Text Mining

UC3M/IC3JM Graduate Workshop, Fall 2024

Patrick Kraft

2024-12-12

Outline for Today

1. Structural Topic Models
2. Text Classification
3. Advanced Topics: Neural Nets, Word Embeddings, and Transformers

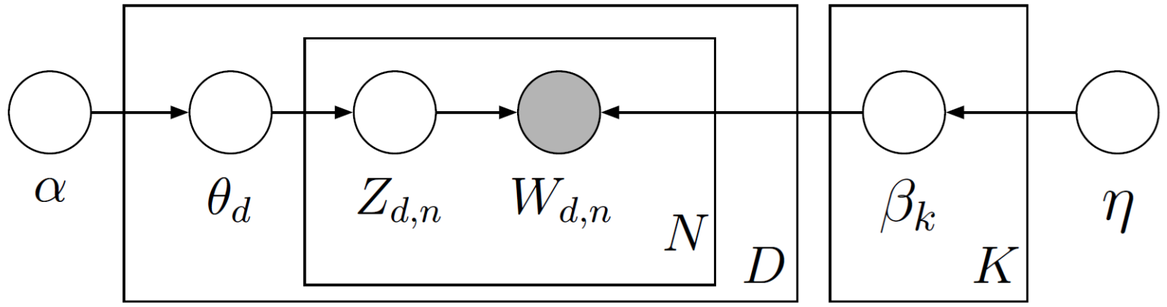
1. Structural Topic Models

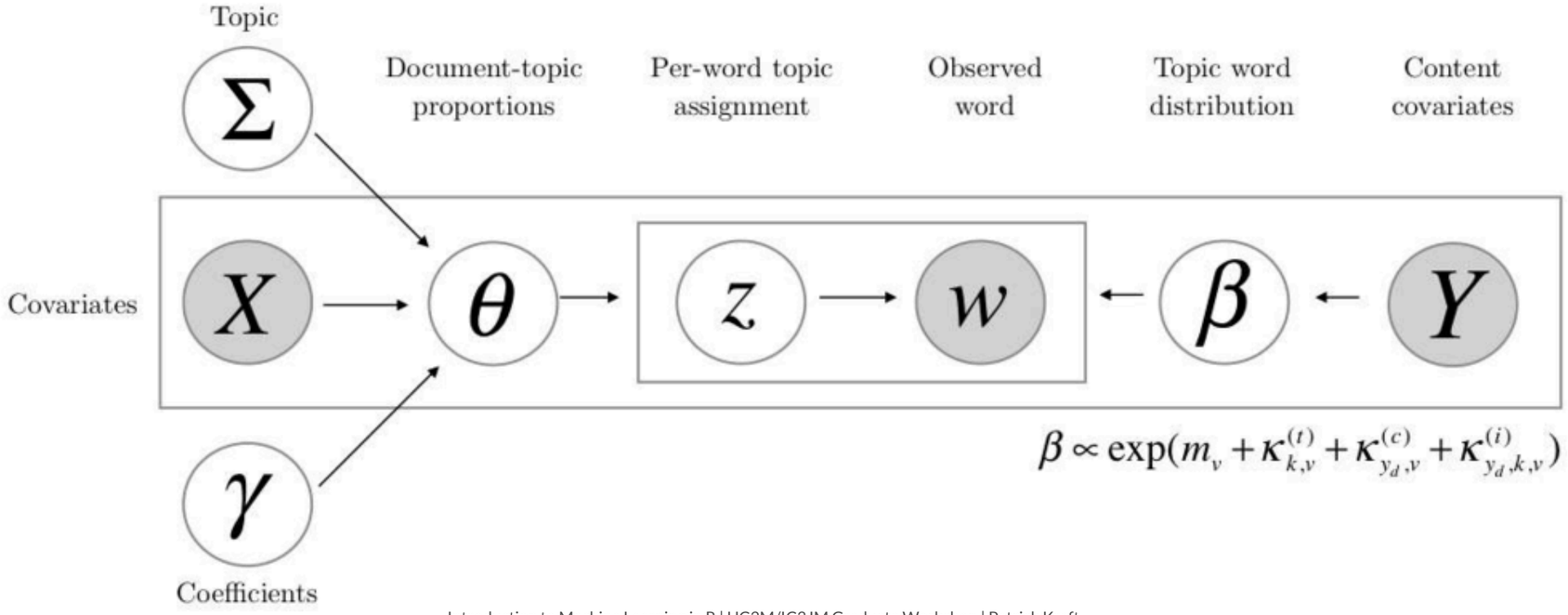
Structural Topic Model (Roberts, Stewart, Tingley, Lucas, et al. 2014; Roberts, Stewart, and Tingley 2014)

STM = LDA + contextual information.

- **topic prevalence** varies by covariates
  - e.g.women may report issues with depression more than men do.
- **topic content** varies by covariates
  - e.g. young people (X = 0) may talk about depression differently to the way old people (X = 1) talk about it.
- Including covariates allows for
  a. more **accurate estimation** and
  b. better **interpretatability**

Compare: Plate Diagrams

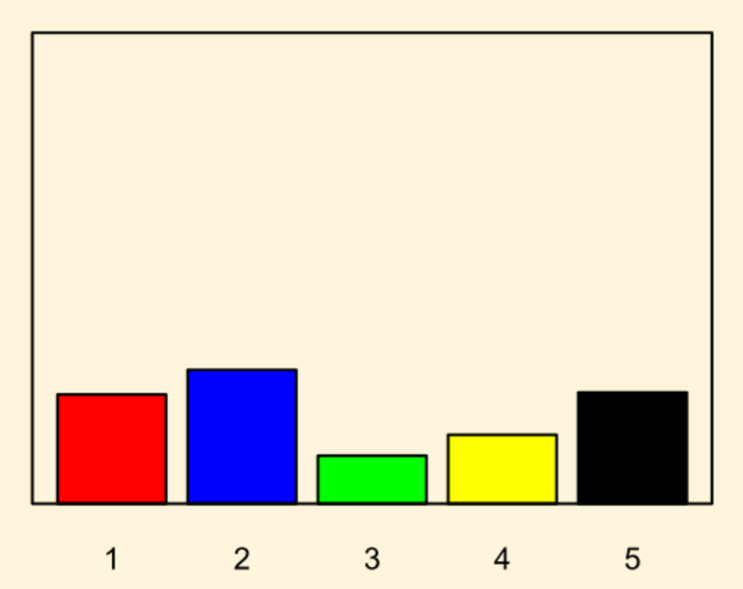




Compare: Per Document Topic Distribution

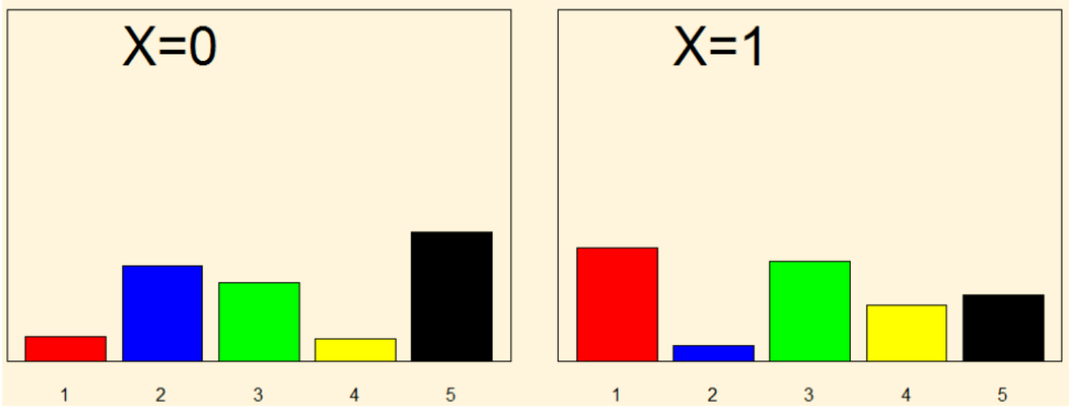
LDA

- each document has some topic distribution.



STM

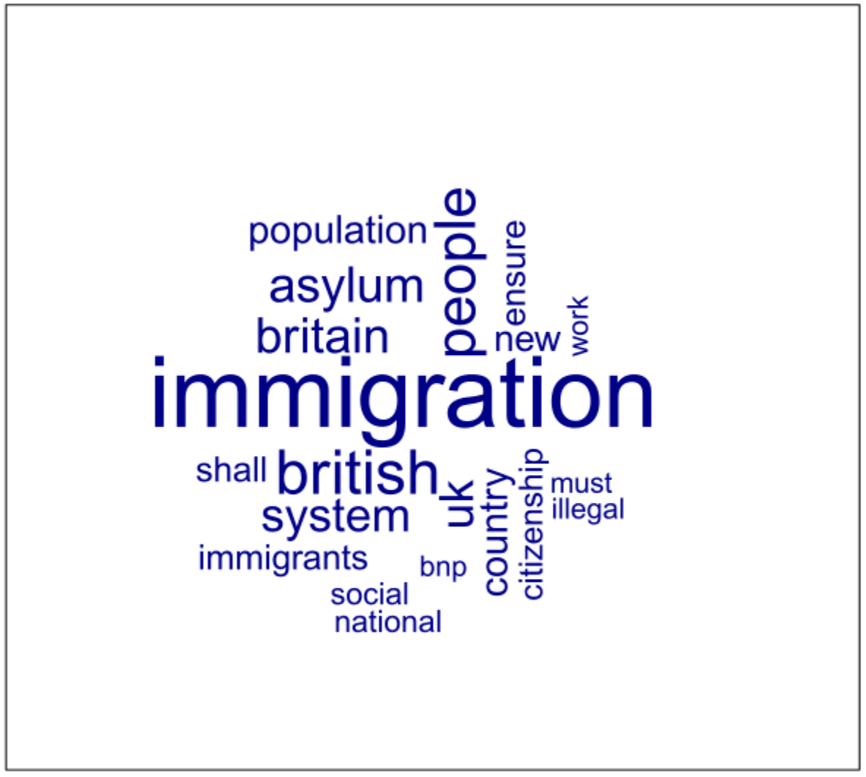
- topic distribution ('prevalence') is a function of the document metadata. e.g. perhaps male author (X = 0) documents have different topics relative to female (X = 1) author docs



Compare: Per Topic Word Distribution (β)

LDA

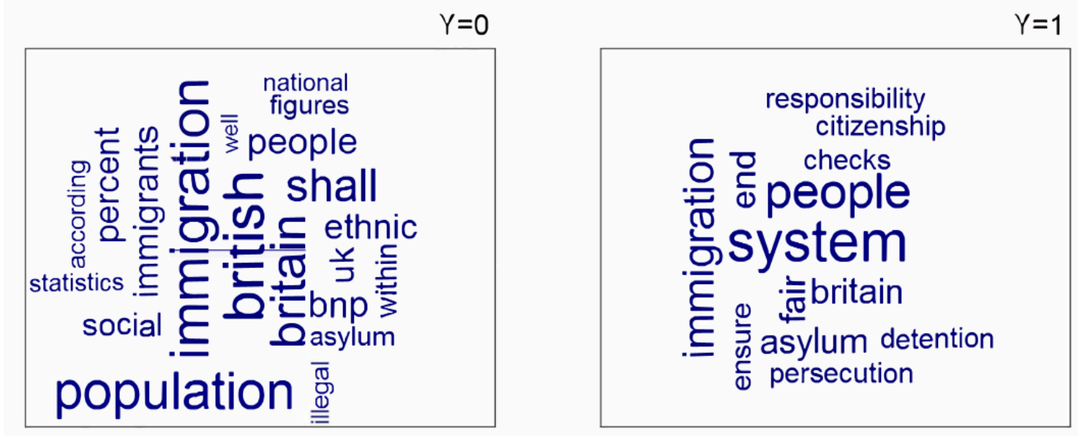
- topic ('immigration') has a given distribution over words.



