Text-as-data for Social Science I

KAPS Summer Methods Workshop

Taegyoon Kim | KAIST

Jul 11, 2024

Overview

Instructor

Taegyoon Kim | Webpage

- Biography
  - Assistant Professor at School of Digital Humanities and Computational Social Sciences, KAIST (2023. 9 -)
  - Postdoctoral Fellow at Kellogg School of Management, Northwestern University (2022. 9 - 2023. 8)
  - Ph.D. in Political Science and Social Data Analytics, Pennsylvania State University (2017. 8 - 2022. 8)
- Research areas
  - Digital media and democracy
  - Scientific expertise in policy and politicization of science
  - Computational methods (e.g., text-as-data)
- Student recruitment
  - M.S. or Ph.D. degree in Computational Social Science (fully funded)
  - Through both School of Digital Humanities and Computational Social Sciences and Graduate School of Data Science

Broad exploration of various text-as-data methods

What we aim and do not today

- Understanding of when and how to use key methods
- Intuitive understanding of the workings of methods
  - With references to materials for a full coverage of the mathematics/statistics under the hood
- Introduction to programming (mainly Python, a bit of R)
  - Serving as a starting point for customized/advanced applications
  - Elementary level of programming knowledge is assumed
- Unfortunately little time for discussing applied research
- More advanced methods, particularly ones based on neural networks, will not be covered
  - E.g., fine-tuning language models, neural embeddings and clustering (for topic models), using LLMs for measurement
- For more comprehensive/in-depth course-length learning, reference NLP for Humanities and Social Sciences

Representing Texts

Before We Start

Ideally

- We should have discussed the key principles associated with how to selecting texts for research
- See the Week 2 of the NLP for DHCSS for detailed discussions of approaches to choosing texts for research, potential biases, etc.

Representing Texts: themes

Things to be covered

- Unit of analysis
- Tokenization (segmentation)
- Text normalization (= cleaning)
- BoW / vector space models
- Cosine similarity
- TF-IDF weighting
- Guided coding
  - String manipulation and regex (Python)
  - Segmentation, normalization, representation (Python)
  - Text pre-processing in Korean (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

	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

The key consideration is our research question

Unit of Analysis

Barbera et al. (2019)

- Topic models (**L**atent **D**irichlet **A**llocation) on tweets from ordinary users and 500+ legislators in the U.S.
- See if the topics in the former at **t** predicts the latter at **t+1**

Unit of Analysis

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

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

Hammer et al. (2019)

- Supervised learning to detect threatening speech on YouTube comments
- Comments on YouTube videos are split into individual sentences

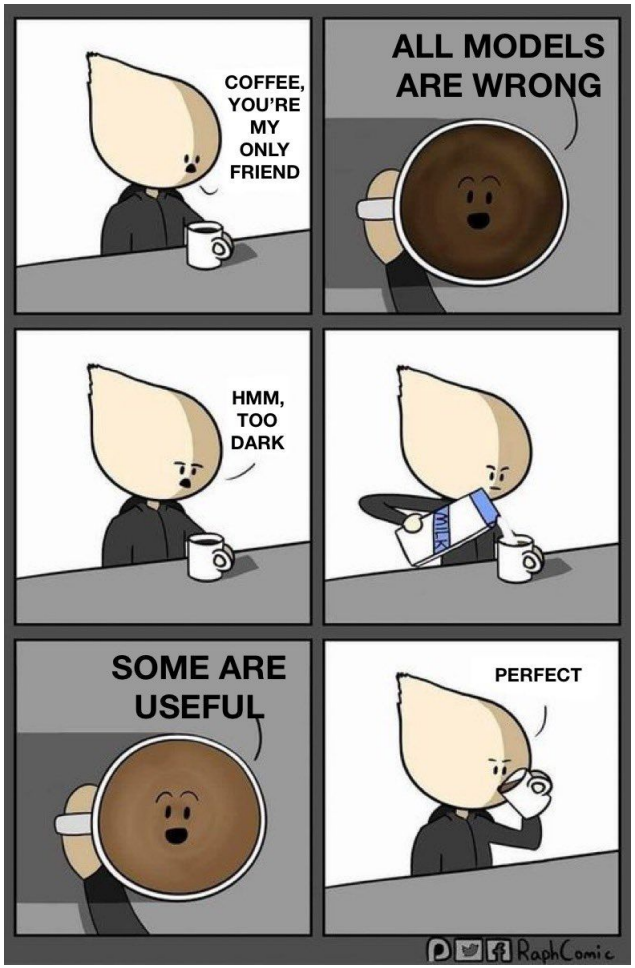
Unit of Analysis

Other considerations

- Rarity of quantity of interest
- P-value (!?)
  - E.g., legislators' tweets aggregated at the weekly level vs. monthly level

Models of Text Representation

“All models are wrong, some are useful” (George Box, 1976)



Tokenization

Breaking up a text into discrete words

- Tokenization is a form of segmentation (= word segmentation)
- Token: each individual “word” in the document
  - Possibly including numbers, punctuation, or other symbols
- Tokenization: the process of splitting a document into its constituent words

Tokenization

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

⟶ “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 learn tokenization.”

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

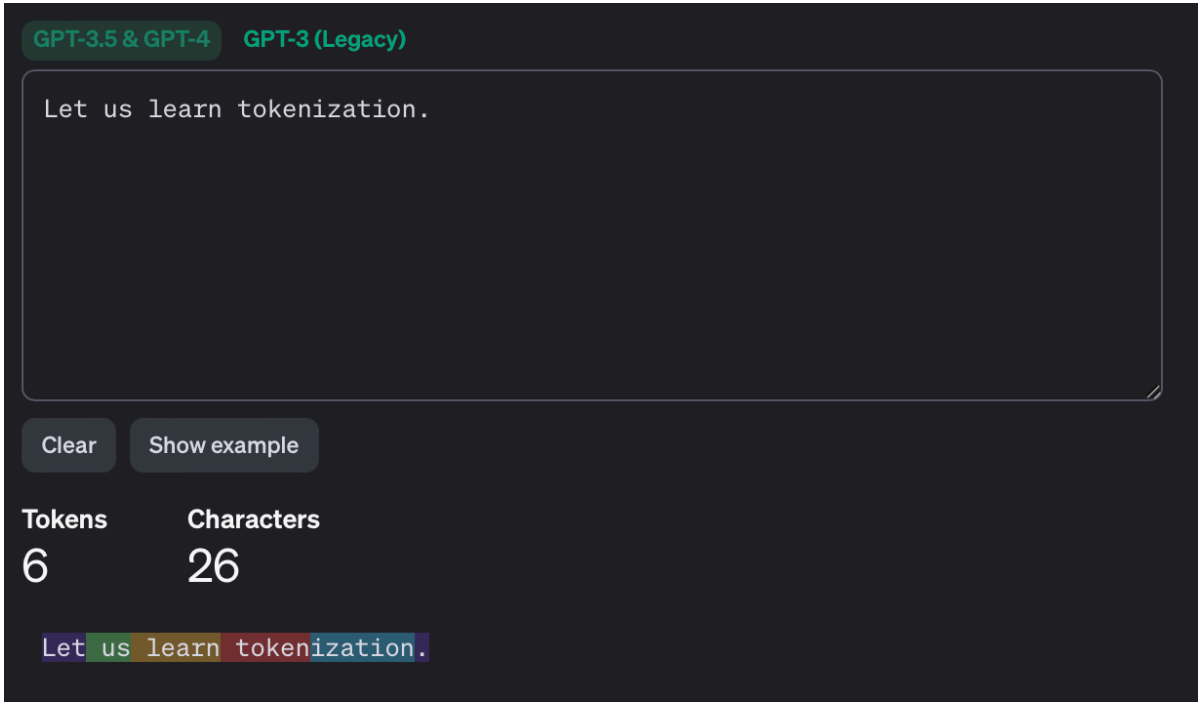
Tokenization

Why word-level?

- Words: most common for many downstream analyses
  - Word embeddings, topic models, etc.
- Subwords
  - Now prevalent in neural NLP
  - Handling of OOV (out-of-vocabulary) words
  - More efficiency (consider “tokenization”)
  - E.g., Byte Pair Encoding (BPE), WordPiece
- Character: no meaning (although computationally very efficient)

Tokenization

Subword tokenization in recent GPTs



Tokenization

How to tokenize?

- In English (and many other languages, including Korean), we can rely heavily on white space
  - Many algorithms build not only on white space but also on various patterns
  - E.g., appstrophies: [“don”, “’”, “t”] vs. [“do”, “n’t”]
  - E.g., punctuations: [‘vehicle?’] vs. [‘vehicle’, ‘?’]
- Tools include `NLTK`, `spaCy`, `Keras`, etc.
- In some languages, words cannot be separated deterministically, and they need models (e.g., Chinese, Japanese, etc.)

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

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
- Few specialized libraries or algorithms in Python
- Useful cues: newline characters (`\n`) or double newline characters (`\n\n`)

Text Normalization

A set of approaches to reducing complexity in text

- The output from tokenization will contain too many words
- Putting words in a standard form can be useful for information retrieval
  - Findings 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 (`<3`, `:)`, `^^`, `-_-;`), etc.
- Punctuation itself can be of interest (studying writing styles)

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?

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
- The rationale
  - Discriminatory power
  - Computational savings

(How) Should We Normalize?

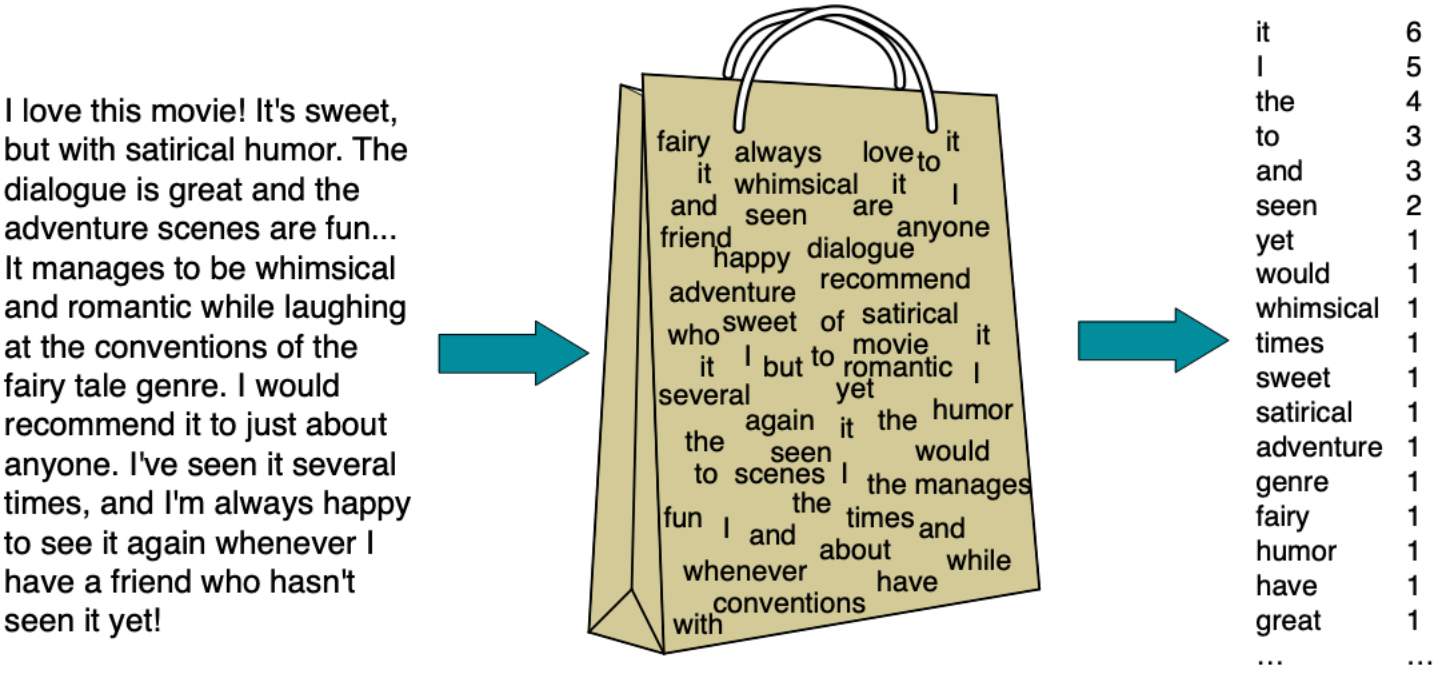
Difficult to know its consequences *a priori*

- Before analysis: carefully think about the pros and cos in each of the steps
- After analysis: conduct robustness check

The Bag of Words (BoW) model

The most common 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 of 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

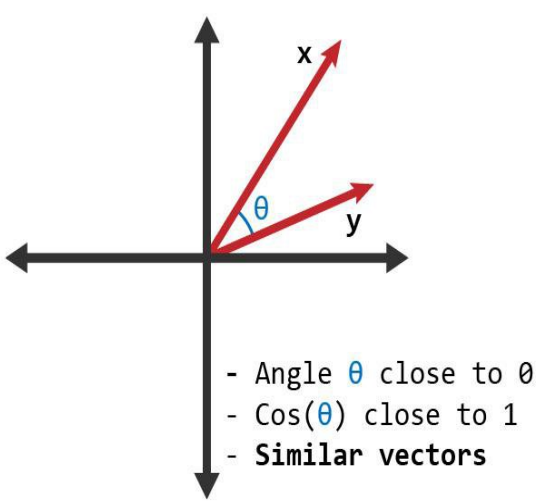
Cosine similarity between vectors $\vec{A}$ and $\vec{B}$ is given by:

