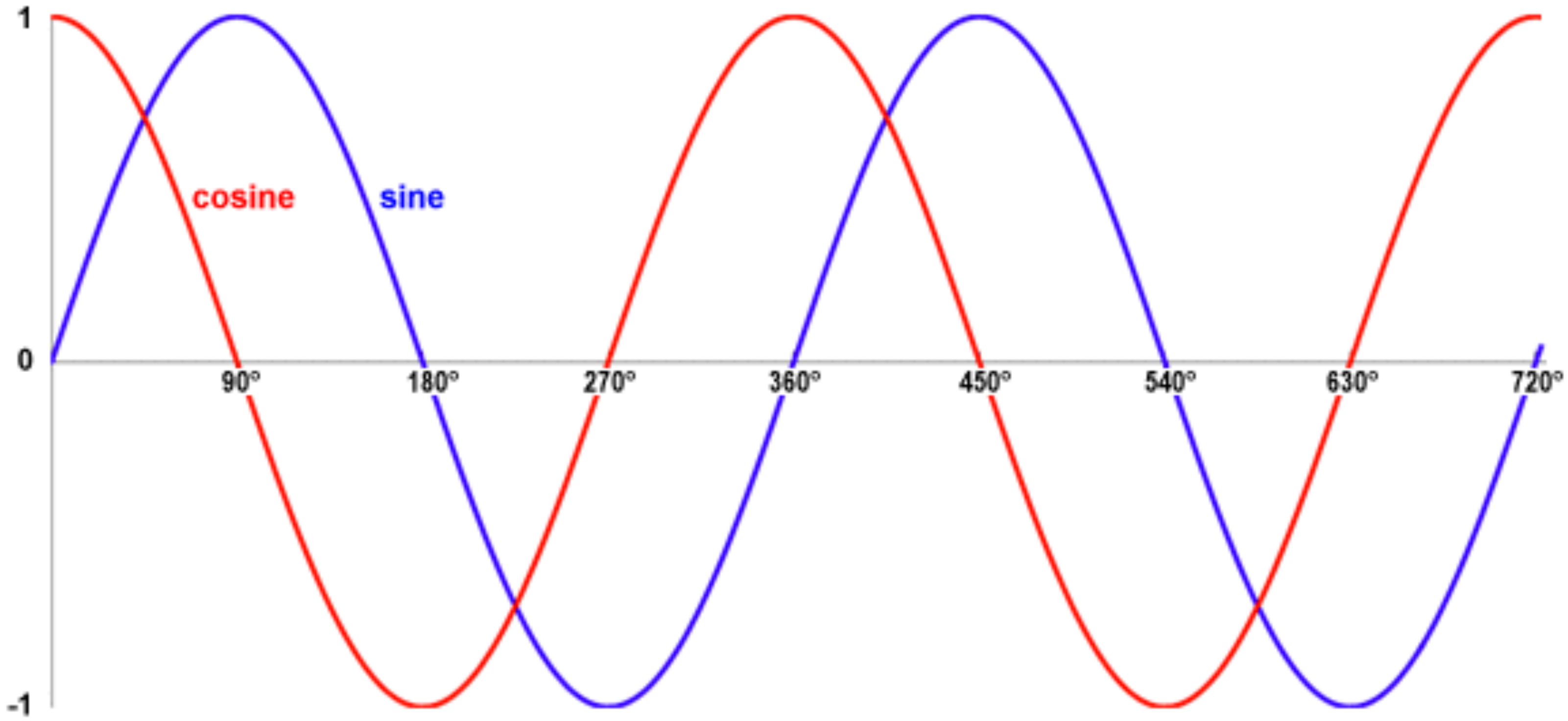
$\vec{A} \cdot \vec{B}$ is the inner product, and $\|\vec{A}\|$ and $\|\vec{B}\|$ are defined as

$$\|\vec{A}\| = \sqrt{a_1^2 + a_2^2 + \ldots + a_n^2} \|\vec{B}\| = \sqrt{b_1^2 + b_2^2 + \ldots + b_n^2}$$

Cosine Similarity



Cosine Similarity



TF-IDF Weighting

TF (Term Frequency) - IDF (Inverse Document Frequency)

- Count vectors consider the frequencies of words
- However, some words are too frequent across different documents
  - E.g., *the*, *a*, *an*, etc.
- We want to weight how unique a word to a document

TF-IDF Weighting

TF-IDF is a numerical statistic that reflects *how important a word is to a document* in a corpus.

TF-IDF Weighting

The **TF-IDF** value is obtained by multiplying **TF** (**T**erm **F**requency) and **IDF** (**I**nverse **D**ocument **F**requency) for a term in a document, highlighting the importance of rare terms

$$\text{TF-IDF}(t, d, D) = \text{TF}(t, d) \times \text{IDF}(t, D)$$

TF-IDF Weighting

Term Frequency

- Reflects how frequently a term occurs in a document, normalized by the document length

$$ \text{TF}(t, d) = \frac{\text{Number of times term } t \text{ appears in document } d}{\text{Total number of terms in document } d} $$

TF-IDF Weighting

Inverse Document Frequency

- Scales down terms that occur very frequently across the corpus and are less informative

$$ \text{IDF}(t, D) = \log\left(\frac{\text{Total number of documents } D}{\text{Number of documents with term } t \text{ in it} + 1}\right) $$

TF-IDF Weighting

Many versions of TF-IDF: link

Count Vectors Vs. TF-IDF Vectors

Count Vectors

Term	can	you	fly	sleep
‘can you fly’	1	1	1	0
‘can you sleep’	1	1	0	1

TF-IDF Vectors

Term	can	you	fly	sleep
‘can you fly’	0.5	0.5	0.7	0
‘can you sleep’	0.5	0.5	0	0.7

Summary

The process of transforming raw texts into numbers involve a number of important decisions

- Segmentation
- Normalization
- Representation

$\to$ It is worth thinking ahead of and reviewing the potential consequences

Guided Coding

Normalization, representation, and comparison in Python
(Link)