STM

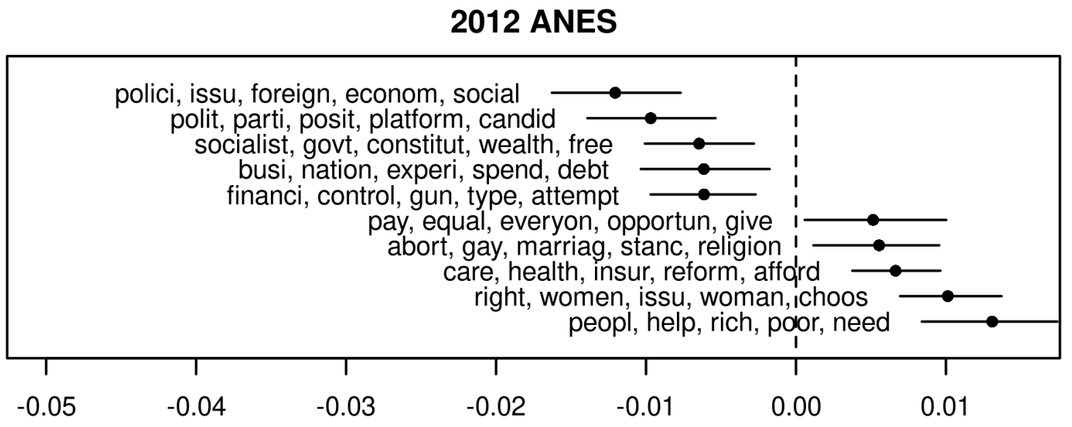
- word distribution ('content') is a function of the document metadata

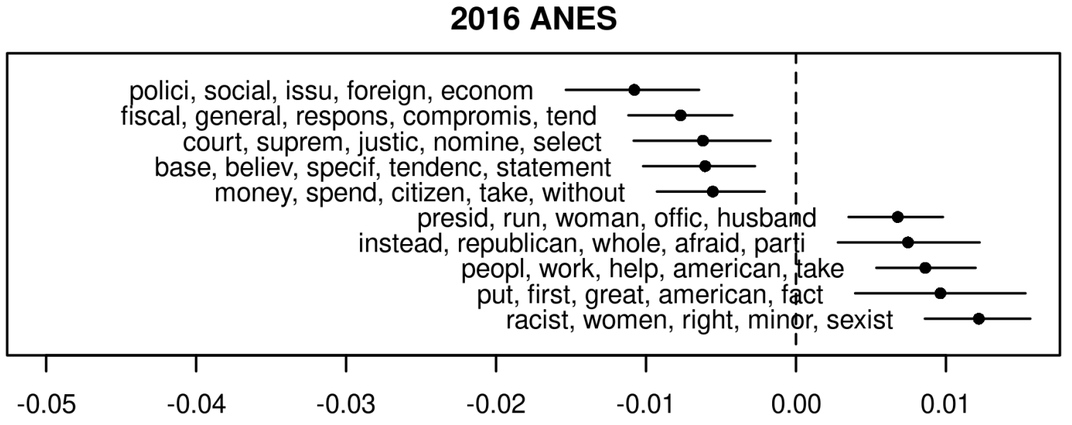
e.g. perhaps right parties (Y = 0) talk about a given topic differently to left (Y = 1) parties. In practice, content needs to a single discrete variable.

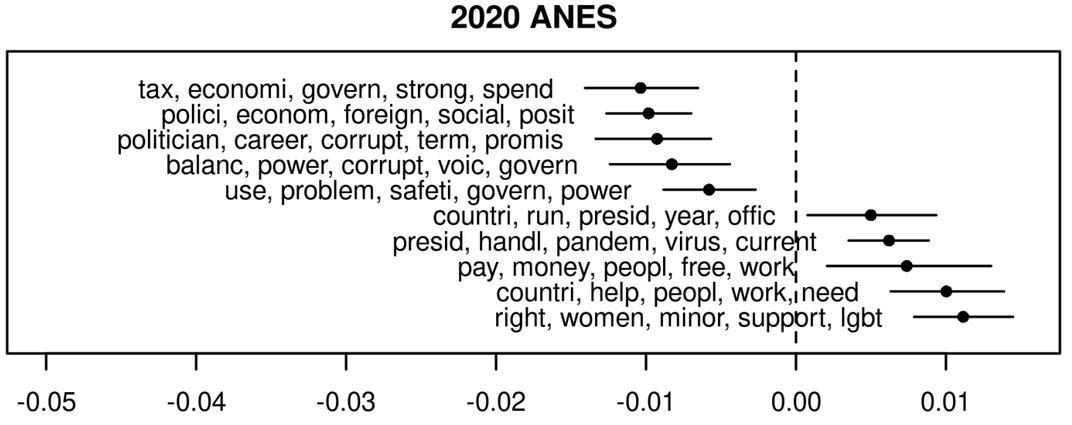


Example: Gender Differences in Open-Ended Responses

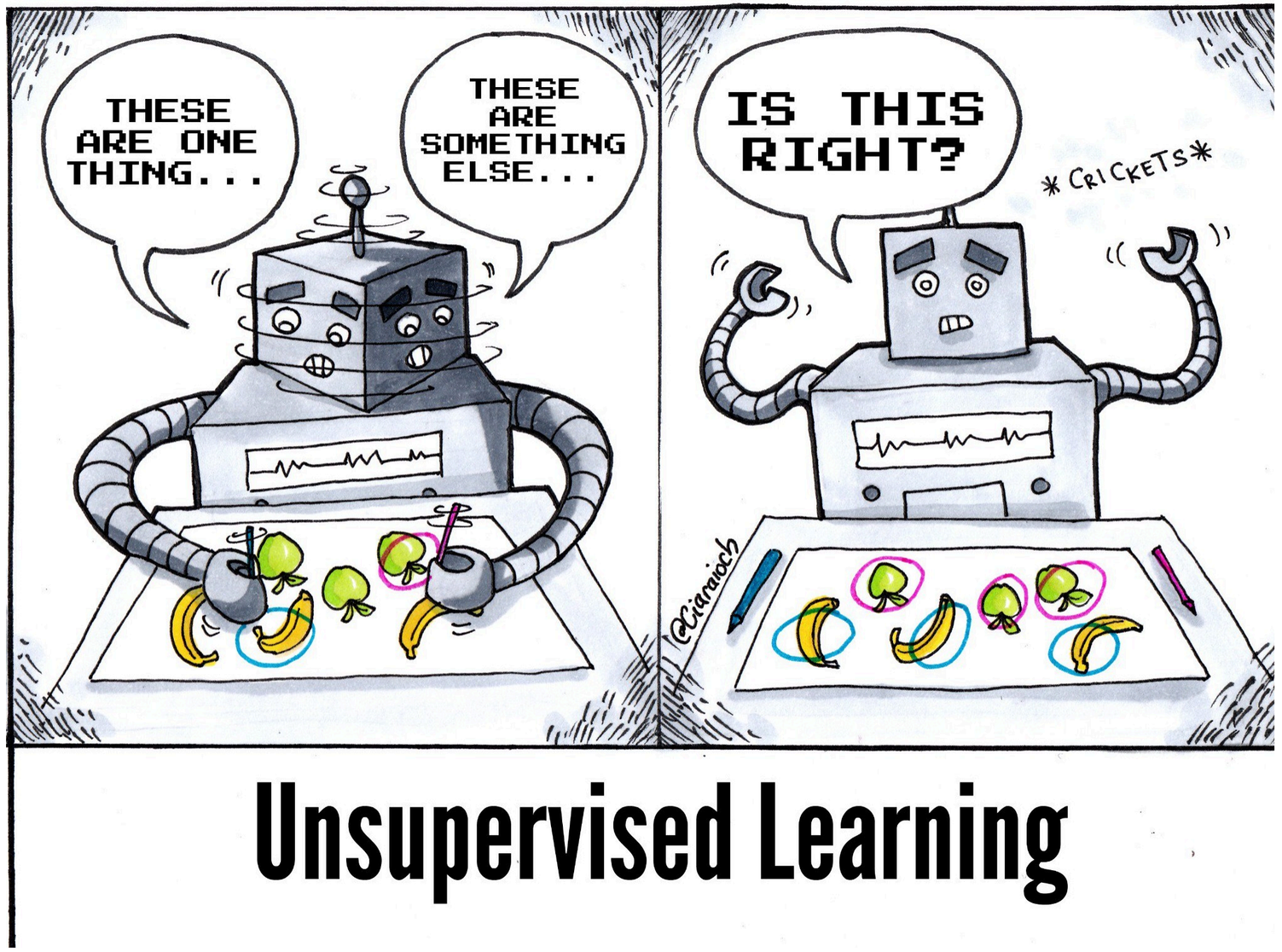
- Kraft (2024): “Overall, the results indicate that gender differences in conventional knowledge metrics are at least partly driven by the fact that the issues women care about are not represented in standard item batteries.”
- Kraft and Dolan (2023): “In this paper, we propose an inductive framework to develop more gender-balanced knowledge batteries by including political issues that are of particular relevance to women and women’s lives.”







R Session


THESE ARE ONE THING...
THESE ARE SOMETHING ELSE...
IS THIS RIGHT?
CRICKETS
@ciaraioch
Unsupervised Learning

2. Text Classification

Text Classification: Keyword counting

- Count instances of a particular word or phrase

- Illustration:
  - Research question: How does press coverage of US presidential politics change over time?
  - Research strategy: Count number of articles that use the phrase “White House”

→ Very simple, but it generates many false positives and false negatives

Text Classification: Dictionary methods

1. Identify keywords that discriminate well between categories

2. Possibly, associate scores β with the keywords (e.g., +1 = positive tone, -1 = negative tone)

3. Count keywords and calculate aggregated score for each text

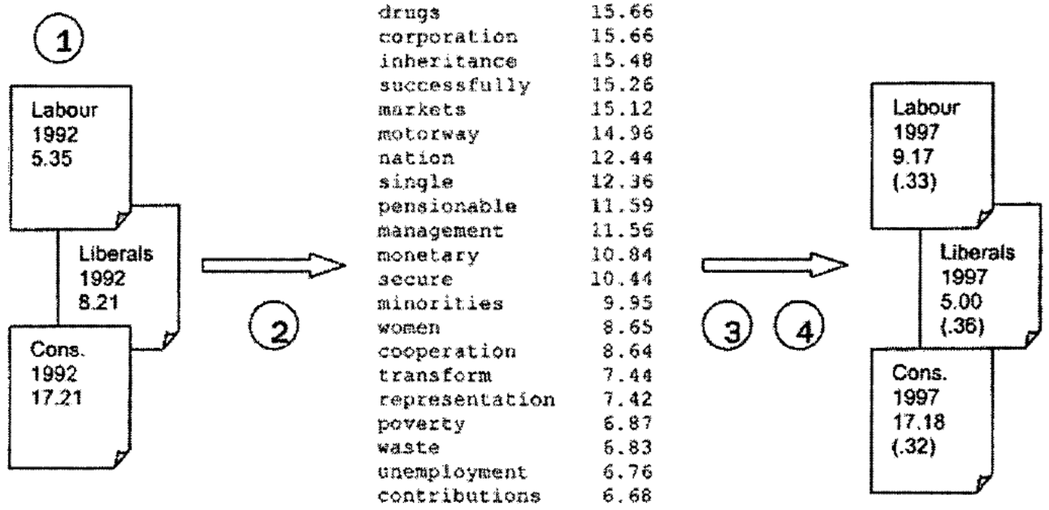
For example:

$$\text{aggregate score} = \beta_1 \text{keyword}_1 + \ldots + \beta_p \text{keyword}_p$$