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

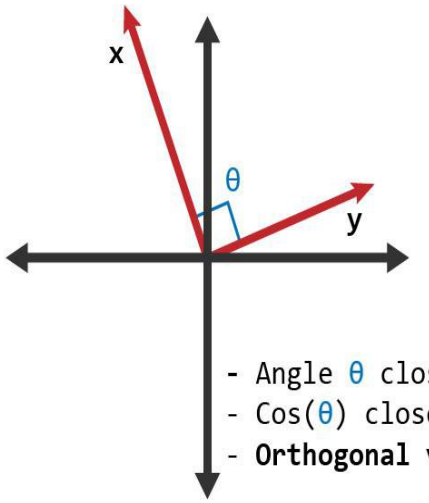
$\vec{A} \cdot \vec{B}$ is the inner product, and $\|\vec{A}\|$ and $\|\vec{B}\|$ are defined as

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

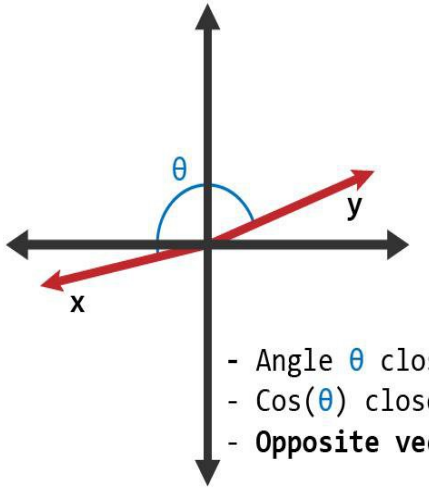
Cosine Similarity



- Angle θ close to 0
- Cos(θ) close to 1
- **Similar vectors**

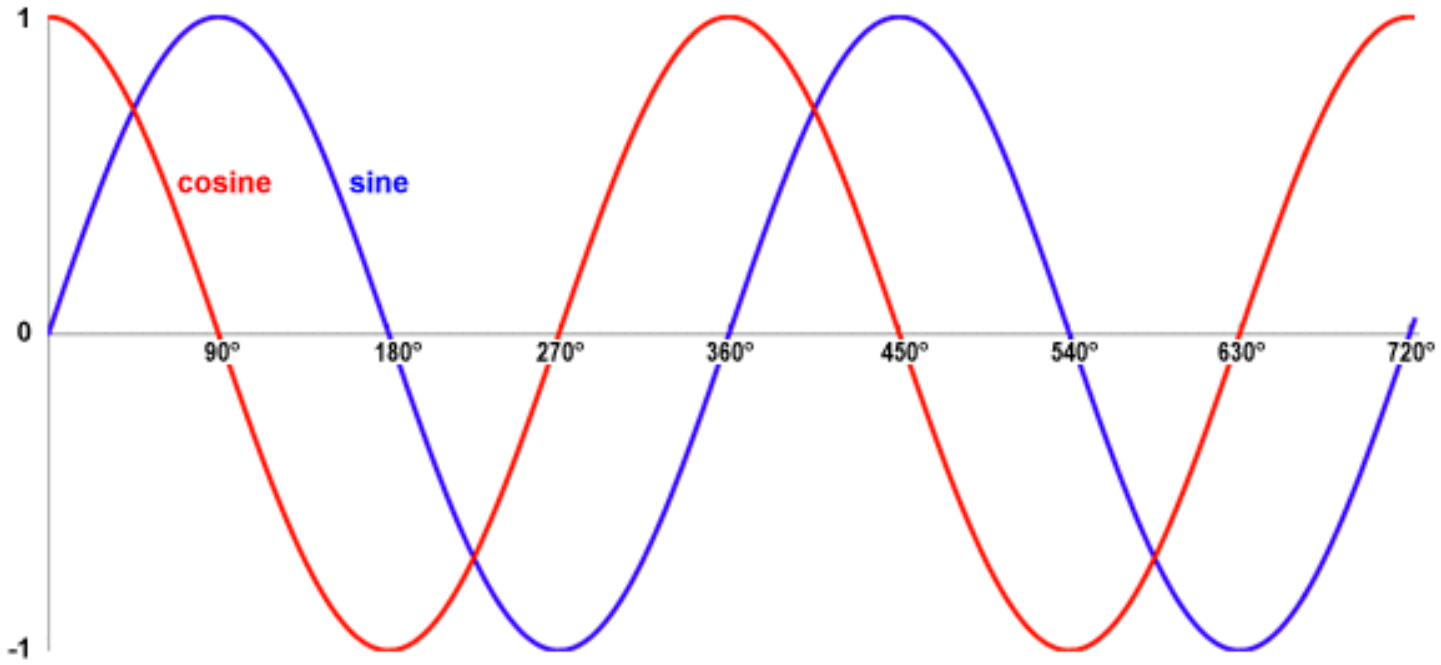


- Angle θ close to 90
- Cos(θ) close to 0
- **Orthogonal vectors**



- Angle θ close to 180
- Cos(θ) close to -1
- **Opposite vectors**

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

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

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 (e.g., segmentation, normalization, representation, etc)

→ Thus it is worth thinking ahead of and reviewing the potential consequences

Guided Coding

String manipulation and regex in Python (Link)

Normalization, representation, and comparison in Python (Link)

Text pre-processing in Korean (Link)

Supervised Learning

Supervised Learning: themes

Things to be covered

- Overview of supervised learning
- Step 1: Building a labeled data set
- Step 2: Extracting features
- Step 3: Selecting and training model(s)
- Step 4: Evaluating performance
- Guided labeling: text classification with movie reviews data in Python

Supervised Learning

We will focus on (text) classification with supervised learning

- Goal
  - To classify documents into pre-defined categories
  - E.g., sentiment of comments, stance on policy issues, topic of news articles, etc.
- We need
  - Human-labeled data set
  - Model (algorithm) that maps documents (i.e., their features) to labels
  - Evaluation approaches
    - Performance metrics, cross-validation, etc.

Supervised Learning

Supervised vs. Unsupervised

	Supervised	**Unsupervised**
Objective	Trained on a labeled data to learn a mapping from input to output	Find patterns or structures within data without labels
Outcome	Pre-defined categories	Not quite pre-defined
Model evaluation	Explicit metrics such as accuracy, precision, recall, or R^2	Can involve qualitative assessment
Examples	Classification/regression for texts	Topic models

Supervised Learning

Regression vs. Classification

- Regression
  - The outcome of interest is continuous or ordered (beyond binary)
  - E.g., OLS regression (+ non-linear regression algorithms such as random forest regression)
- Classification
  - The outcome is a value in an unordered set (i.e. categories)
- The two approaches share the broad principles of supervise leaning and can be adapted

Supervised Learning vs. Dictionary Methods

Limitations of dictionary methods

- Lack of learning (as in the the name machine “learning”)

- (Largely) ignores context
  - Polysemy, co-occurrences/interactions, etc.
  - Interactions are effectively modeled in random forest, deep neural networks, and large language models

→ Therefore, (generally) suboptimal performance

Overview of Process

Broad process

- Step 1: build a labeled data set
- Step 2: extract features
- Step 3: select and train model(s)
- Step 4: evaluate performance

Overview of Process

Step 1: build a labeled data set

- Documents with human-annotated labels (a.k.a. ground-truth) : C
- Randomly split into a training set and a test set: $C = C_{train} + C_{test}$
- E.g., identifying YouTube comments containing hate speech
  - C: 10,000 comments labeled for the presence of hate speech
  - $C_{training}$: 8,000 comments for training
  - C_{test}: 2,000 comments for test

Overview of Process

Step 1: build a labeled data set

Doc number	Text	y
1	This is great!	0
2	%@% ***k off!	1
…		
9999	This is sick	0
10000	Love BTS <3	0

Overview of Process

Step 2: extract features

- Generate X_{train} (feature matrix) from C_{train} (train set)
- E.g., count vectors, TF-IDF, or embeddings

Index	Token 1	Token 2	…	Token V-1	Token V
1	3	1.4	…	1.7	6
2	-0.8	6.4	…	5.7	-1.6
…					
7999	-2.8	0.9	…	3.3	-0.6
8000	3.7	1.4	…	5.7	-5.8

Overview of Process

Step 3: select/train model(s)

Index	y	Token 1	Token 2	…	Token V-1	Token V
1	0	3	1.4	…	1.7	6
2	1	-0.8	6.4	…	5.7	-1.6
…						
7999	0	-2.8	0.9	…	3.3	-0.6
8000	0	3.7	1.4	…	5.7	-5.8

Overview of Process

Step 3: select/train model(s)

- Choose a model F (e.g., logistic regression) and learn model parameters β (e.g., an array of coefficients)
  - The model provides a mapping between X_{train} and y_{train}
- Loss (cost) function: measures how much model predictions ($\hat{y}_{train}$) differ from the true labels (y_{train})
  - β is estimated in a way that minimizes the difference
  - $\hat{y}_{train} = F(\hat{\beta} * X_{train})$
- As a result, we get a classifier: $\hat{y} = F(\hat{\beta} * X)$

Overview of Process

Step 3: select/train model(s)

Index	y	$\hat{y}$	Token 1	Token 2	…	Token V-1	Token V
1	0	0	3	1.4	…	1.7	6
2	1	1	-0.8	6.4	…	5.7	-1.6
…							
7999	0	0	-2.8	0.9	…	3.3	-0.6
8000	0	0	3.7	1.4	…	5.7	-5.8

Overview of Process

Step 4: evaluate performance

- We held out another labeled set C_{test} (n = 2,000) (**why?**)
- Use the classifier $F(\hat{\beta} * X)$ to generate predictions $\hat{y}_{test}$
- Compare the predictions $\hat{y}_{test}$ and the true labels y_{test}
- Performance metrics include accuracy, precision, recall, etc.
- *(Then use the classifier for unlabeled data)*

Overview of Process

Step 4: evaluate performance

Index	**y**	$\hat{y}$	**Token 1**	**Token 2**	**…**	**Token V-1**	**Token V**
1	1	1	3.12	1.99	…	5.77	0.36
2	1	0	-0.8	1.14	…	9.71	-1.66
…							
1999	0	0	-2.11	0.95	…	1.23	-0.62
2000	0	0	3.71	1.48	…	1.7	-5.84

Bias, Variance, and Overfitting

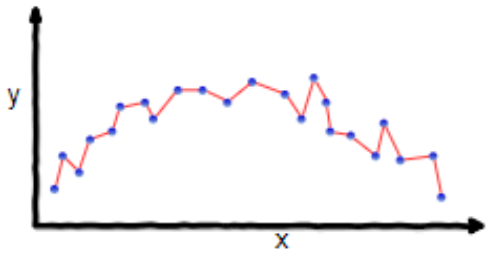
Bias

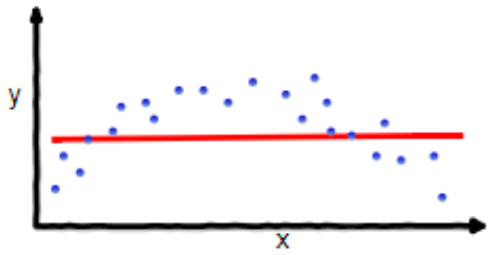
- The degree to which the model's predictions deviate from the true labels in a systematic manner
- A model with high bias make predictions that are consistently off-target

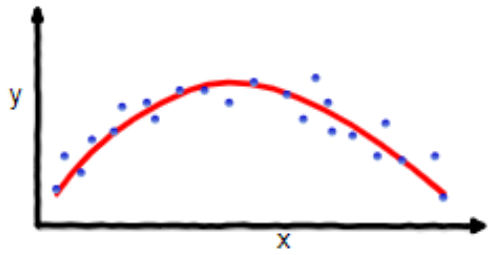
Variance

- The degree to which the model generalizes to different data
- High variance means low generalizability