→ A generalization of keyword counting, using more than one word

Research example: Scaling with Wordscores (Laver, Benoit, and Garry 2003)

FIGURE 1. The Wordscore procedure, using the British 1992–1997 manifesto scoring as an illustration



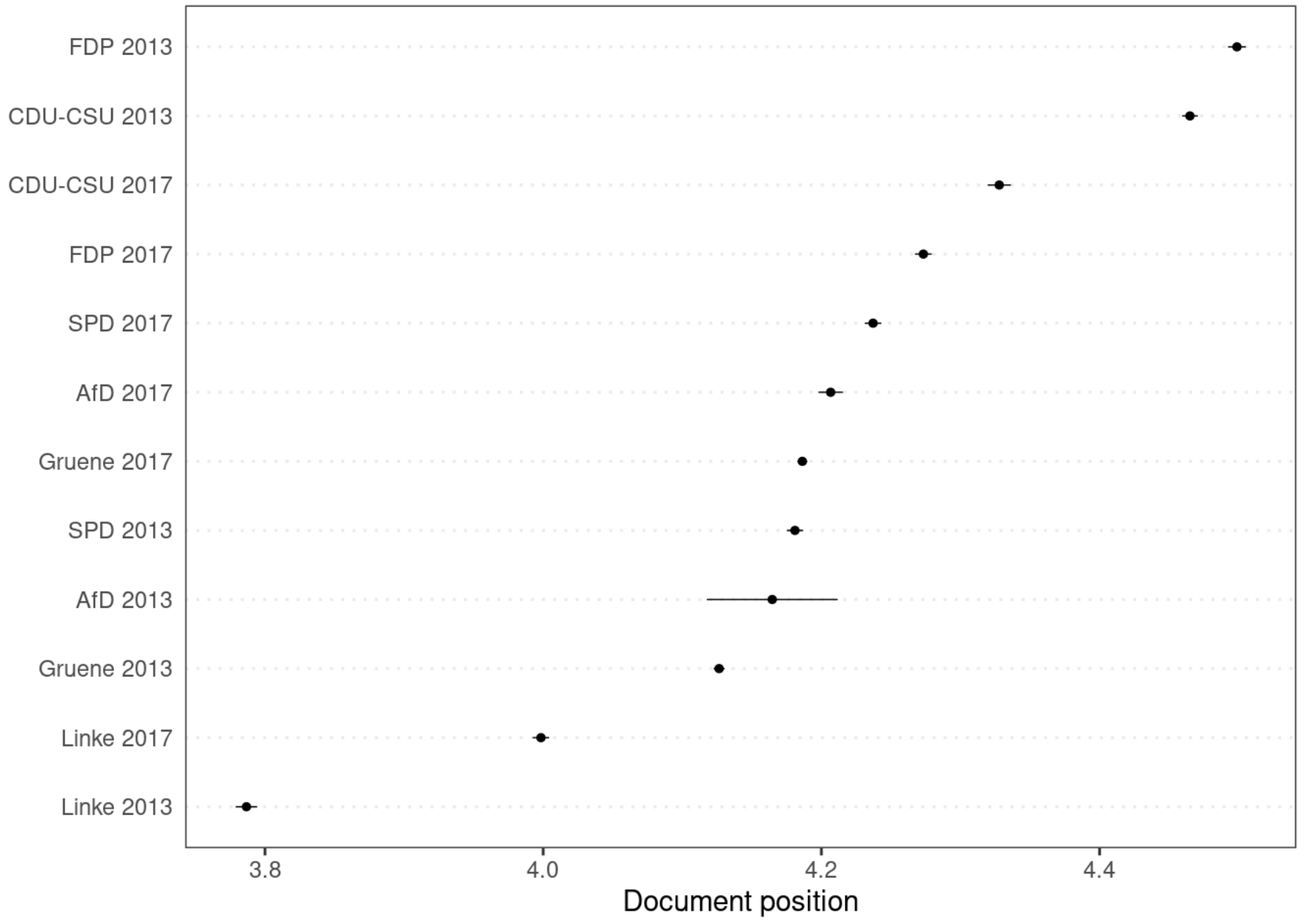
Step 1: Obtain reference texts with a priori known positions (`setref`)
Step 2: Generate word scores from reference texts (`wordscore`)
Step 3: Score each virgin text using word scores (`textscore`)
Step 4: (optional) Transform virgin text scores to original metric

Note: Scores for 1997 virgin texts are transformed estimated scores; parenthetical values are standard errors. The scored word list is a sample of the 5,299 total words scored from the three reference texts.

Source: Laver et al. (2003), Fig. 1

Research example: Scaling with Wordscores (Laver et al. 2003)

- German party manifestos (2013, 2017)



Source: https://tutorials.quanteda.io/machine-learning/wordscores/

Text Classification: Supervised Learning

- **Step 1**: Code/label a training set
  - Define a concept of interest
  - Create labeling instructions!
  - Code/label the training data
- **Step 2**: Train SML classifier
  - Can use the document-term matrix as input, along with available metadata
  - Identification of keywords for a dictionary and training a classifier are closely related
- **Step 3**: Validate model output
- **Step 4**: Classify (remaining) documents

Step 1: Code/label a training set

- Create an unambiguous coding/labeling scheme
  - Codebook can be pre-tested and revised
  - Check inter-coder reliability
  - Select documents to be coded/labeled: random sampling
  - Rule of thumb: 500 documents for manual coding? (the more the better)
  - Should always use at least 2 coders for same texts, 3 is better

Step 1: Inter-coder reliability (1)

Cohen's Kappa

- Measures the agreement between two raters
- Each rater classifies N items into C categories
- From 0 (no agreement) to 1 (perfect agreement)
- Values above 0.6 are acceptable, above 0.8 excellent
- More raters → Fleiss' kappa

See discussion of different metrics on Wikipedia.]

$$\kappa = \frac{p_o - p_e}{1 - p_e}$$

- p_0 = relative observed agreement among raters
- p_e = hypothetical probability of chance agreement

Step 1: Inter-coder reliability (2)

- Two annotators, Alice and Bob, label 100 social media posts as “Positive” or “Negative” sentiment:

Category	Alice: Positive	Alice: Negative	Total
Bob: Positive	60	10	70
Bob: Negative	15	15	30
Total	75	25	100

Calculating Cohen’s Kappa:

1. **Observed Agreement (Po):** This is the proportion of times Alice and Bob agreed.
   - `Po = (60 + 15) / 100 = 0.75`
2. **Expected Agreement (Pe):** This is the proportion of agreement expected by chance.
   - Probability of both saying “Positive”: `(75/100) * (70/100) = 0.525`
   - Probability of both saying “Negative”: `(25/100) * (30/100) = 0.075`
   - `Pe = 0.525 + 0.075 = 0.6`
3. **Cohen’s Kappa (K):**
   - `K = (Po - Pe) / (1 - Pe) = (0.75 - 0.6) / (1 - 0.6) = 0.375`

Step 1: Inter-coder reliability (3)

- **Interpretation:**
  - Kappa value of 0.375 indicates “fair” agreement between Alice and Bob (better than what expected by random chance)
  - But only values above 0.6 are acceptable, above 0.8 excellent
  - High agreement between annotators (high Kappa) ensures reliable training data
  - If Kappa is low…
    - **Refine labeling guidelines:** Make instructions clearer to reduce ambiguity
    - **Provide more training:** Help annotators better understand the task/criteria
    - **Resolve disagreements:** Discuss differing labels and reach a consensus

Step 2: Train SML classifier (1)

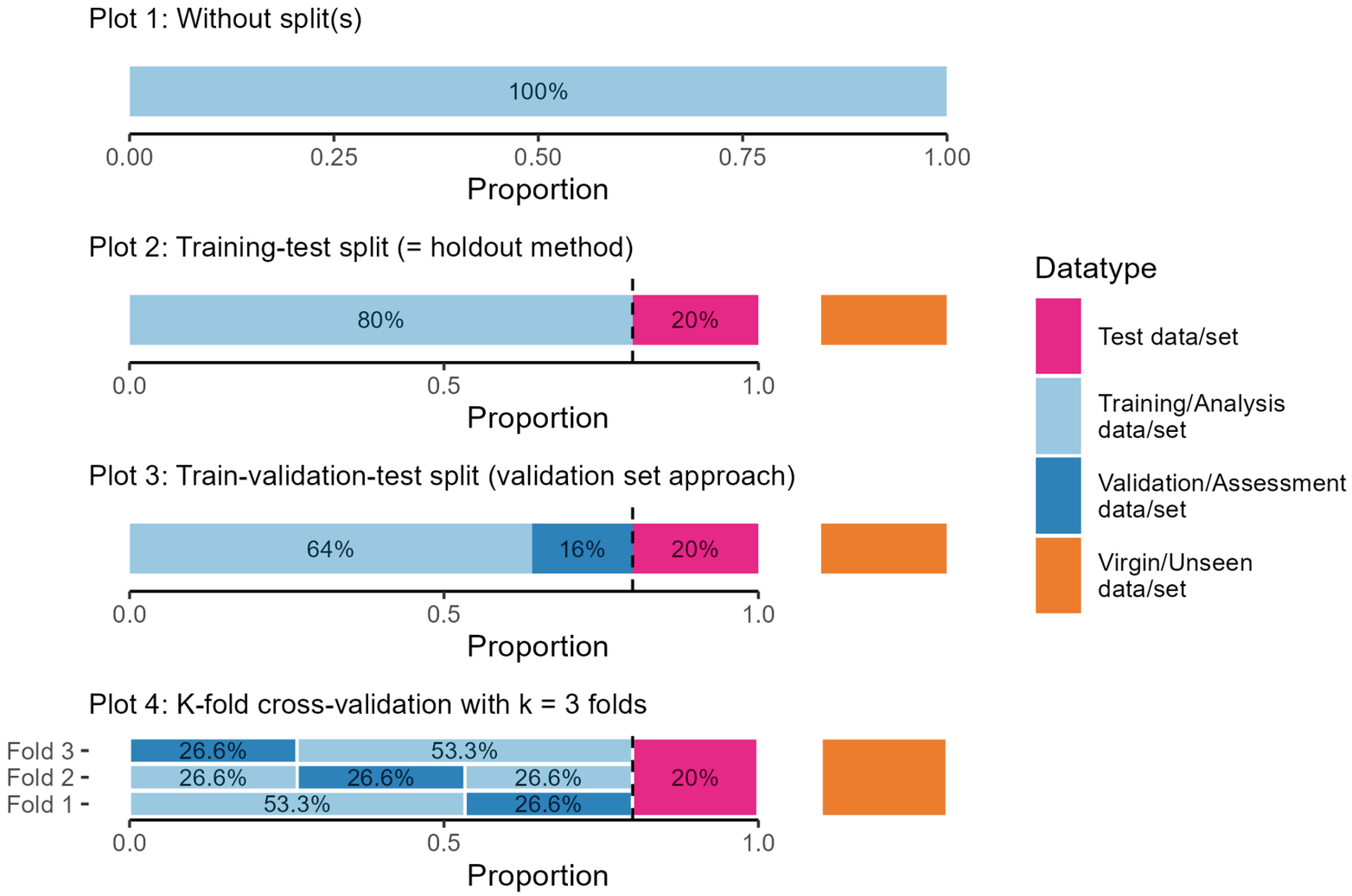
1. Separate training and test set
   - Usually, training set is 70-80% of labelled data

2. Apply model on training set
   - Lasso Regression, Random Forests, Gradient Boosting, LLMs, …

3. Learn/tune model such that it minimizes classification error for new data (using cross-validation)

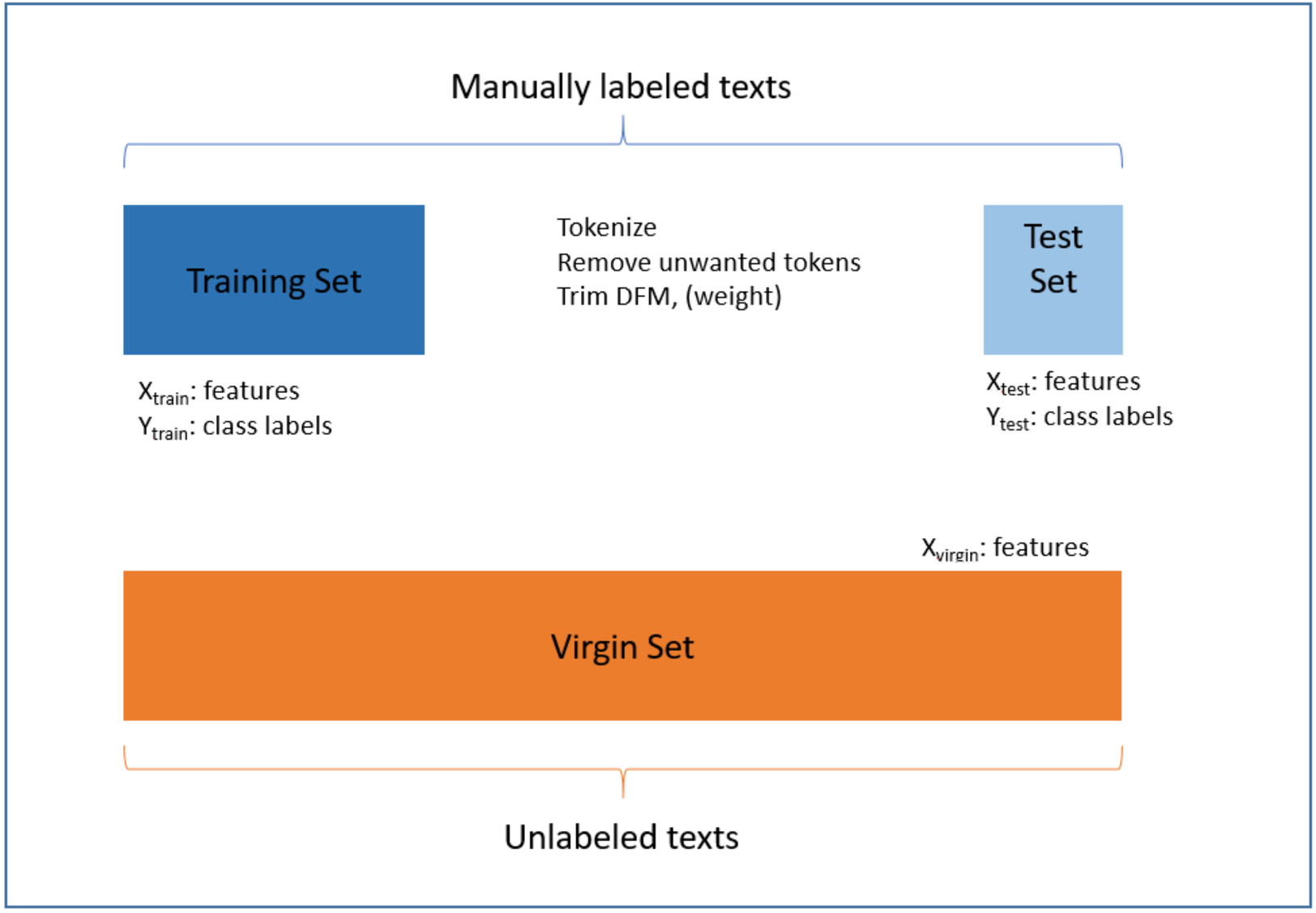
4. Model evaluation on test set

Step 2: Train SML classifier (2)

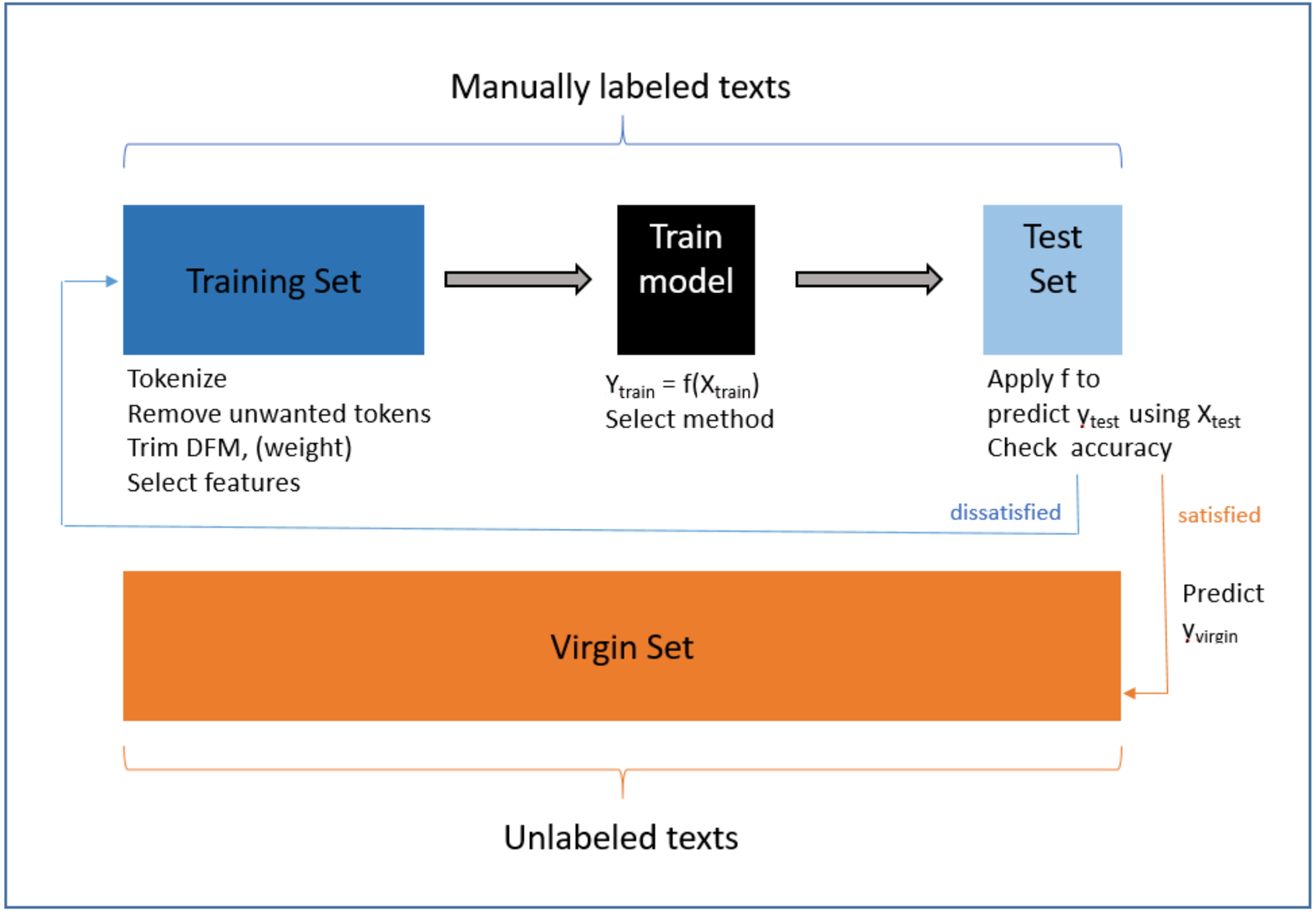


Note: Graph visualizes different evaluation protocols and what percentage of the data is used for training, validation and testing; Plot 3: Assigns 20% to the test data, whereby the remaining 80% are split 20/80 into validation data (16% of all data) and training data (64% of all data); Plot 4: 3-fold cross-validation randomly splits the full training data (here 80% of all data) 3 times, always keeping one third of the data for validation; © Paul C. Bauer

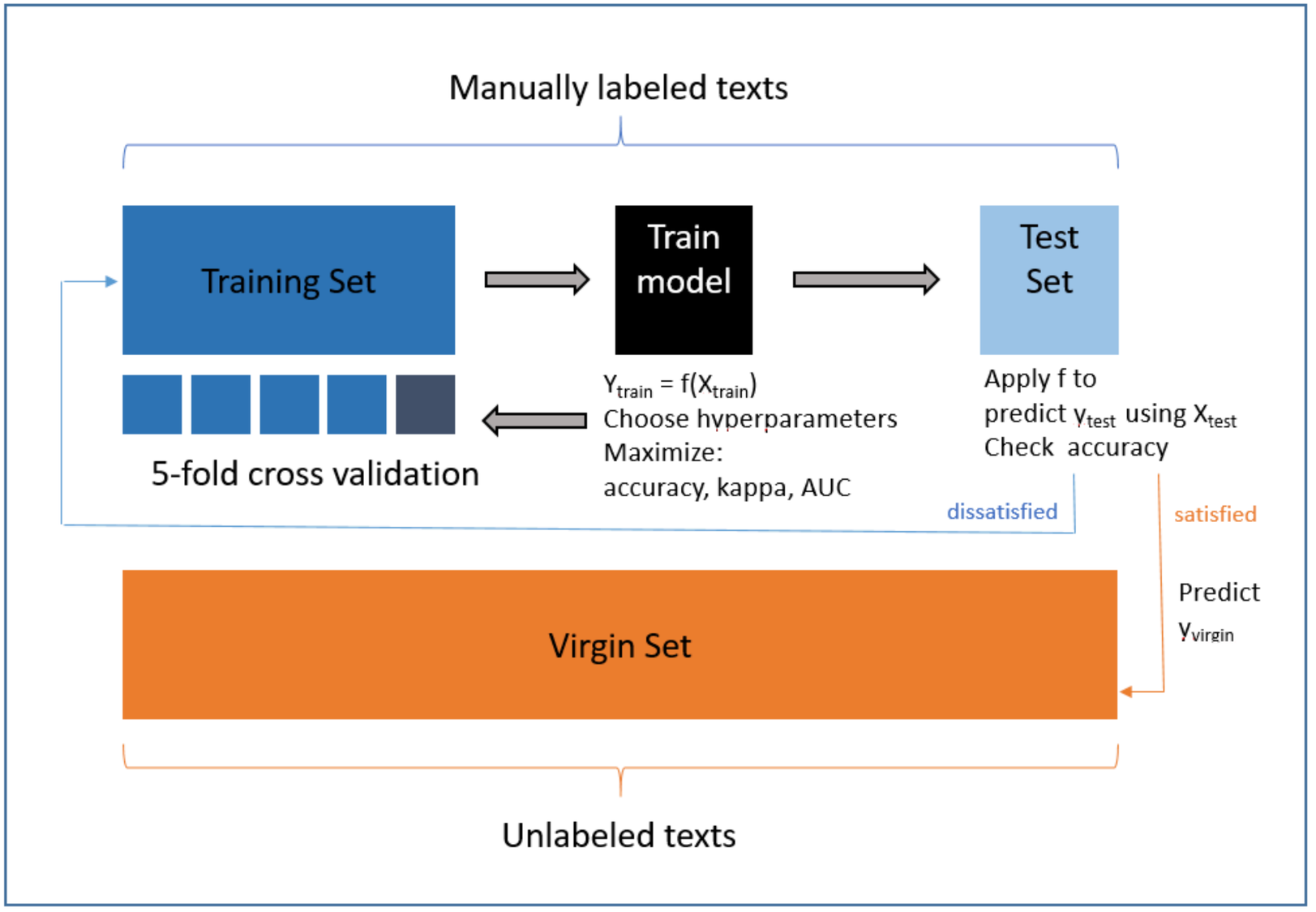
Step 2: Train SML classifier (3)