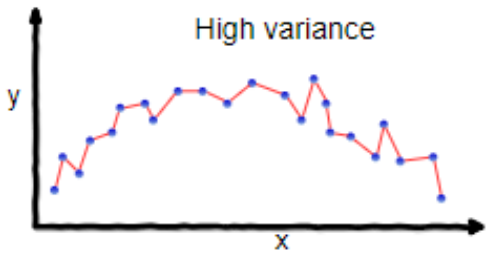
Bias, Variance, and Overfitting



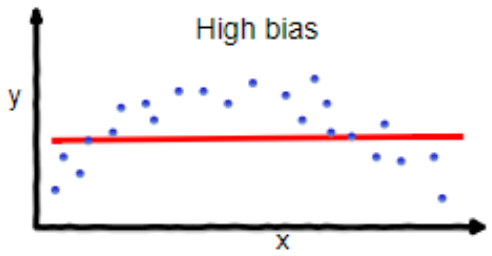




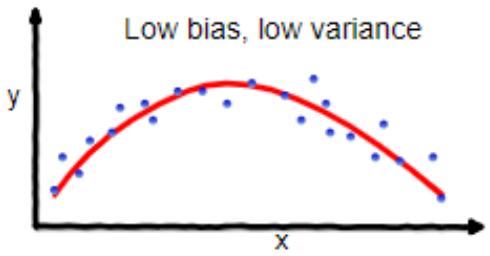
Bias, Variance, and Overfitting



overfitting



underfitting

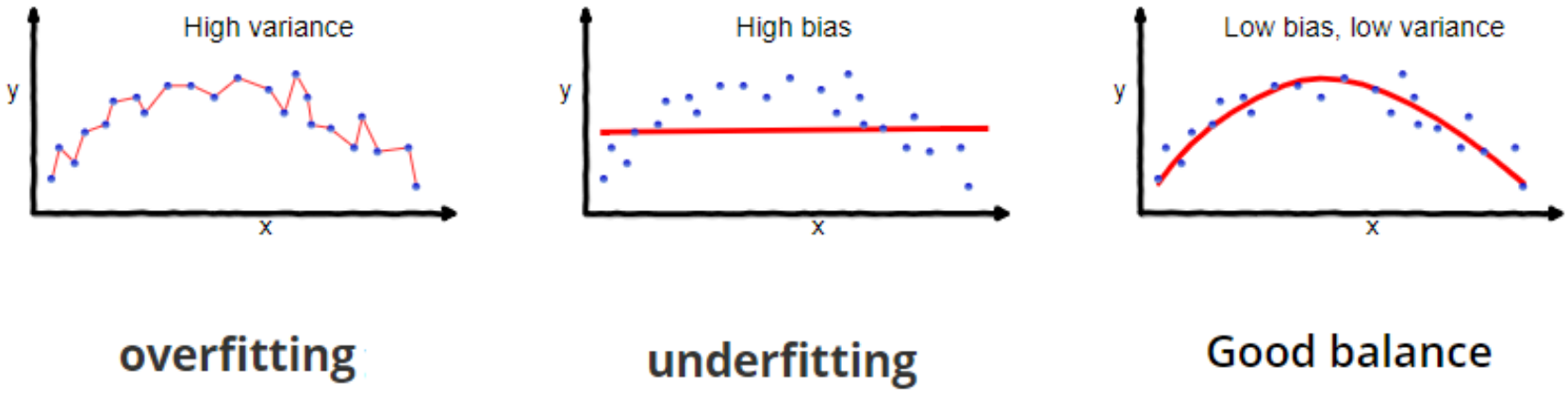


Good balance

Bias, Variance, and Overfitting

Overfitting and underfitting

- If a model learns the training data “too well” (low bias), it can lead to overfit
- This happens when the model mistakes noise for signal
- The model would not generalize to the test set (high variance)



Bias, Variance, and Overfitting

Training-test split

- A minimal measure to prevent overfitting
- The primary goal here is to make our model as generalizable as possible
- “Generalizable” means being able to perform well on unseen documents (other than the documents the model was trained on)
- When a model learns the noise or random fluctuations in the training set, this typically results in a model that performs poorly on new the test set

Step 1: Building a Labeled Data Set

How do we obtain a labeled data set?

- A form of manual content analysis
- “Ground truth” or “gold standard” fed to machines
- Our decisions/labels reflect latent features linked to the categories (some of which we are unconscious of)
- Manually labeled data are used for training (train set) and evaluation (test set)

Step 1: Building a Labeled Data Set

How do we obtain a labeled data set?

- Expert labeling
  - In many projects, a few domain experts work on a labeling (after training)
  - Annotators are trained to learn the concept and related guidelines
  - E.g., a researcher + two RAs from the department
- Crowd-sourced labeling
  - “Wisdom of crowd”: aggregated judgments of (online) non-experts converge to judgments of experts at much lower cost (Benoit et al, 2016)
  - Difficult to educate annotators on sophisticated tasks
  - Inductive measurement based on loose conceptualization

Step 1: Building a Labeled Data Set

Expert labeling vs. Crowd Sourcing

- Deductive vs. inductive
- Degree of training
- Scalability (cost)

Step 1: Building a Labeled Data Set

Selected texts for manual labeling

- Should reflect the entire corpus
- Mismatch leading to low performance: shift/drift
- E.g., drift in anti-vaccine discourse throughout 2020

Step 1: Building a Labeled Data Set

Iterative process

- Definition/operationalization does not often take place at once but in an iterative process
- In many cases, it is difficult to specify an entire annotation guidelines ex ante
- Preliminary labeling rule are written and applied to an initial set of docs
  → Annotators identify ambiguities in the rule
  → Revision of the rule → …

Step 1: Building a Labeled Data Set

Dealing with subjectivity

- Many concepts in humanities and social sciences are not straightforward
- They can involve high levels of subjectivity
- This is, from the beginning, why 1) careful conceptualization and 2) writing an excellent labeling rule, and 3) training coders are extremely important
- **Evaluation metrics**: Krippendorf's α, Cohen's κ (alternatives include Pearson's **r**, Spearman's ρ) (recommended R package: **irr**)

Step 1: Building a Labeled Data Set

Who are the annotators?

- Expert coding
  - Academics/students (Javdani and Chang 2023)
- Crowdsourcing
  - Skewed distribution of worked hours (Difallah et al. 2018)
  - Inattentive workers (Peyton et al. 2022; Ternovski 2022)
  - LLM-based responses (Veselovsky et al. 2023)
  - Demographic characteristic (Al Kuwatly et al. 2020)

Step 2: Extract Features

$$C_{train} \rightarrow \quad X_{train}$$

- Options include count vectors, TF-IDF vectors, word/document embeddings, etc.

Index	Token 1	Token 2	…	Token V-1	Token V
1	3	1.4	…	1.7	6
2	-0.8	6.4	…	5.7	-1.6
…					
7999	-2.8	0.9	…	3.3	-0.6
8000	3.7	1.4	…	5.7	-5.8

Step 3: Select/train Model(s)

So far we have:

- Built a labeled data set (Step 1)
- Generated a feature matrix (Step 2)
- This means that we have the outcome (y) and features (X_{train})

Now we will:

- Select a model F
- Learn model parameters β to build a classifier ($\hat{y} = F(\hat{\beta} * X)$)

Step 3: Select/train Model(s)

Numerous algorithms

- Logistic regression
- Naive Bayes
- Support vector machine
- Tree-based models (decision tree, random forest, XGBoost, etc.)
- Neural networks
- Etc.

Step 3: Select/train Model(s)

Logistic regression

- Used to classify a document into binary categories
  - Multinomial logistic regression for more than two
- One of the most useful analytics tools in science (not just NLP/ML)
- The baseline supervised learning algorithm for classification
- Forms the basis of neural networks

Step 3: Select/train Model(s)

Components of logistic regression (j documents n features)

- Features (e.g., tokens)
  - A document is represented as a vector of features $\vec{x} = [x_1, \ldots, x_n]$
- A classification function (F)
  - $p(y = 1|x)$ is computed for each document given the feature vector and β
- Loss function and algorithm for optimizing it (gradient descent)

Step 3: Select/train Model(s)

How does logistic regression compute predicted probabilities?

- $p(y = 1|x)$
  - We want to know the probability of y = 1 given a feature vector $\vec{x}$ (= $[x_1, \ldots, x_n]$)
  - For a simple count vector, it would be the number times each token appears in the document
- Logistic regression learns β, a vector of coefficients
  - A bias term b: a single number (a.k.a. intercept)
  - Weights $\vec{w}$ (= $[w_1, \ldots, w_n]$)
    - E.g., features signaling hateful intention would get high weights
  - With b, $\vec{w}$, and $\vec{x}$, we compute z (= $(\sum_{i=1}^{n} w_i x_i) + b$)

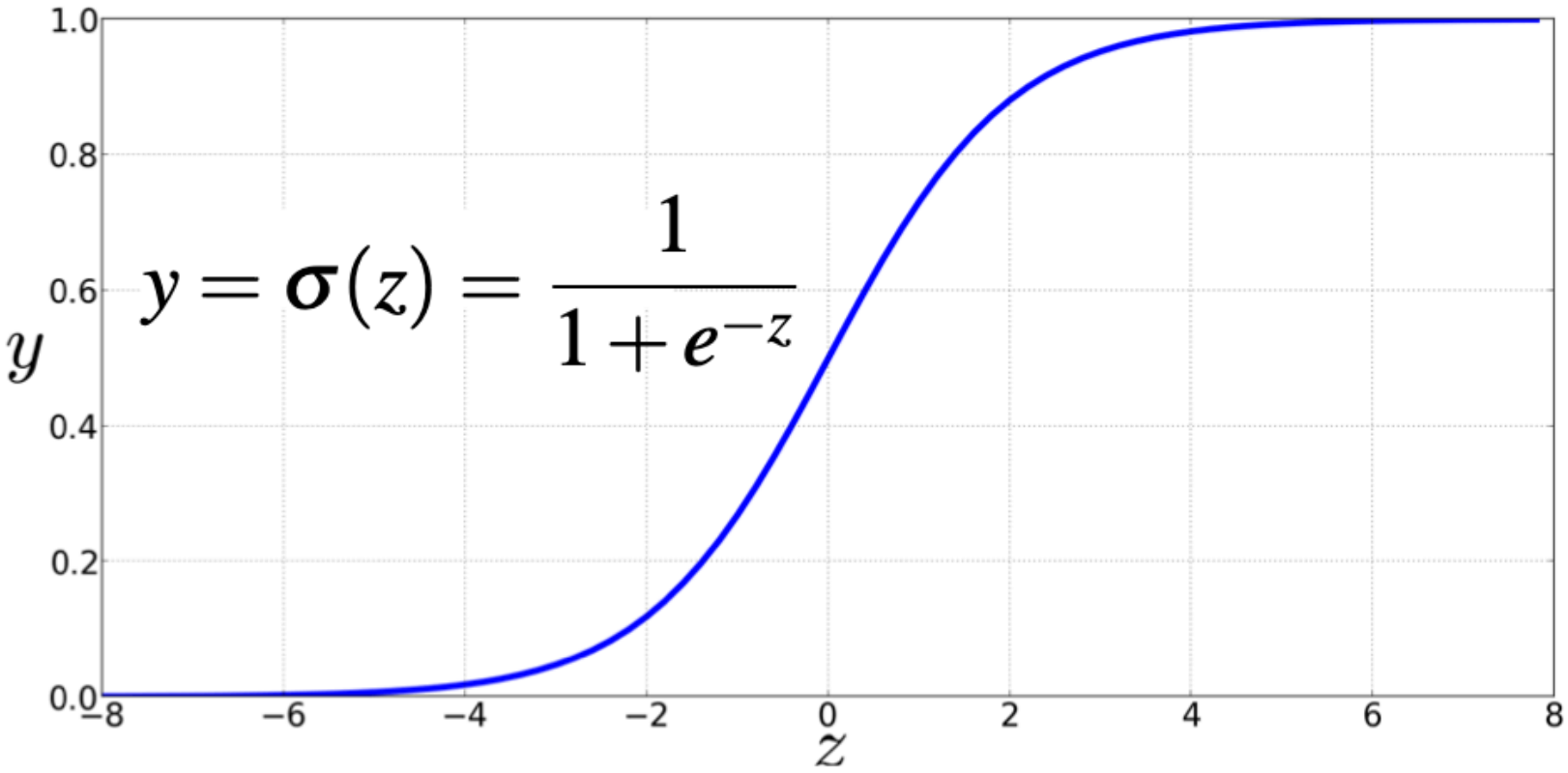
Step 3: Select/train Model(s)

How does logistic regression compute predicted probabilities?

- $z = (\sum_{i=1}^{n} w_i x_i) + b = \vec{w} \cdot \vec{v} +$ b

- $\sigma(z) = \frac{1}{1+e^{-z}} = \frac{1}{1+exp(-z)}$

Step 3: Select/train Model(s)



Step 3: Select/train Model(s)

How does logistic regression compute predicted probabilities?

$$p(y = 1|x) = \sigma(\vec{w} \cdot \vec{x} + b)$$

$$p(y = 0|x) = 1 - \sigma(\vec{w} \cdot \vec{x} + b)$$

Step 3: Select/train Model(s)

How do predicted probabilities turn into binary labels ($\hat{y}$)?

$$\begin{cases} 1 & \text{if } P(y = 1|x) > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

Step 3: Select/train Model(s)

Sentiment classification from movie reviews

“It’s hokey. There are virtually no surprises, and the writing is second-rate. So why was it so enjoyable? For one thing, the cast is great, Another nice touch is the music. I was overcome with the urge to get off the couch and start dancing? It sucked me in, and it’ll do the same to you.”

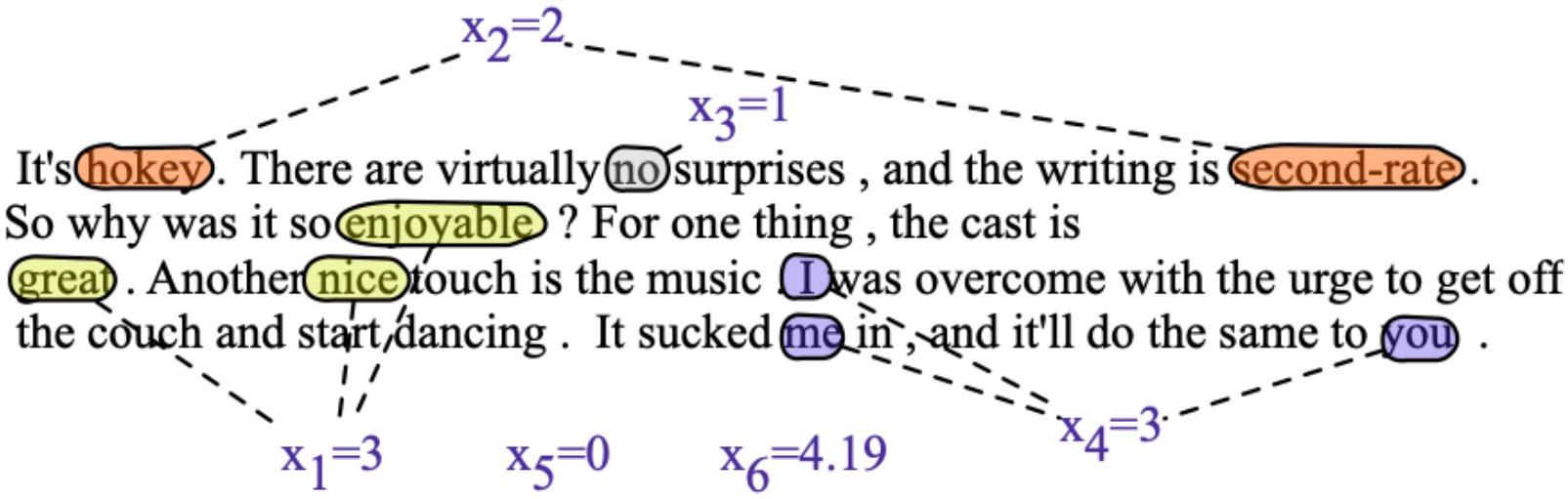
Step 3: Select/train Model(s)

Sentiment classification from movie reviews

Var	Definition
x_1	count(positive lexicon) $\in$ doc)
x_2	count(negative lexicon) $\in$ doc)
x_3	$\begin{cases} 1 & \text{if } \text{“no”} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$
x_4	count(1st and 2nd pronouns $\in$ doc)
x_5	$\begin{cases} 1 & \text{if } \text{“!”} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$
x_6	log(word count of doc)

Step 3: Select/train Model(s)

Sentiment classification from movie reviews



Step 3: Select/train Model(s)

Positive

- $p(+|x) = p(y = 1|x) = \sigma(\vec{w} \cdot \vec{x} + b)$
  $= \sigma([2.5, -5.0, -1.2, 0.5, 2.0, 0.7] \cdot [3,2,1,3,0,4.19] + 0.1)$
  $= \sigma(.833)$
  $= 0.70$

Negative

- $p(-|x) = p(Y = 0|x) = 1 - \sigma(\vec{w} \cdot \vec{x} + b)$
  $= 0.30$

Step 3: Select/train Model(s)

Learning coefficients

- MLE (Maximum Likelihood Estimation)
- Loss function / cross entropy
- Gradient descent

Step 4: Evaluate Performance

We have

- Manually labeled documents
- Split them into C_{train} (training set) and C_{test} (test set)
- Trained a classifier on C_{train} (with y_{train} and X_{train}) $\rightarrow F(\hat{\beta^*} X)$

Now we need to evaluate its performance on C_{test}

- We compare $\hat{y}_{test}$ (predicted labels) against y_{test} (true labels)

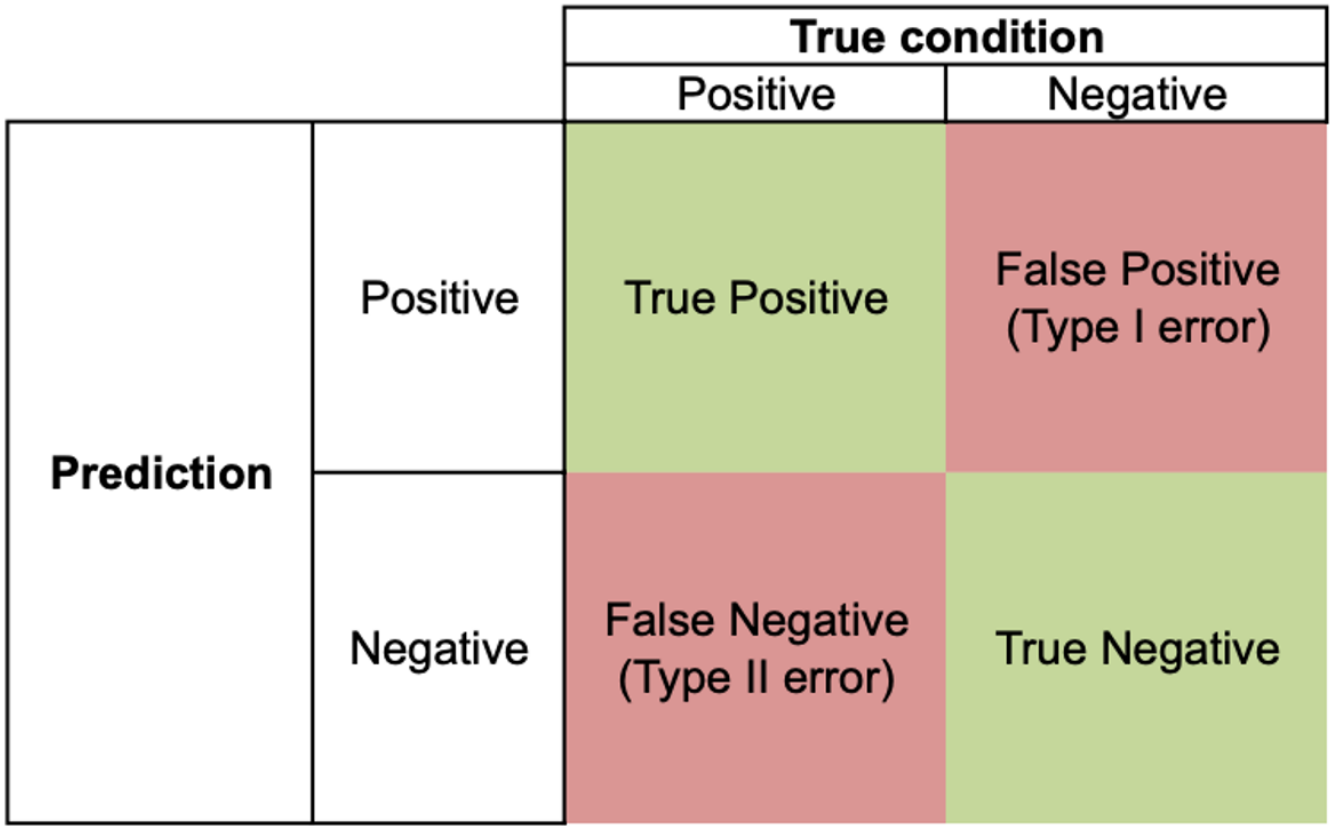
Step 4: Evaluate Performance

Performance metrics

- Accuracy: the proportion of all predictions (both positive and negative) that the model got right
- Precision: the proportion of positive predictions that were actually correct
- Recall: the proportion of actual positives that were correctly predicted
- F-1: the harmonic (as opposed to arithmetic) mean of precision and recall