Step 2: Train SML classifier (4)



Step 2: Train SML classifier (5)



Step 3: Validate Model Output

		Predicted class		
		− / Null / 0	+ / Non-null / 1	Total
True class	− / Null / 0	True Neg. (TN)	False Pos. (FP)	N
	+ / Non-null / 1	False Neg. (FN)	True Pos. (TP)	P
	Total	N*	P*	

Source: modified from James et al. (2013), Ch. 4.4.3, Table 4.6, p. 148

Name	Definition	Synonyms
False Pos. rate	FP/N	Type I error, 1−Specificity
True Pos. rate	TP/P	1−Type II error, power, sensitivity, recall
Pos. Pred. value	TP/P*	Precision, 1−false discovery proportion
Neg. Pred. value	TN/N*	

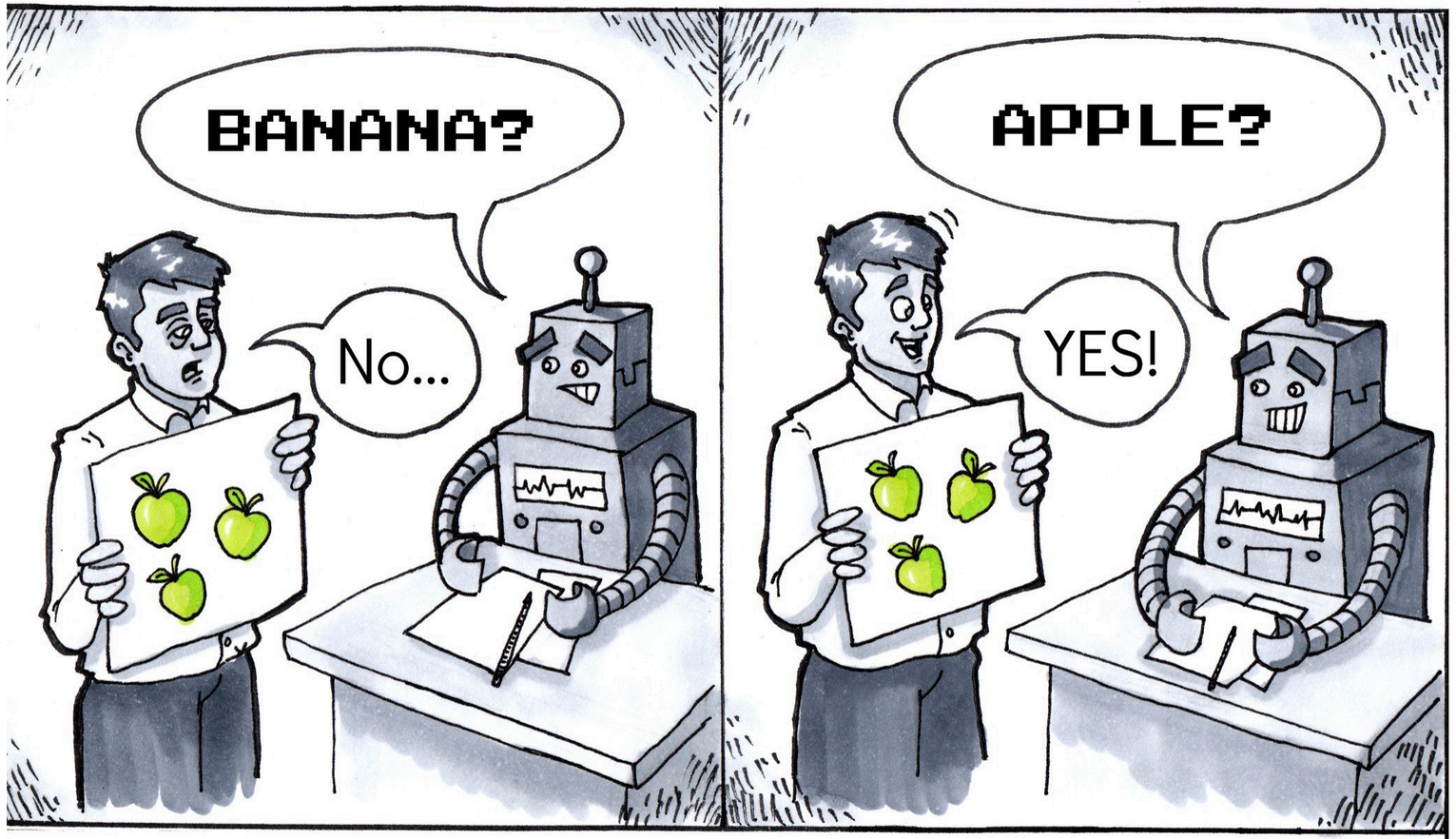
TABLE 4.7. *Important measures for classification and diagnostic testing, derived from quantities in Table 4.6.*

Source: James et al. (2013), Ch. 4.4.3, Table 4.7, p. 149

Principles of Measurement (Grimmer, Roberts, and Stewart 2022)

- Measures (e.g., of populism, racism) should have clear goals
  - Define scope and purpose of the measure
- Source material should always be identified and ideally made public
  - Be specific about source texts, so that the meaning of the measurement is well understood
- The coding process should be explainable and reproducible
  - Provide complete, easy-to-understand documentation how the measure is constructed (codebook for human annotators, model/software description for automated coding)

R Session

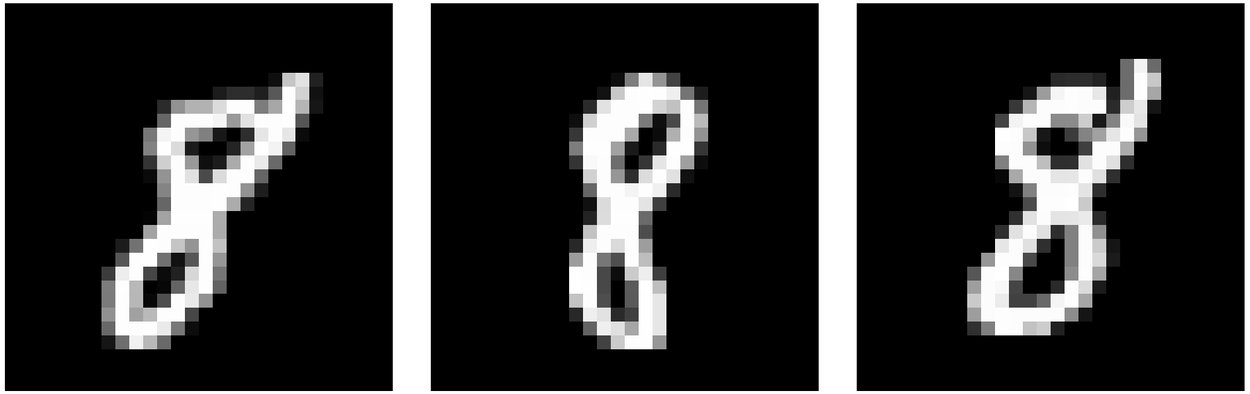


Supervised Learning

Advanced Topics: Neural Nets, Word Embeddings, and Transformers

Neural Networks: Introduction

- Consider the following example: we want to train a machine learning model that **recognizes images of handwritten digits**. Suppose we have a training dataset of 70.000 labeled images that look like this:

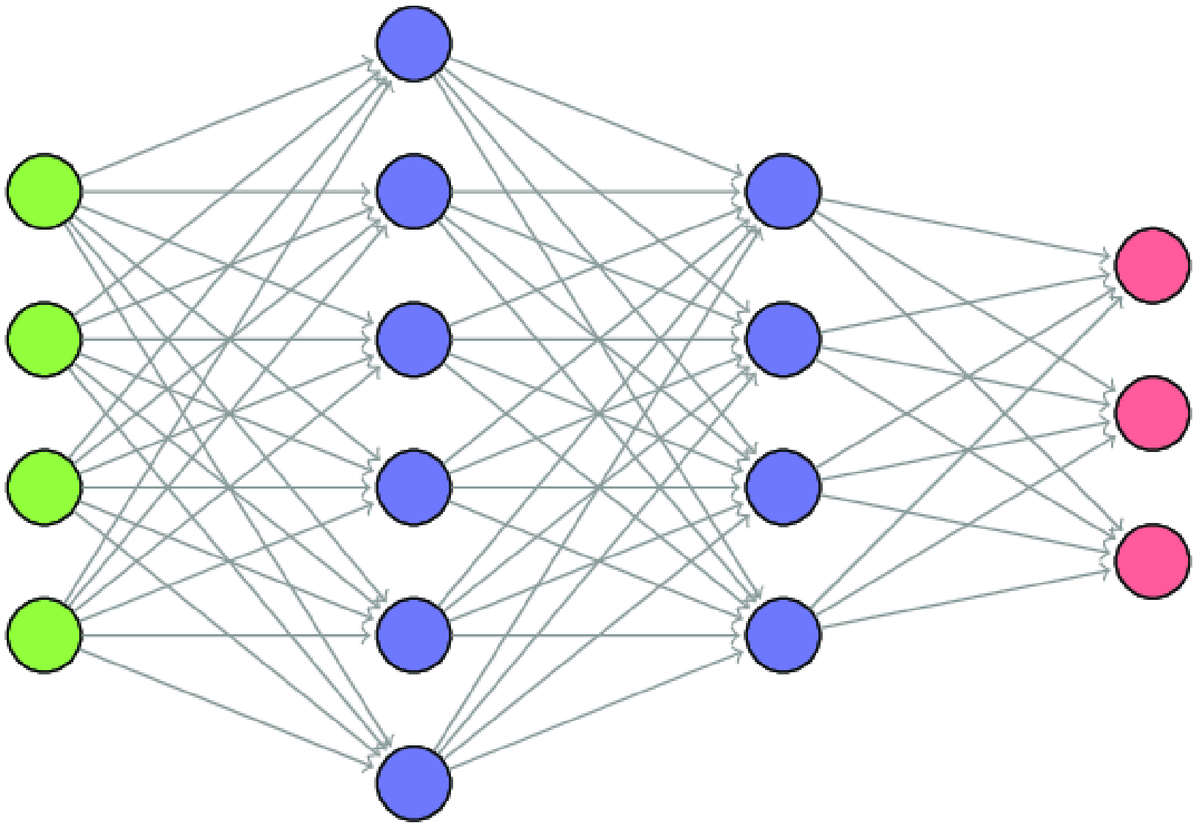


- Recognizing that the images above all represent the digit 8 is trivial for humans, but difficult for computers.

- **Neural networks** are ideal for situations where the relation between the input (intensity matrix) and the output (0-9) is complicated.

Neural Networks: Overview

In a **neural network**, input travels through a sequence of **layers**, and gets transformed into the **output tensor**.