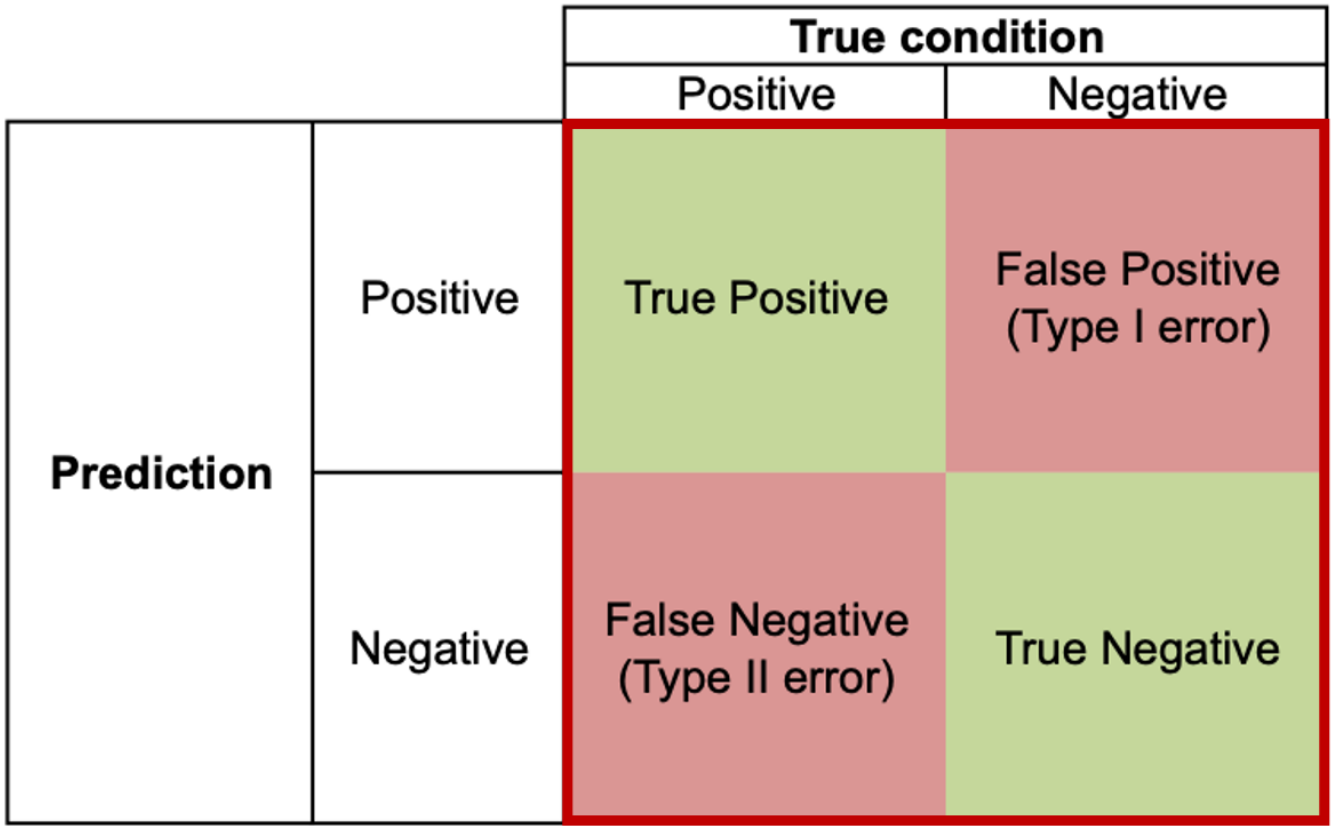
Step 4: Evaluate Performance

Confusion matrix: predictions against true labels



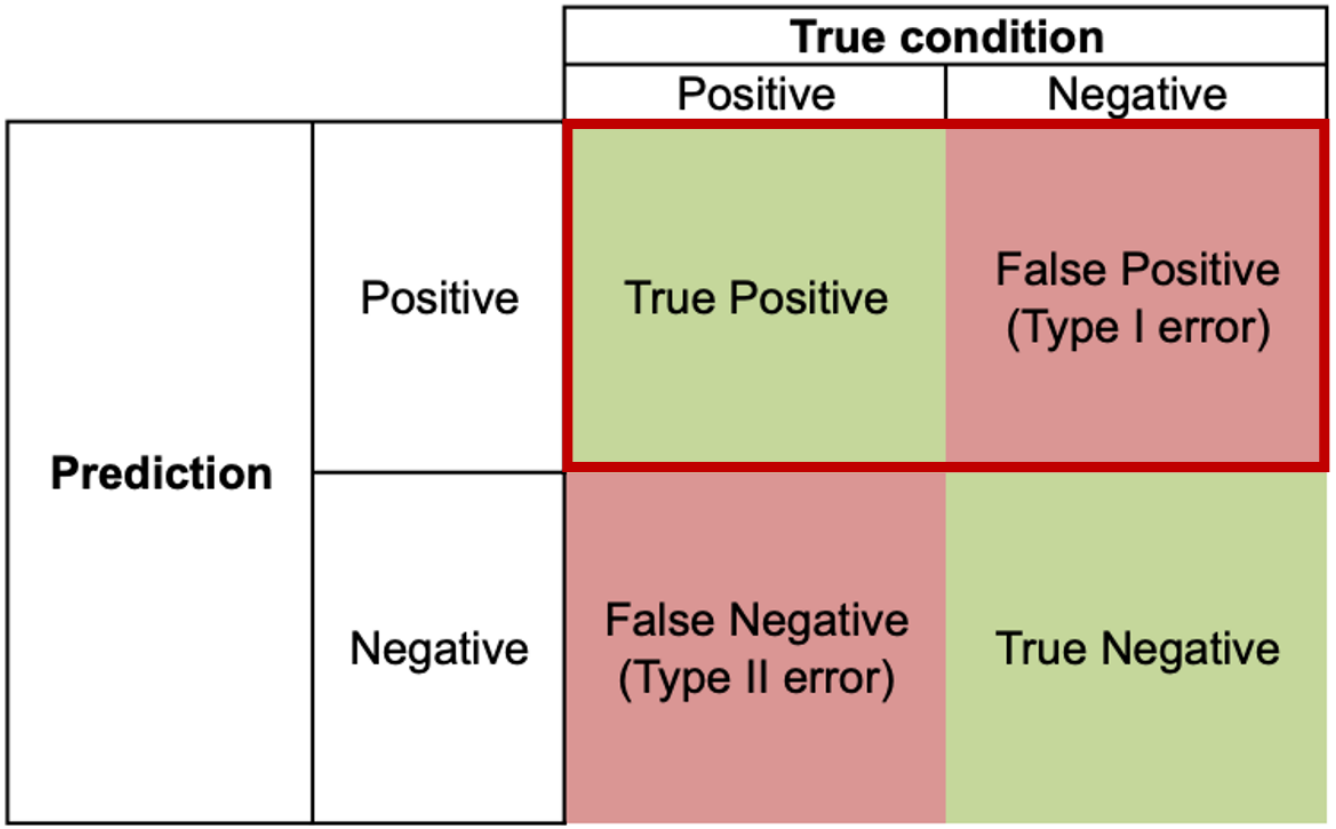
Step 4: Evaluate Performance

Accuracy: $\frac{TP+TN}{TP+TN+FP+FN}$



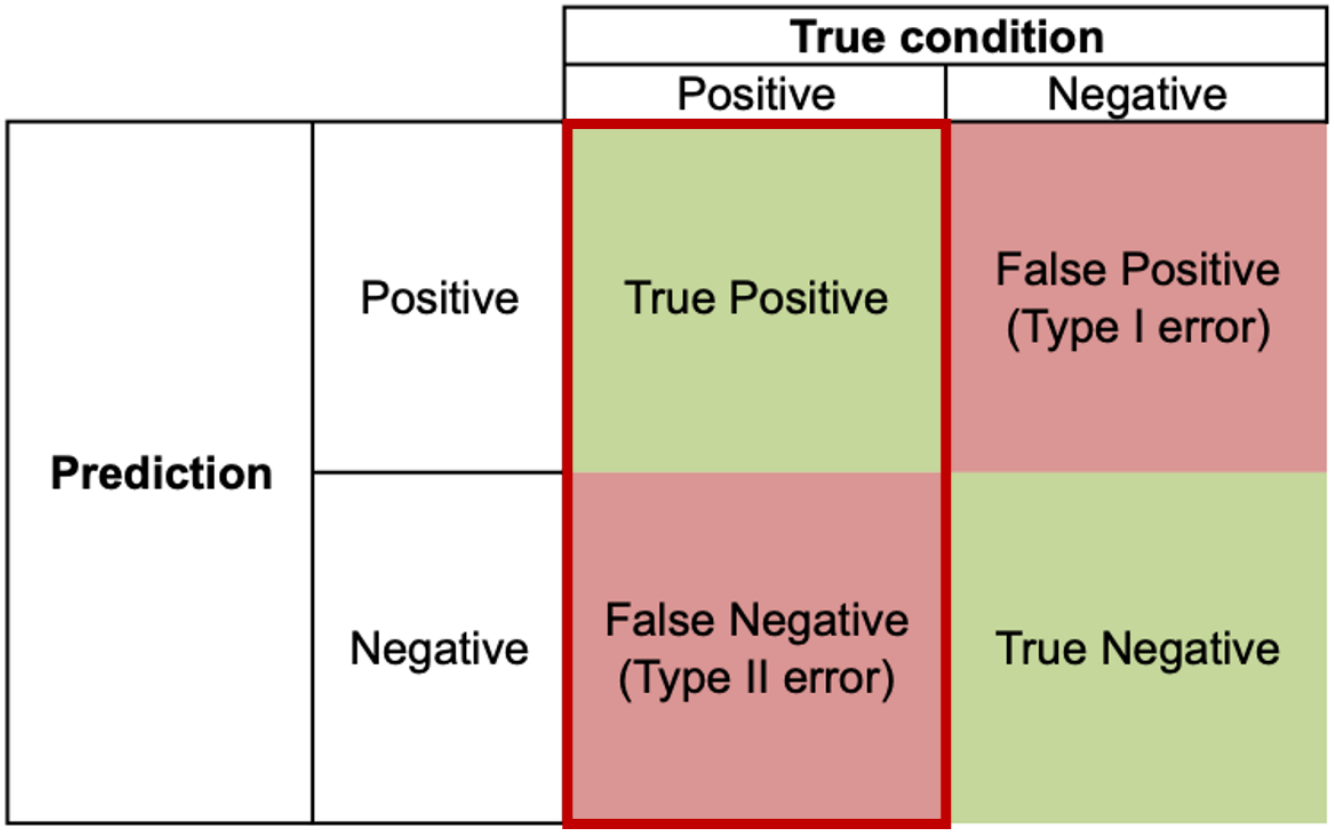
Step 4: Evaluate Performance

Precision: $\frac{TP}{TP+FP}$



Step 4: Evaluate Performance

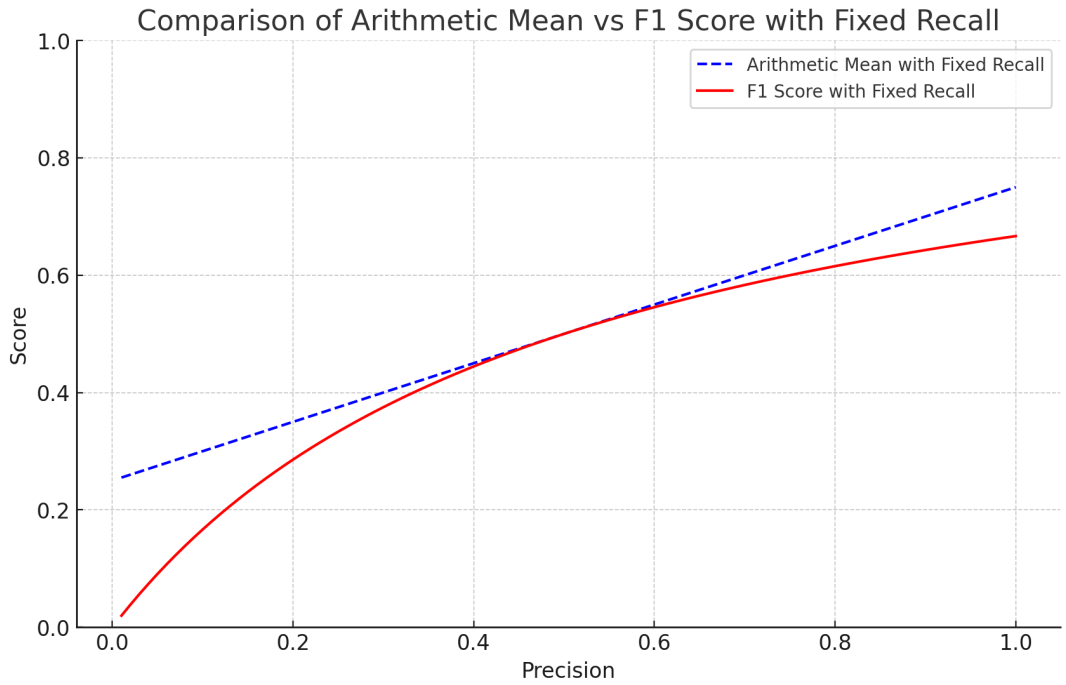
Recall: $\frac{TP}{TP+FN}$



Step 4: Evaluate Performance

F-1: $(2 \times precision \times recall) \,/\, (precision + recall)$

- Why not arithmetic mean $((precision + recall)/2)$?



Step 4: Evaluate Performance

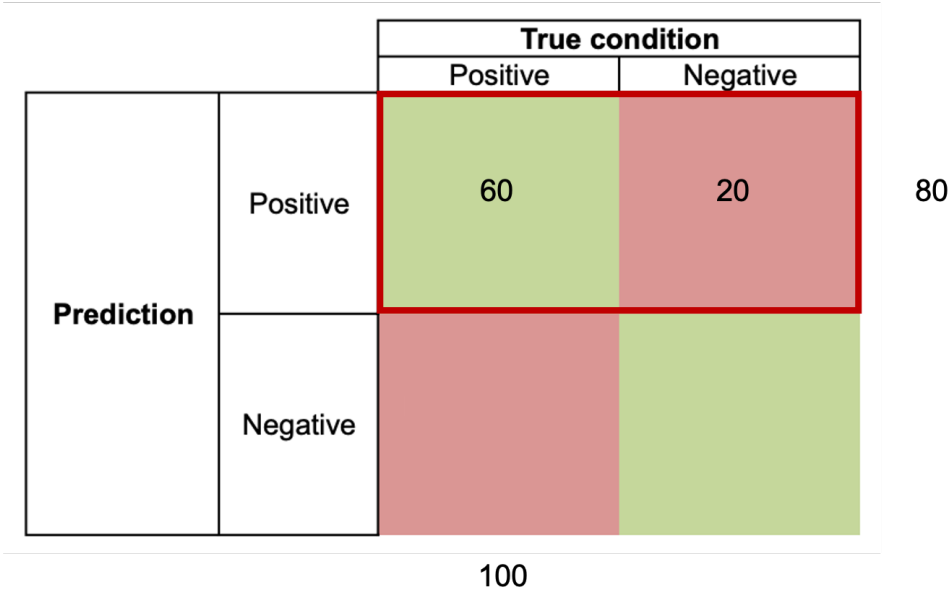
Precision/recall/F-1 & accuracy

- 100 positives
- 80 predicted positives
- 60 true positives

Step 4: Evaluate Performance

Precision/recall/F-1 & accuracy

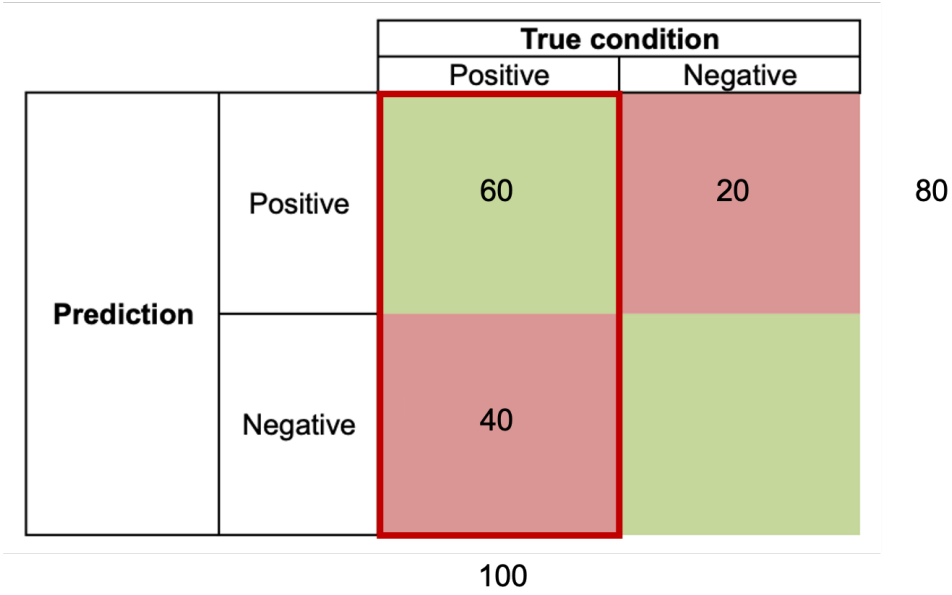
- Precision: $\frac{60}{60+20} = 0.75$



Step 4: Evaluate Performance

Precision/recall/F-1 & accuracy

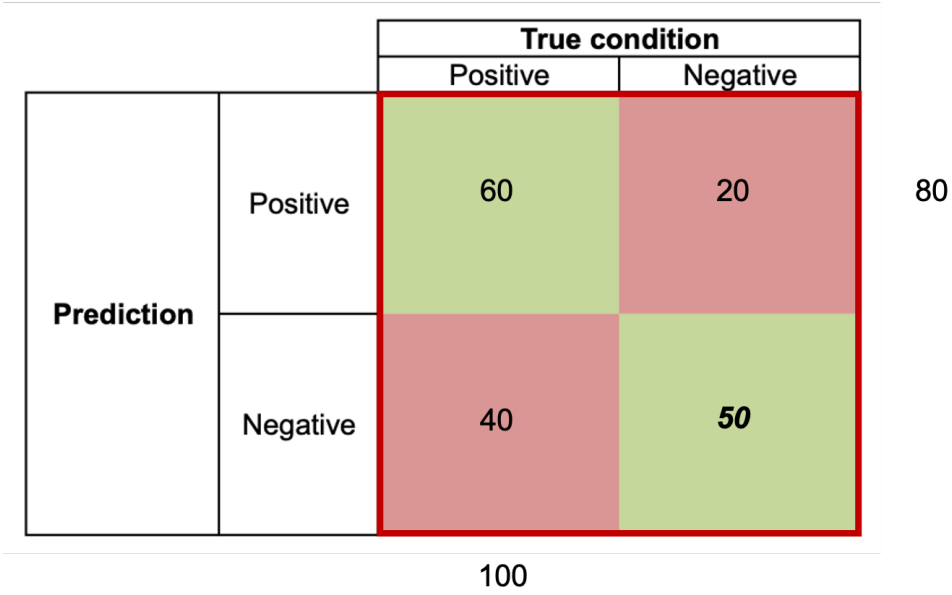
- Recall: $\frac{60}{60+40} = 0.6$



Step 4: Evaluate Performance

Precision/recall/F-1 & accuracy

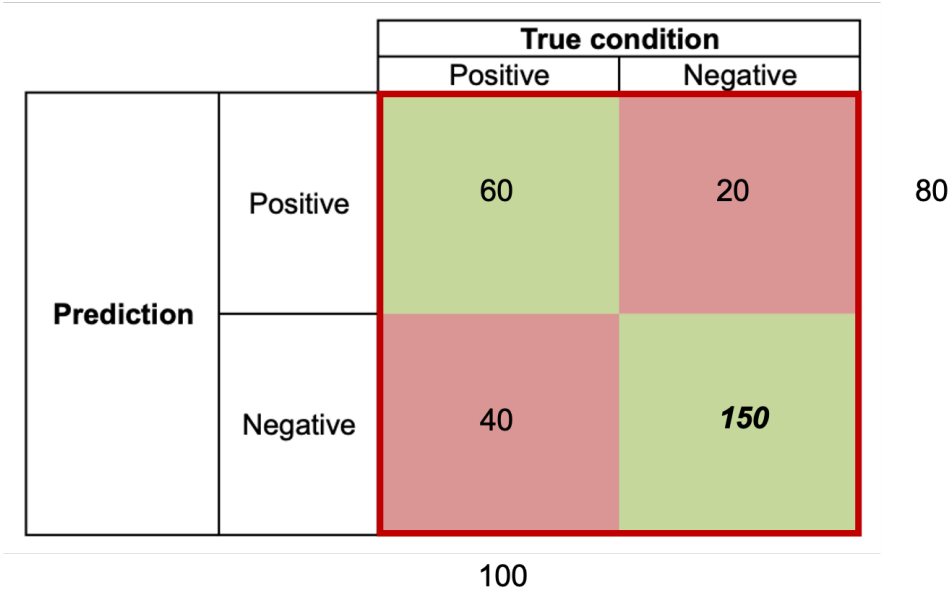
- Precision: $\frac{60}{60+20} = 0.75$
- Recall: $\frac{60}{60+40} = 0.6$
- Accuracy: $\frac{60+50}{60+20+40+50} = 0.65$



Step 4: Evaluate Performance

Precision/recall/F-1 & accuracy

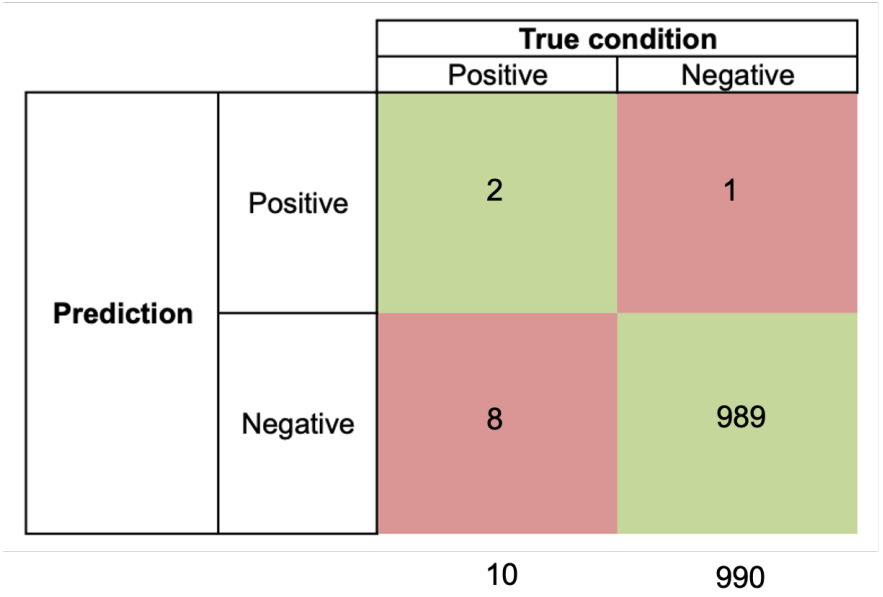
- Precision: $\frac{60}{60+20} = 0.75$
- Recall: $\frac{60}{60+40} = 0.6$
- Accuracy: $\frac{60+150}{60+20+40+150} = 0.78$



Step 4: Evaluate Performance

An extremely imbalanced case

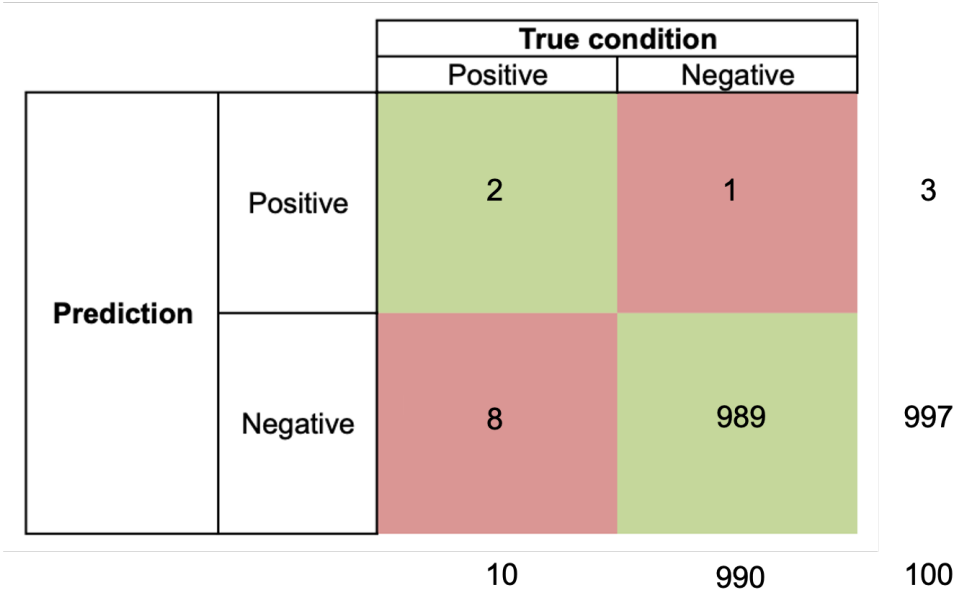
- Accuracy: ??
- Precision: ??
- Recall: ??
- F-1: ??



Step 4: Evaluate Performance

An extremely imbalanced case

- Accuracy: 0.991
- Precision: 0.66
- Recall: 0.2
- F-1: 0.31



Step 4: Evaluate Performance

Reminders

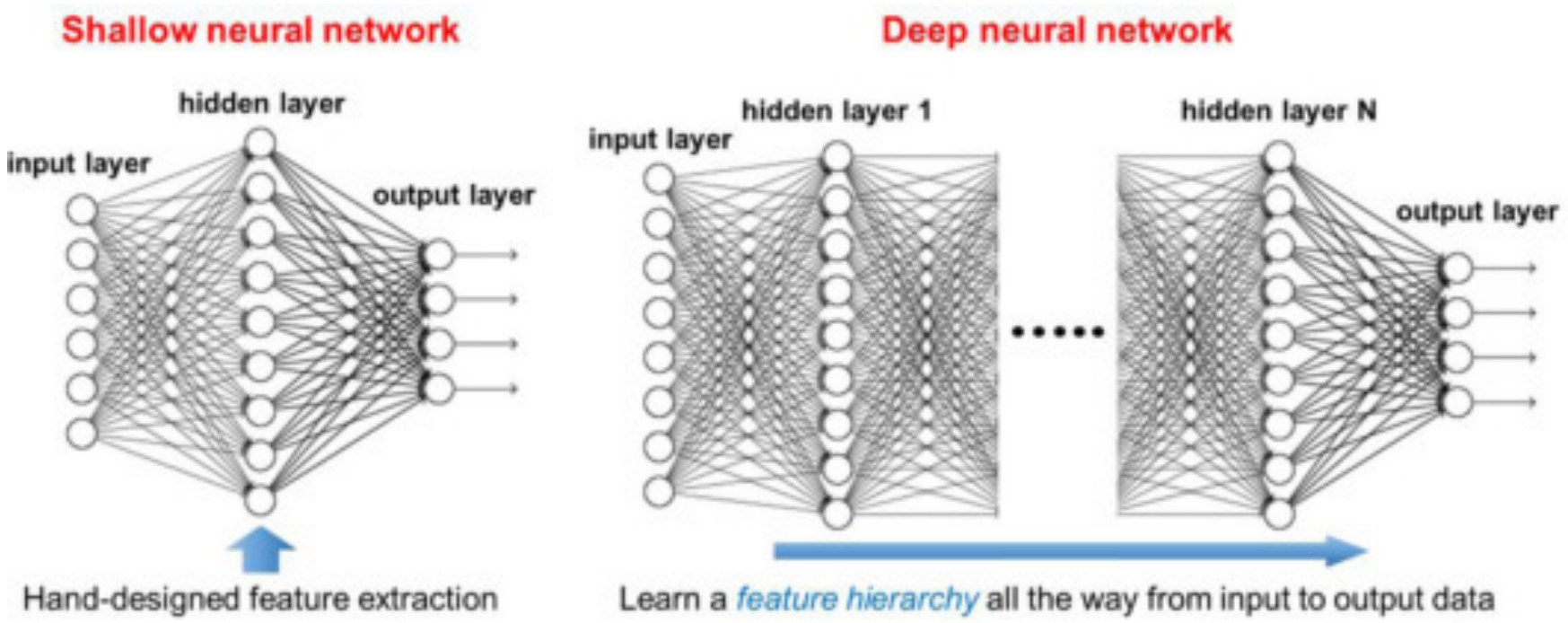
- Random train-test split: C_{train}, C_{test}
  - E.g., 10,000 comments labeled for hate speech into 8,000 and 1,000
- Our classifier learned **parameters**, maximizing performance on C_{train} and evaluating it on C_{test}

Step 4: Evaluate Performance

Parameters vs. hyper-parameters

- Parameter
  - Learned (estimated) from data (internal to the model)
  - E.g., logistic regression weights/coefficients (= β)
- Hyper-parameters
  - Defines the model structure itself (not internal to the model)
  - E.g., the size of a regularization term in logistic regression, the number of layers or learning rate in neural networks, etc.

Step 4: Evaluate Performance



Step 4: Evaluate Performance

Hyper-parameters influence model performance, and we want to “tune” them

- With different hyper-parameter values, we could fit a model configured with each value on C_{train} and evaluate performance on C_{test}
- E.g., regularization strength λ in logistic regression
  - Train different models with different λ values on C_{train}
  - Evaluate on C_{test}
  - Pick the best performing model

Step 4: Evaluate Performance

What could go wrong?

- In comparing different models (different hyperparameter values), we might overfit on C_{train}
- By repeatedly using the training set, our comparison can be affected by the specific characteristics of the training set

Step 4: Evaluate Performance

Validation set

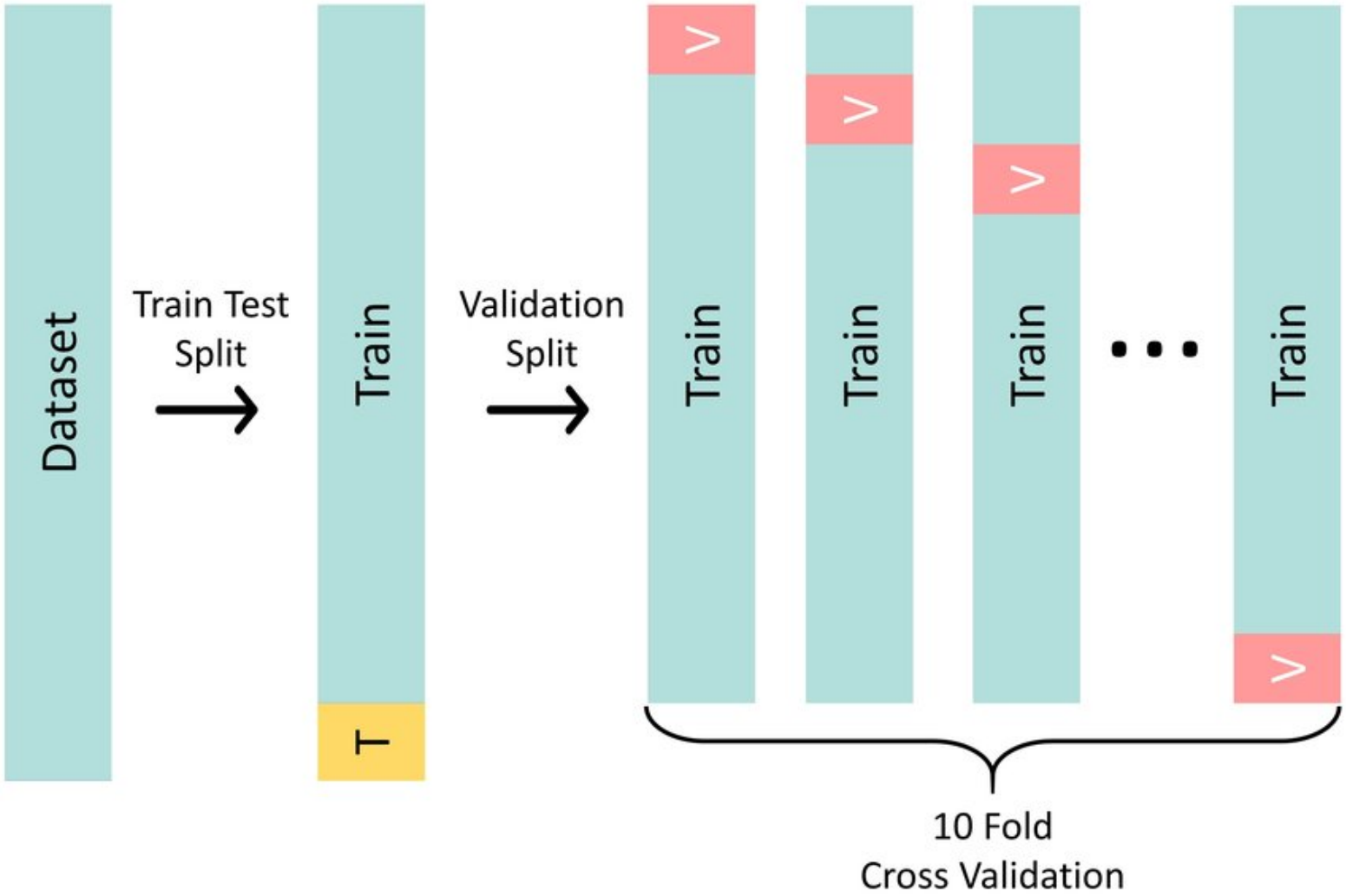
- We split the labeled data set into a training set, a “validation set”, and a test set: C_{train}, $C_{validation}$, and C_{test}
- Train a model on C_{train} and see how it performs on $C_{validation}$
- Repeat this step for multiple configurations (e.g., hyper-parameters)
- Pick the best-performing configuration
- Train a final model based on the configuration on C_{train} + $C_{validation}$
- Evaluate on C_{test}

Step 4: Evaluate Performance

K-fold cross-validation

- Randomly split C_{train} into K equal parts or "folds" (commonly 5 or 10)
- For each iteration
  - Treat one fold as the "validation set"
  - Train your model on the remaining $K - 1$ folds
  - Evaluate performance on the validation set kept aside
- After cyclig through all iterations
  - Aggregate the performance metrics obtained from each iteration
  - Choose the classifier with the highest cross-validated performance
  - This step may invovle not just hyperparameter tuning but also things like feature repesentation, etc.
- (Re)train the chosen best classifier on C_{train} (all K folds combined) and evalute on C_{test}

Step 4: Evaluate Performance



Summary

Supervised text classification provides a highly useful tool to assign labels to documents

- Be aware of the principles of building a labeled data set
  - Conceptualization, intercoder reliability, annotator bias, etc.
- Validate (with appropriate evaluation metrics)!