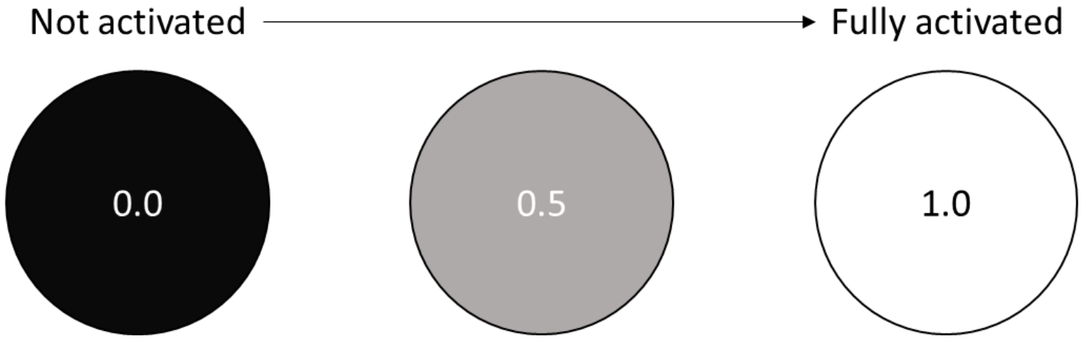


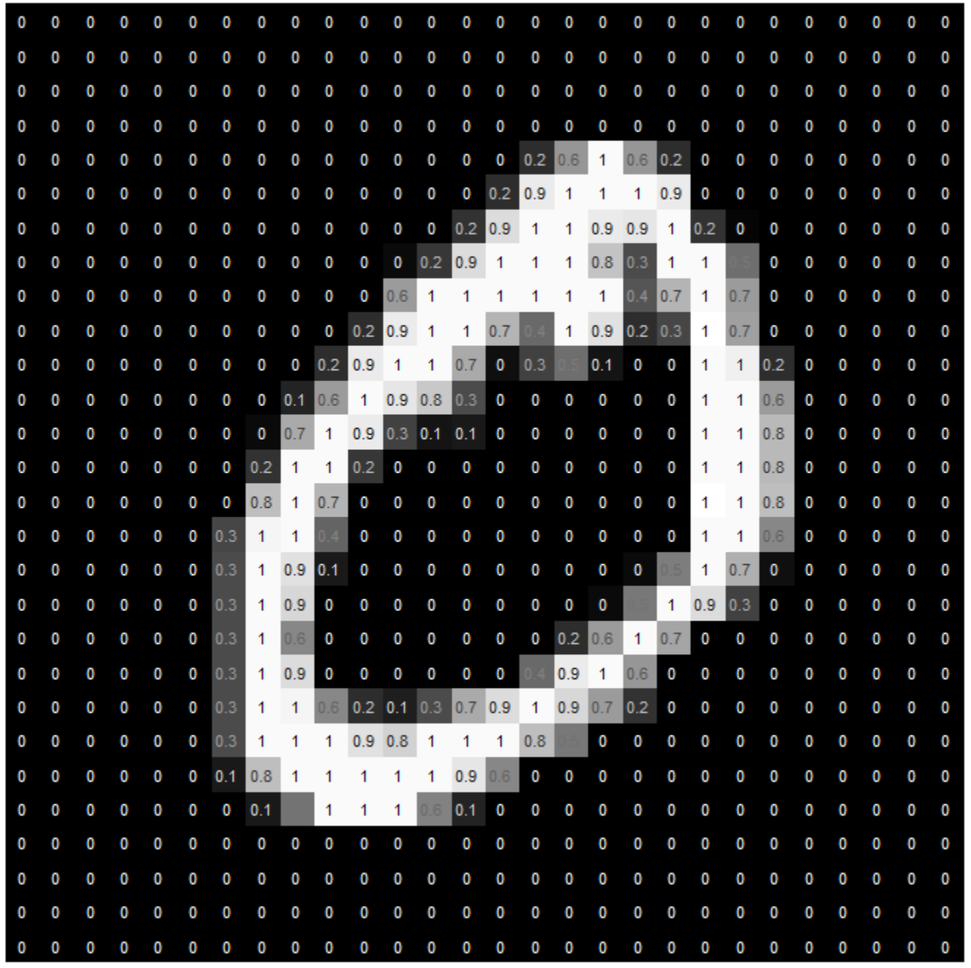
Layers consist of **nodes** and the **connections** between these nodes and the previous layer.

Neural Networks: Input Layers and Nodes

Nodes (called neurons) contain a numeric value (usually between zero and one).

- zero: the feature is not present in the data
- one: the feature is present in the data



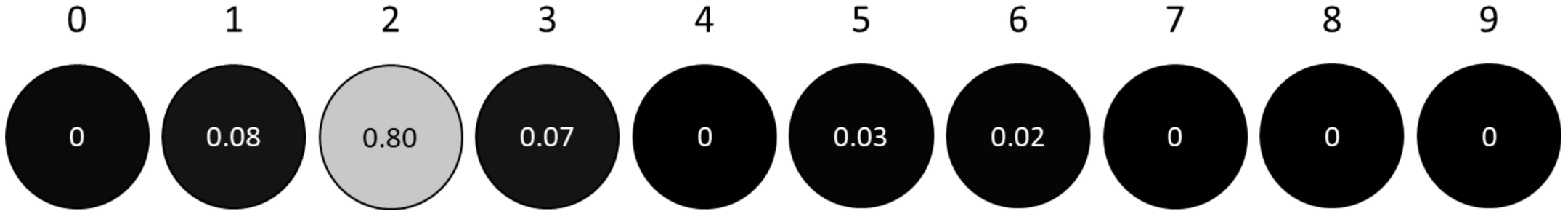


In our image recognition example, each image is stored as a 28x28 intensity matrix.

Thus, the **input layer** of our neural network consists of 784 (=28 x 28) nodes.

Neural Networks: Output Layer

- The **output layer** in our example will consist of 10 nodes (1 per digit).



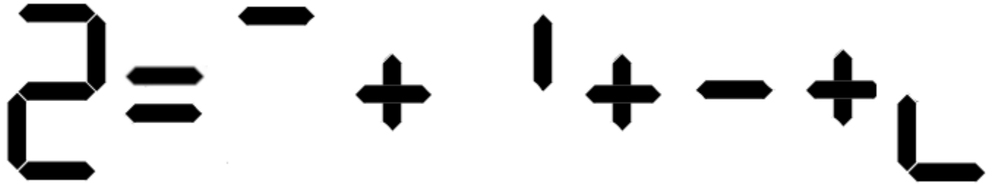
- The values of these nodes will sum to one, such that we can interpret the outcomes as **probabilities**.

- **Interpretation:** The model assigns a probability of 80% to the image representing a two.

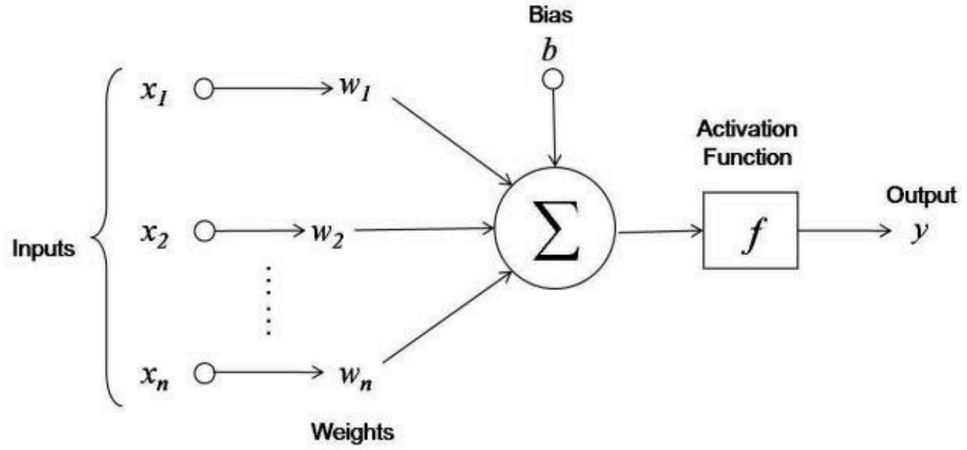
Neural Networks: Hidden Layers

- All layers between the input and output layer are called **hidden layers**.
- Nodes in the hidden layer(s) represent intermediary features that we don't explicitly define. We let the model decide the optimal features.

- Recognizing a digit is more difficult than recognizing a horizontal or vertical line. Hidden layers automatically split the problem into smaller problems that are easier to model



Neural Networks: Activation

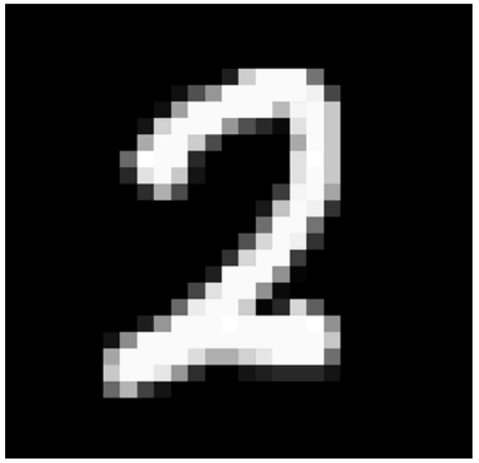


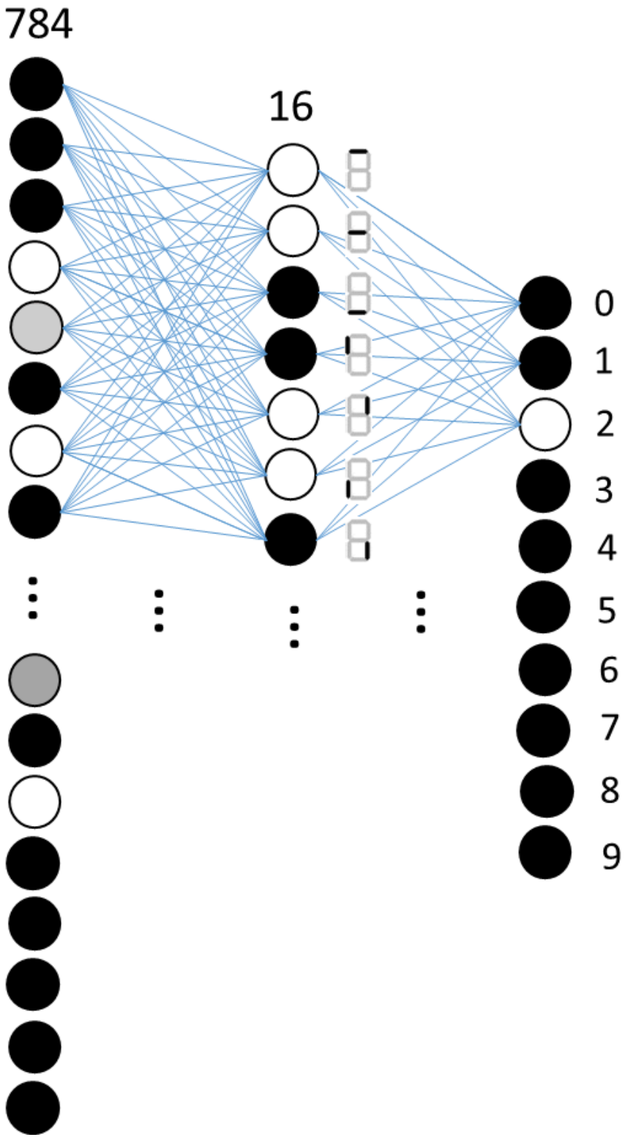
Source: Arthur Arnx

- First, the model assigns a **weight** to each connection and computes their **linear combination** for a given node:
  $z = b + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n = b + \vec{w} \cdot \vec{x}$
- An extra **bias** term is added, which can be seen as a node in the previous layer that is always activated
- Second, an **activation function** is applied (e.g., sigmoid, softmax, etc.), which adds non-linearity in the model. Without the activation function the model would be identical to linear regression:
  $y = f(b + \vec{w} \cdot \vec{x})$