Summary

For a discussion of effectively building training data for imbalanced classification

- See the Week 6 of the NLP for DHCSS

More recent developments in pre-trained transformer models (e.g., BERT, GPT, etc) let us capitalize on the knowledge about language in those models

- This yields high-quality “contextual embeddings”
- We can “fine-tune” these models for text classification
- See the Week 13 and Week 14 of the NLP for DHCSS

Guided Coding

Text classification with movie reviews data (link)

Keyword-based Methods

Keyword-based Methods: themes

Things to be covered

- Counting keywords
- Emotion/sentiment dictionary
- Moral foundation dictionary
- Validations and limitations
- Guided coding: sentiment analysis (Python) & discriminating words (R)

Keyword-based Methods

Supervised learning assigns texts into known categories

- Construct (or outsource) training data (documents are annotated)
- Build a statistical model that provides a mapping between texts and labels
- Label unseen texts (predict their labels)

Keyword-based methods offer alternatives

- Word-level approach
- A list/dictionary of keywords (sometimes with weights)
- Quick, clear, and easy to communicate (but usually less accurate)

Counting Keywords

E.g., we want to measure attention to (or interest in) presidential politics in U.S. news media

- Count the number of news articles that mention the words `president` or `White House`
- How can this approach go wrong?

Counting Keywords

False positive (Type I Error)

- Occurs when a test incorrectly indicates the presence of a condition when it is not actually present
  - E.g., Diagnose a patient has a disease when the patient does not

False negative (Type II Error)

- Occurs when a test fails to detect the presence of a condition when it is actually present
  - E.g., Diagnose a patient does not have a disease when the patient does

Couting Keywords

However, keywords carefully curated (along with validation) can be highly effective

- Validation: the process of assessing the degree of validity

Couting Keywords

Example: Yian et al. (2021)

- Goal is to examine the use of scientific knowledge in policy during the COVID-19 pandemic
- Identifying policy documents and scientific publications about the COVID-19
  - Policy documents about the pandemic in many countries (**Overton**)
  - Scientific publications cited in the policy documents (**Dimensions**)

Couting Keywords

Example: Yian et al. (2021)

- Keywords to identify policy documents about the pandemic (see here for non-English keywords)
  - `"2019-nCoV"`
  - `"COVID-19"`
  - `"coronavirus"`
  - `"corona virus"`
  - `“SARS-CoV-2”`

Couting Keywords

Example: Yian et al. (2021)

- Keywords to identify scientific publications about the pandemic
  - `"2019-nCoV"`
  - `"COVID-19"`
  - `“SARS-CoV-2”`
  - `"HCoV-2019"`
  - `"hcov"`
  - `"NCOVID-19"`
  - `"severe acute respiratory syndrome coronavirus 2"`
  - `"severe acute respiratory syndrome corona virus 2"`
  - (`"coronavirus"` OR `"corona virus"`) AND (`Wuhan` OR `China` OR `novel`)

Couting Keywords

Note that humans are limited in recalling relevant keywords

- Relevant when you want to go as representative/comprehensive as possible
- This might lead to low recall (omission of relevant documents)
- Statistical approaches to deal with incomprehensiveness of human-generated keywords (e.g., King et al. (2017))

Dictionary Methods

Generalization of keyword counting

- The previous examples of the use of keywords are often *ad hoc*
- Pre-defined set of keywords associated with certain concepts like sentiment
- We can measure *a host of* concepts using dictionary methods
  - Sentiment, emotion, morality, aggression, etc.
- The goal of dictionary methods is often identical to supervised learning
  - Assign documents into known categories
- High level of domain expertise / qualitative judgment can be required

Dictionary Methods

Two approaches

- We can construct our own dictionary
- Alternatively (and much more common), there are a variety of off-the-shelf dictionaries

Dictionary Methods

E.g., measuring positive sentiment in newspaper articles

- We find a dictionary of keywords with each associated with a score signaling the degree of pleasantness (e.g., “fun”, “excellent”, etc.)
- Count the number of times each keywords appears and add them up
- (Optional) we normalize by document length (word count)

Dictionaries for Affect

Many concepts and meanings are named under *affect*

- *Affect* is a term that encompasses emotion, sentiment, personality, mood, attitudes, etc. (Picard, 1995; Scherer, 2000)
- See [JM] Chp. 25 (Section 25.1) for more theoretical discussions of emotions

Dictionaries for Affect

Keywords in affect dictionaries

- Affective lexicons: keywords that carry particularly strong cues to affect meanings (= connotations)

Linguistic Inquiry and Word Count (LIWC)

Most recent version (LIWC-22) (proprietary)

- Not just affect: + 100 linguistic dimensions
  - E.g., positive/negative, moralization, I-word, etc.
- Uses a dictionary to calculate the percentage of words in the text that match certain dimension

Linguistic Inquiry and Word Count (LIWC)

E.g., "I am very disappointed"

Your text sample is 4 words. The LIWC-22 analysis of the text sample you entered is below. Note that LIWC-22 actually produces about 100 different output dimensions. Remember: the more text that you have available for analysis, the more trustworthy and reliable your results will be.

RESULTS

Traditional LIWC Dimension	Your Text	Average for Social Media Language
I-words (I, me, my)	25.00	5.44
Positive Tone	0.00	5.93
Negative Tone	25.00	2.34
Social Words	0.00	6.74
Cognitive Processes	0.00	8.86
Allure	0.00	8.62
Moralization	0.00	0.27
Summary Variables		
Analytic	0.00	47.06
Authentic	89.41	62.38

Traditional LIWC dimensions reflect percentage of total words within the text you provided. The Summary Variables are composites derived from scientific research that have been converted to 100-point scales, where 0 = "very low" along the dimension and 100 = "very high." Analytic refers to analytical or formal thinking. Authentic is a property of language that reflects when someone is speaking in an unfiltered, off-the-cuff fashion.

Want to learn more about the meaning of the LIWC output? See: Interpreting LIWC Output.

ANALYZE ANOTHER TEXT

Linguistic Inquiry and Word Count (LIWC)

Very widely used for tasks including

- Detecting political sentiment from tweets (Tumasjan et al. 2010)
- Predicting the onset of depression in individuals based on text from social media (De Choudhury et al. 2013)
- Differentiating happy romantic couples from unhappy ones based on their instant messages (Hancock et al. 2007)

Linguistic Inquiry and Word Count (LIWC)

Kramer et al. (2014)

- Examines emotional contagion on Facebook
- N = 689, 003 Facebook users
- Manipulated content shown on news feeds to test emotional contagion hypothesis
- Treatment 1: positive content more visible on news feed
- Treatment 2: negative content more visible on news feed
- Control: no news feed intervention

Linguistic Inquiry and Word Count (LIWC)

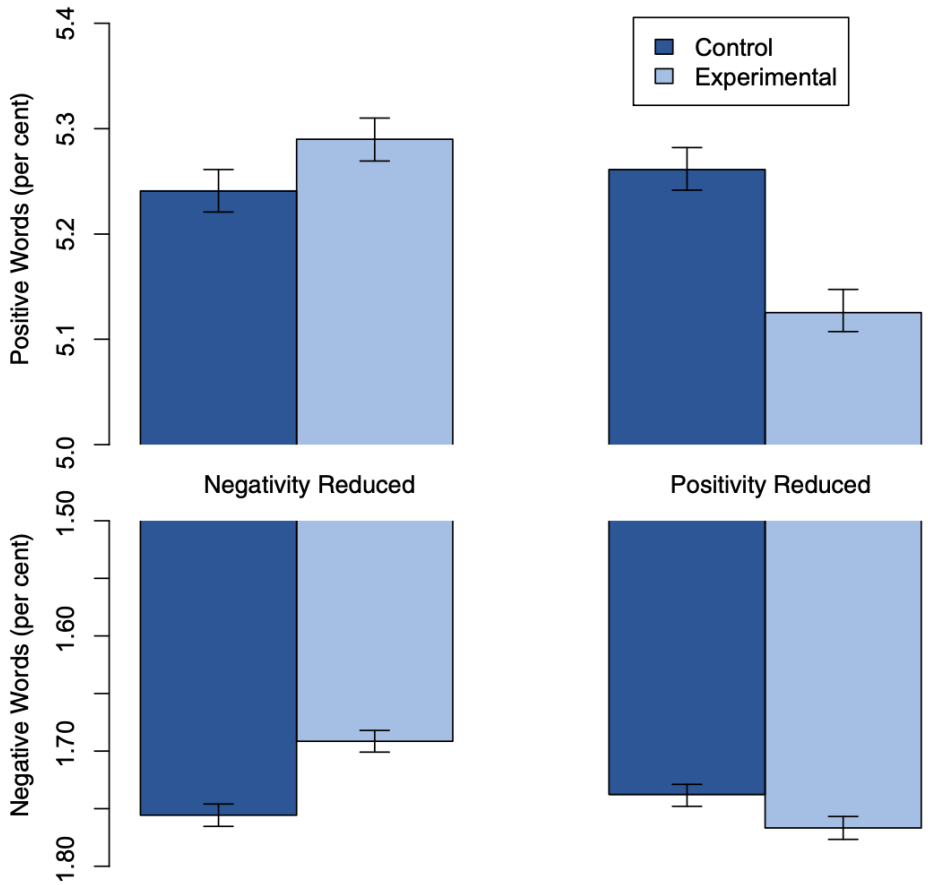
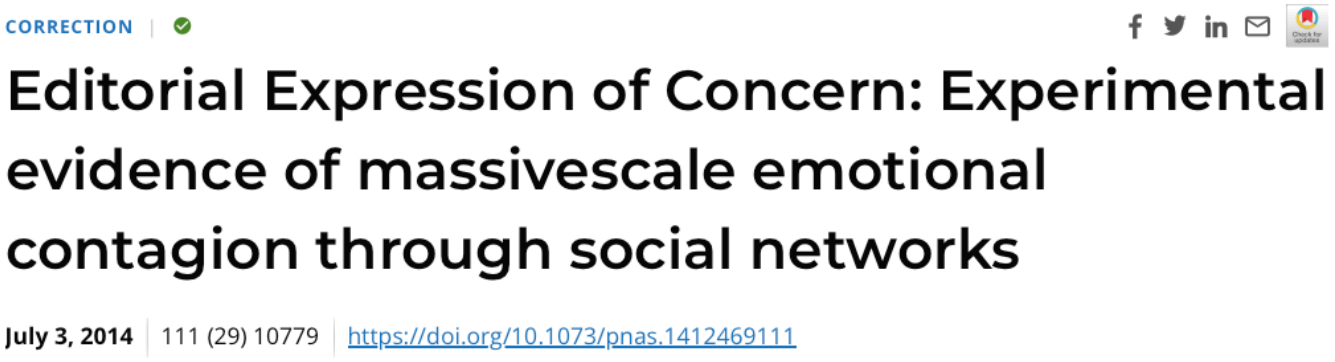


Fig. 1. Mean number of positive (*Upper*) and negative (*Lower*) emotion words (percent) generated people, by condition. Bars represent standard errors.