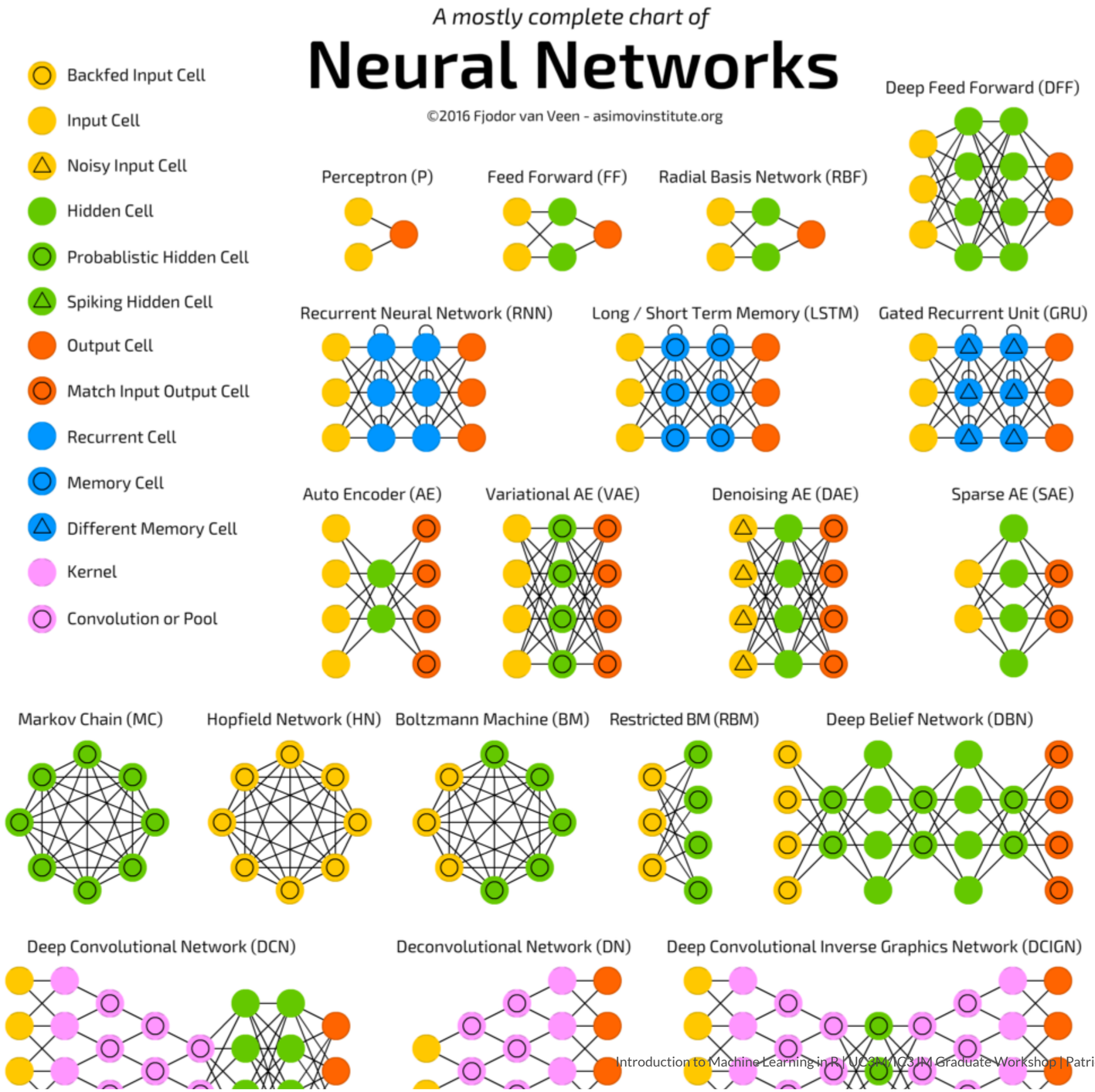
Neural Networks: Connecting Input and Output Layers

- A pattern of connections in the first layer will activate certain features in the hidden layer.
- These features in the hidden layer will in turn activate cells in the output layer.
- The output cell with the highest value is our predicted outcome.
- Question: How is the value of cells in the hidden and output layer determined?





Types of Neural Networks


A mostly complete chart of
Neural Networks
©2016 Fjodor van Veen - asimovinstitute.org
Backfed Input Cell
Input Cell
Noisy Input Cell
Hidden Cell
Probablistic Hidden Cell
Spiking Hidden Cell
Output Cell
Match Input Output Cell
Recurrent Cell
Memory Cell
Different Memory Cell
Kernel
Convolution or Pool
Perceptron (P)
Feed Forward (FF)
Radial Basis Network (RBF)
Deep Feed Forward (DFF)
Recurrent Neural Network (RNN)
Long / Short Term Memory (LSTM)
Gated Recurrent Unit (GRU)
Auto Encoder (AE)
Variational AE (VAE)
Denoising AE (DAE)
Sparse AE (SAE)
Markov Chain (MC)
Hopfield Network (HN)
Boltzmann Machine (BM)
Restricted BM (RBM)
Deep Belief Network (DBN)
Deep Convolutional Network (DCN)
Deconvolutional Network (DN)
Deep Convolutional Inverse Graphics Network (DCIGN)

Word Embeddings: Introduction

- **Distributional Hypothesis:** "A word is characterized by the company it keeps." (Firth, 1957)

- So: words that are used in the same contexts tend to be similar in their **meaning**. How can we use this insight to understand what words mean?

- If the word `liberal` shows up 'near' `privatization` in one country but 'near' `socialist` in another, we can learn something about those polities.

- We can systematically learn about **analogies** and **similarities**.

Word Embeddings: Intuition

- Predictive approach: try to predict a word from knowing its neighbors. To do this, we *learn* an **embedding vector** that captures meaning of a given word in our corpus a numerical way.
- e.g. `Word2Vec` (Mikolov et al. 2013): a powerful way to create the word vectors. Comes in two types/models, **Continuous Bag of Words** (CBOW) and **Skip-Gram**.

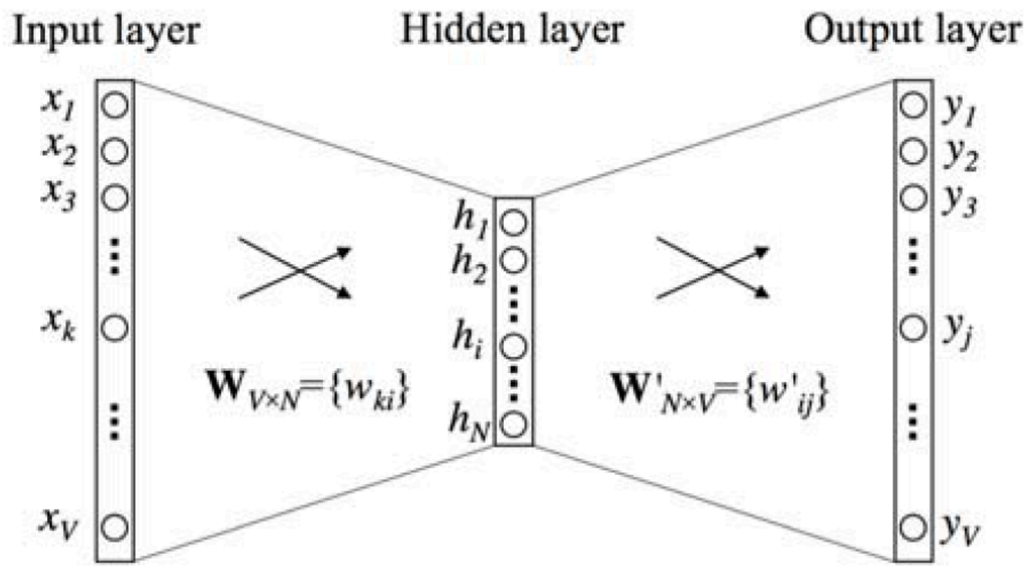
- **Continuous Bag of Words** predicts target words from source context words.
- For example, it predicts `dog` from `the quick brown fox jumped over [ ]`

→ useful in **small** datasets.

- **Skip-gram** predicts source context words from target words
- For example, it predicts `the quick brown fox jumped over [ ]` from `dog`.

→ useful in **large** datasets

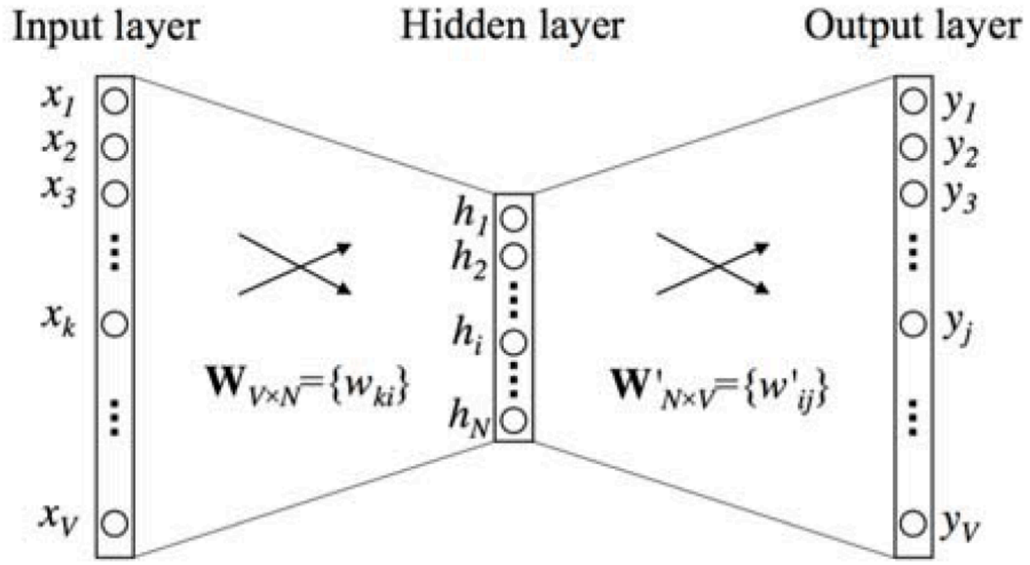
Word Embeddings: Estimation (1)



Simple CBOW Model

- We have a **vocabulary** of size V: this is the size of the input 'layer' (the context words) and the size of the output 'layer' (the target words).
- We want to create **embedding vectors**, which sum up how a given word is related to every other word.
- Our **inputs** (i.e., the context words) will be transformed via a **hidden layer** of N 'neurons'.
  - How many neurons?
    More → better at representing our inputs (bigger embedding vector). But that means we need more data.