Linguistic Inquiry and Word Count (LIWC)

Huge concerns were raised about the ethics of the study



CORRECTION

Editorial Expression of Concern: Experimental evidence of massivescale emotional contagion through social networks

July 3, 2014 | 111 (29) 10779 | https://doi.org/10.1073/pnas.1412469111

Valence Aware Dictionary and sEntiment Reasoner (VADER)

Hutto et al.(2014)

- Improves LIWC and other sentiment dictionaries focused on social media
- Not just polarity but also intensity
- Initialisms, emoticons, or slangs
- Crowd-sourced labeling of keywords

Valence Aware Dictionary and sEntiment Reasoner (VADER)

Hutto et al.(2014)

- Grammatical and syntactical cues
  - Punctuation: good vs. good!!!
  - Capitalization: great vs. GREAT
  - Degree modifiers: extremely good vs. good
  - Contrastive conjunction: “The food here is great, but the service is horrible”
  - Negation: “The food here isn’t really all that great”

Valence Aware Dictionary and sEntiment Reasoner (VADER)

Example keywords and weights

- `great`: 3.1
- `good`: 1.9
- `sucks`/`sux`: –1.5
- `:(`: –2.2
- `horrible`: –2.5

Valence Aware Dictionary and sEntiment Reasoner (VADER)

	Correlation to ground truth (mean of 20 human raters)	3-class (positive, negative, neutral) Classification Accuracy Metrics			Ordinal Rank (by F1)		Correlation to ground truth (mean of 20 human raters)	3-class (positive, negative, neutral) Classification Accuracy Metrics		
		Overall Precision	Overall Recall	Overall F1 score				Overall Precision	Overall Recall	Overall F1 score
Social Media Text (4,200 Tweets)							**Movie Reviews (10,605 review snippets)**			
Ind. Humans	**0.888**	0.95	0.76	0.84	2	1	0.899	0.95	0.90	0.92
VADER	0.881	**0.99**	**0.94**	**0.96**	1*	2	0.451	0.70	0.55	0.61
Hu-Liu04	0.756	0.94	0.66	0.77	3	3	0.416	0.66	0.56	0.59
SCN	0.568	0.81	0.75	0.75	4	7	0.210	0.60	0.53	0.44
GI	0.580	0.84	0.58	0.69	5	5	0.343	0.66	0.50	0.55
SWN	0.488	0.75	0.62	0.67	6	4	0.251	0.60	0.55	0.57
LIWC	0.622	0.94	0.48	0.63	7	9	0.152	0.61	0.22	0.31
ANEW	0.492	0.83	0.48	0.60	8	8	0.156	0.57	0.36	0.40
WSD	0.438	0.70	0.49	0.56	9	6	0.349	0.58	0.50	0.52
Amazon.com Product Reviews (3,708 review snippets)							**NY Times Editorials (5,190 article snippets)**			
Ind. Humans	0.911	0.94	0.80	0.85	1	1	0.745	0.87	0.55	0.65
VADER	0.565	0.78	0.55	0.63	2	2	0.492	0.69	0.49	0.55
Hu-Liu04	0.571	0.74	0.56	0.62	3	3	0.487	0.70	0.45	0.52
SCN	0.316	0.64	0.60	0.51	7	7	0.252	0.62	0.47	0.38
GI	0.385	0.67	0.49	0.55	5	5	0.362	0.65	0.44	0.49
SWN	0.325	0.61	0.54	0.57	4	4	0.262	0.57	0.49	0.52
LIWC	0.313	0.73	0.29	0.36	9	9	0.220	0.66	0.17	0.21
ANEW	0.257	0.69	0.33	0.39	8	8	0.202	0.59	0.32	0.35
WSD	0.324	0.60	0.51	0.55	6	6	0.218	0.55	0.45	0.47

Table 4: VADER 3-class classification performance as compared to individual human raters and 7 established lexicon baselines across four distinct domain contexts (clockwise from upper left: tweets, movie reviews, product reviews, opinion news articles).

Valence Aware Dictionary and sEntiment Reasoner (VADER)

	3-Class Classification Accuracy (F1 scores) Test Sets			
	Tweets	Movie	Amazon	NYT
VADER	**0.96**	0.61	**0.63**	**0.55**
NB (tweets)	0.84	0.53	0.53	0.42
ME (tweets)	0.83	0.56	0.58	0.45
SVM-C (tweets)	0.83	0.56	0.55	0.46
SVM-R (tweets)	0.65	0.49	0.51	0.46
NB (movie)	0.56	**0.75**	0.49	0.44
ME (movie)	0.56	**0.75**	0.51	0.45
NB (amazon)	0.69	0.55	0.61	0.48
ME (amazon)	0.67	0.55	0.60	0.43
SVM-C (amazon)	0.64	0.55	0.58	0.42
SVM-R (amazon)	0.54	0.49	0.48	0.44
NB (nyt)	0.59	0.56	0.51	0.49
ME (nyt)	0.58	0.55	0.51	0.50

Table 5: Three-class accuracy (F1 scores) for each machine trained model (and the corpus it was trained on) as tested against every other domain context (SVM models for the movie and NYT data were too intensive for our multicore CPUs with 94GB RAM)

Validating Dictionaries

- Validation
  - (Semi-)randomly sample a small number of texts
  - Generate human labels
  - Compare dictionary-based labels and human labels
    - We will discuss useful metrics next week
- Note that dictionaries do not always travel across contexts
  - E.g., “tax”, “cost”, “cancer” are considered signaling negative connotations, but they do not in accounting/finance
- Easier when the dictionary and the research objectives are highly aligned
  - E.g., keywords related to the COVID-19 from Yian et al. (2021)

Limitations of Dictionaries

Barbera et al. (2021): sentiment in NYT articles

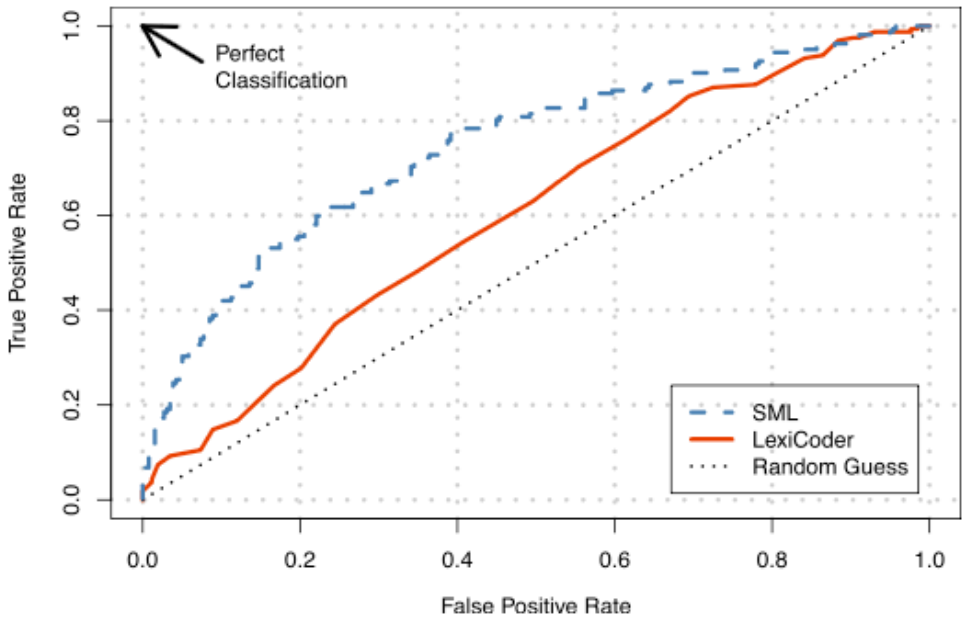


Figure 5. Receiver Operator Characteristic Curve: Lexicoder versus SML.
Note: The x-axis gives the false positive rate—the proportion of all negatively toned articles in CF Truth that were classified as positively toned—and the y-axis gives the true positive rate—the proportion of all positively toned articles in CF Truth that were classified as positive. Each point on the curve represents the misclassification rate for a given classification threshold. The corpus used in the analysis is based on the keyword search of *The New York Times* 1980–2011.

Limitations of Dictionaries

Widmann and Wich (2023): classifying emotions

Emotions	Actual	Predicted	Precision	Recall	F1
		ed8 dictionary			
Anger	508	281	0.83	0.46	0.59
Fear	189	287	0.43	0.66	0.52
Disgust	86	182	0.30	0.63	0.40
Sadness	201	289	0.41	0.59	0.48
Joy	143	179	0.46	0.58	0.52
Enthusiasm	220	248	0.44	0.50	0.47
Pride	158	247	0.31	0.48	0.38
Hope	305	303	0.53	0.53	0.53
		Word-embeddings-based neural network approach			
Anger	508	500	0.80	0.78	0.79
Fear	189	152	0.61	0.49	0.55
Disgust	86	67	0.60	0.47	0.52
Sadness	201	122	0.70	0.42	0.53
Joy	143	92	0.68	0.44	0.54
Enthusiasm	220	176	0.64	0.51	0.57
Pride	158	123	0.52	0.41	0.46
Hope	305	265	0.69	0.60	0.64
		Transformer-based (ELECTRA) approach			
Anger	508	495	0.85	0.83	0.84
Fear	189	221	0.60	0.70	0.64
Disgust	86	89	0.61	0.63	0.62
Sadness	201	181	0.64	0.57	0.60
Joy	143	122	0.70	0.59	0.64
Enthusiasm	220	242	0.62	0.68	0.65
Pride	158	151	0.61	0.58	0.60
Hope	305	352	0.68	0.78	0.73

Limitations of Dictionaries

Dictionaries can still be useful

- Low resource (no need to build a training set)
- Fast implementation and less computing power
- Transparent (much less black-boxy)
- Useful for preliminary analysis

Discriminating Words

How can we identify words that distinguish two groups?

- Political parties
- Gender identities
- Periods or generations

Discriminating Words

The resulting keywords are of interest in and of itself

- Serve as a lens through which to examine framing, topic, etc.
- E.g., how do Democrats and Republicans in the U.S. talk differently about abortion?
- We can apply this to generate keywords, which in turn can be used as a dictionary (p. 181 in [**GRS**])
  - E.g., pro-vaccine vs. anti-vaccine comments during the COVID-19 pandemic

Discriminating Words

Some approaches

- Difference of frequencies: $|f^i(horrible) - f^j(horrible)|$
- Difference in proportions: $|p^i(horrible) - p^j(horrible)|$
- Classification: take weights from “g = f(w)”

→ See Monroe et al. (2008) for detailed discussions

Discriminating Words

Fightin' Words (Monroe et al., 2008)

- Words that are used differently by two political parties
- The extent to which each word is used differently by two political parties

Discriminating Words

The log odds ratio (LOR) of the word “horrible” is given by:

$$LOR(horrible) = \log\left(\frac{p^i(horrible)}{1 - p^i(horrible)}\right) - \log\left(\frac{p^j(horrible)}{1 - p^j(horrible)}\right)$$

Which simplifies to:

$$= \log\left(\frac{f^i(horrible)}{n^i - f^i(horrible)}\right) - \log\left(\frac{f^j(horrible)}{n^j - f^j(horrible)}\right)$$

Where $p^i(horrible)$ and $p^j(horrible)$ are probabilities of “horrible” in corpus **i** and **j**, f^i and n^j are the number of times “horrible” appears in the respective corpus, and n^i and n^j are the numbers of words in the respective corpus.

To further incorporate a prior estimate of what we expect the frequency of each word **w** to

be, we add the counts from the entire corpus to the numerator and denominator:

$$\delta_w^{(i-j)} = \log\left(\frac{f_w^i + \alpha_w}{n_i + \alpha_0 - (f_w^i + \alpha_w)}\right) - \log\left(\frac{f_w^j + \alpha_w}{n_j + \alpha_0 - (f_w^j + \alpha_w)}\right)$$

We also need an estimate for the variance of the log-odds-ratio:

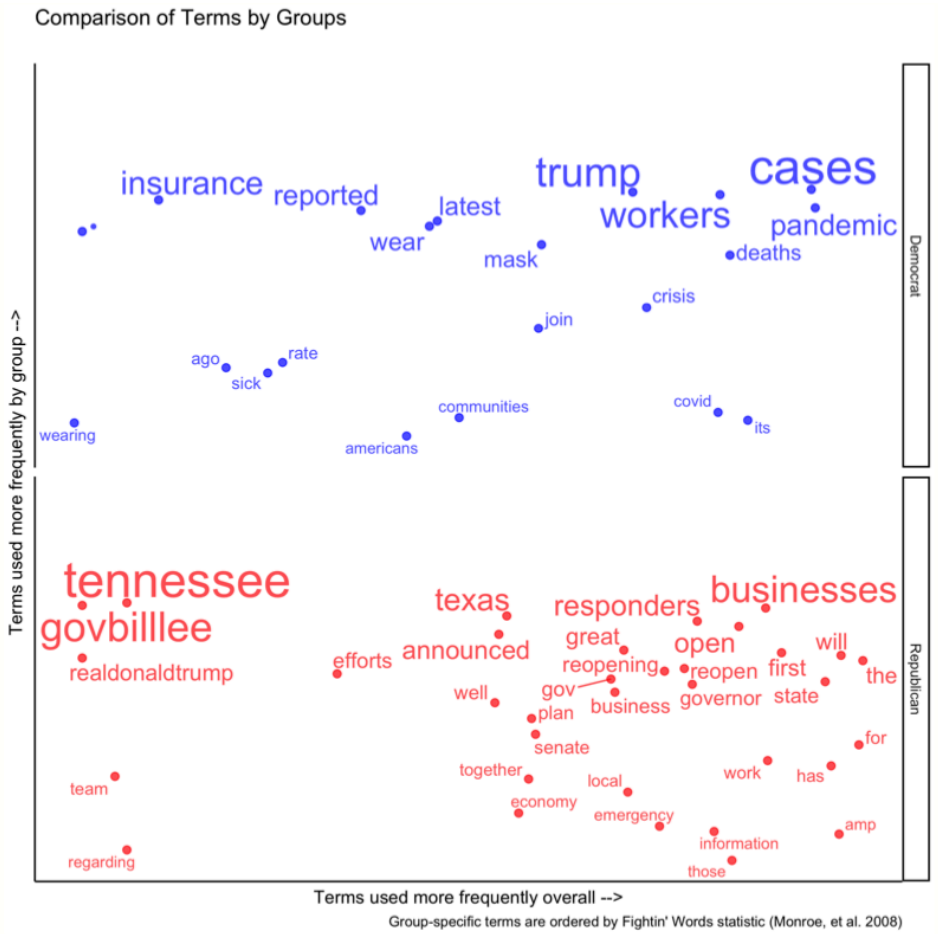
$$\sigma^2(\hat{\delta_w}^{(i-j)}) \approx \frac{1}{f_w^i + \alpha_w} + \frac{1}{f_w^j + \alpha_w}$$

The final statistic of interest (zeta or z-score):

$$\hat{\zeta_w}^{(i-j)} = \frac{\hat{\delta_w}^{(i-j)}}{\sqrt{\sigma^2(\hat{\delta_w}^{(i-j)})}}$$

Discriminating Words

Tweets about the pandemic: Democrats vs. Republicans



Summary

Counting theoretically-relevant keywords can be highly effective

Dictionaries are increasingly less useful but still serve useful roles

Keywords that discriminate between groups can be useful

Guide coding

VADER in Python (link) and Fightin' Words in R (link)