Word Embeddings: Estimation (2)



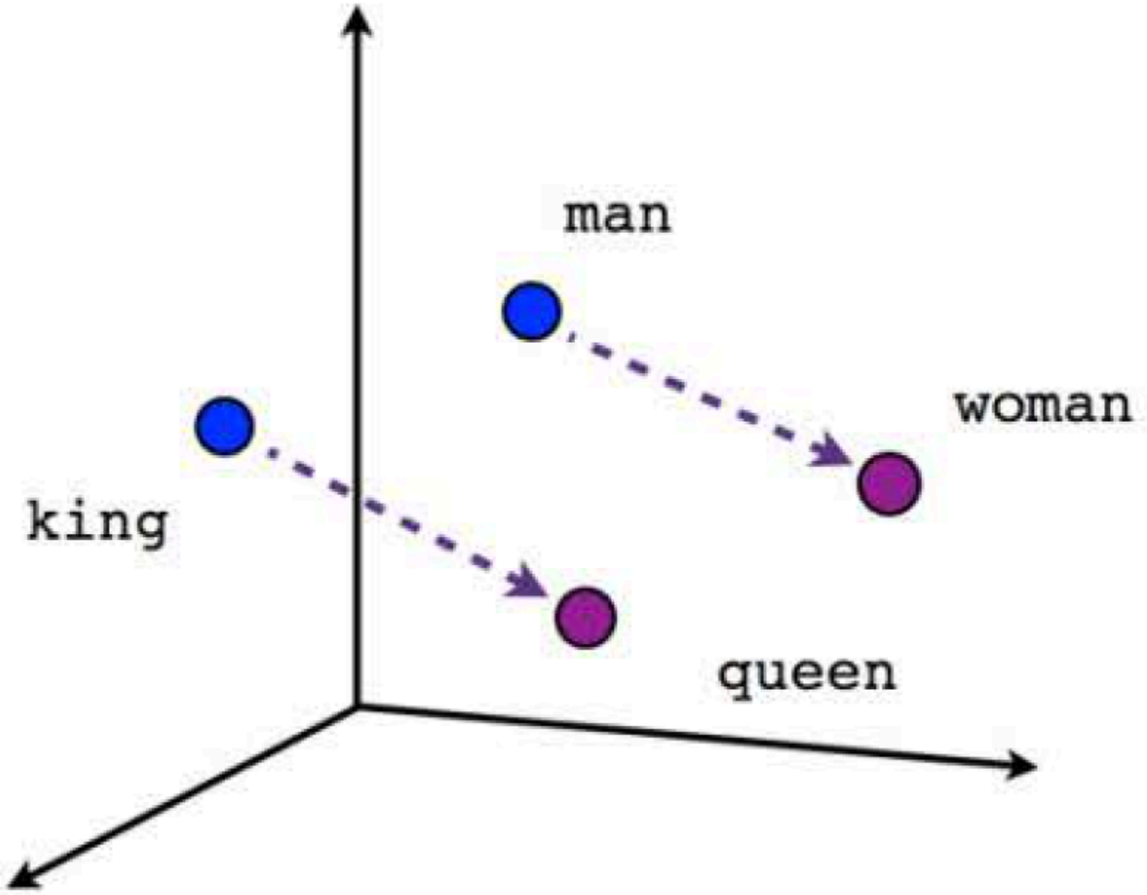
Simple CBOW Model

- We go from the **input layer** to the **hidden layer** via a weight matrix W_I of size $V \times N$—each row represents a context word.
- We go from the **hidden layer** to the **output layer** via a (different) weight matrix W_O of size $N \times V$—each column represents a target word.
- Then we'll use softmax to calculate $Pr(\text{word}_k|\text{target})$. We'll maximize this probability, and that will optimize the weight matrices.
- The **embedding vector** for a given word is a row of W_I.

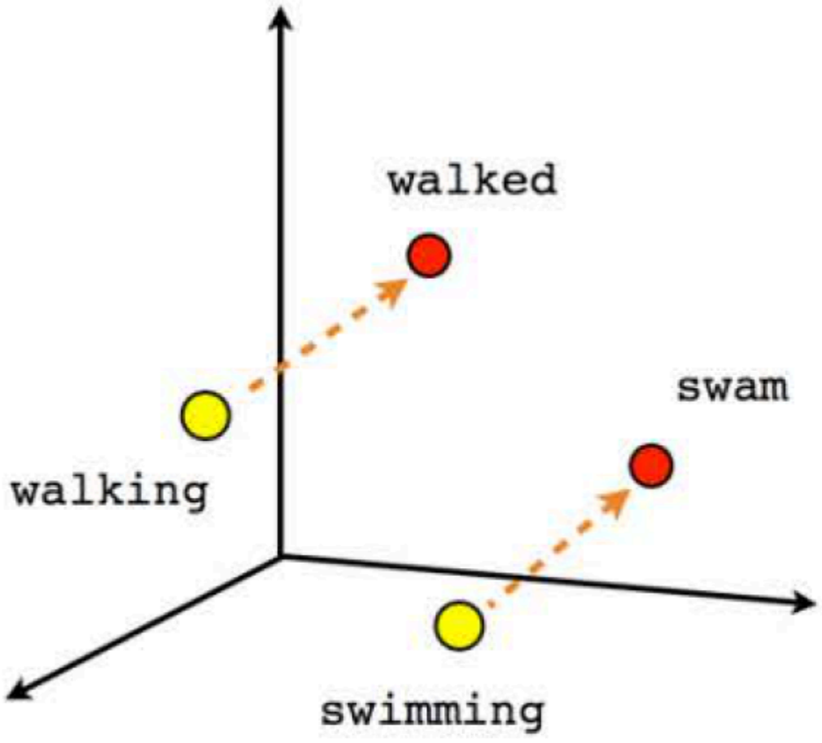
Word Embeddings: Interpretation (1)

- Once we've optimized, we can extract the word specific vectors from W_I as **embedding vectors** (equal in length to number of neurons). These real valued vectors can be used for analogies and related tasks.

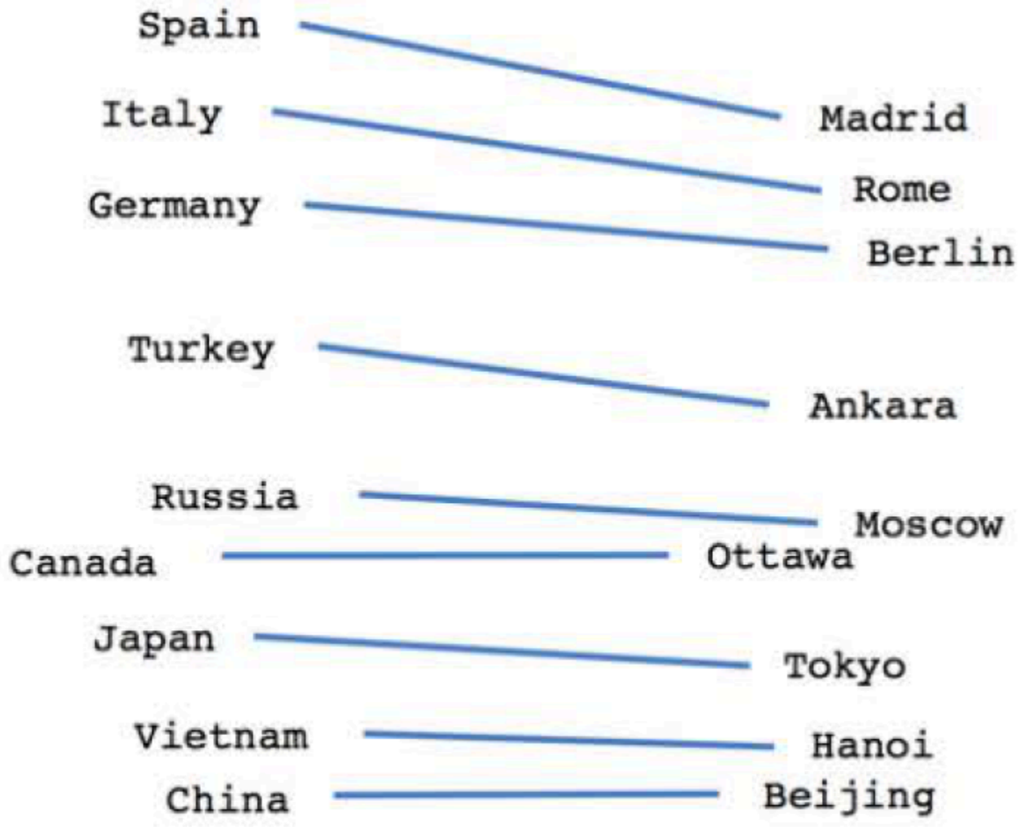
- Example: $v_{\text{queen}} - v_{\text{woman}} + v_{\text{men}} \approx v_{\text{king}}$


man
woman
king
queen

Word Embeddings: Interpretation (2)



Verb tense



Country-Capital

Word Embeddings: Comments

- **Why** does `Word2Vec` work? Area of some interest: see e.g. Levy & Goldberg in NIPS, 2014.

- Producing word embedding vectors is computationally expensive, and you need a lot of data. So **pre-trained** embeddings (esp from Wikipedia) are available. E.g.: https://wikipedia2vec.github.io/wikipedia2vec/pretrained/

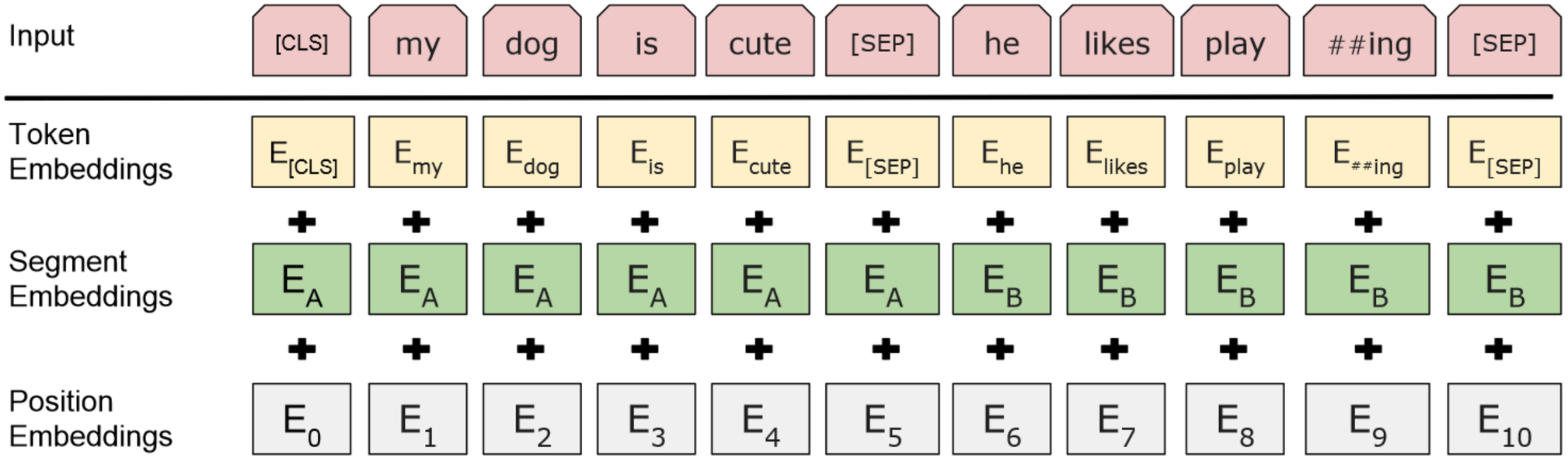
- `GloVe`, Global Vectors for Word Representation, is method that works via dimension reduction of counts of co-occurrences (rather than trying to predict, as `Word2Vec`): https://nlp.stanford.edu/projects/glove/

- There are frameworks to examine whether embeddings are statistically different (between words and/or across texts) and to make embeddings dependent on covariates (e.g., Rodman 2020; Rodriguez, Spirling, and Stewart 2023)

Beyond Word Embeddings: Transformers

Bidirectional Encoder Representations from Transformers (BERT) (Devlin 2018): state-of-the-art deep learning approaches for representing words in vectors

- Pre-trained language model which provides sentence-level and word-level numeric representations
- Creates representations that consider a given word's context



BERT input representation

Beyond Word Embeddings: Resources

- BERT-as-service
  - https://bert-as-service.readthedocs.io/en/latest/
- HuggingFace Transformers library
  - https://huggingface.co/transformers/
- From NLP to CSS Tutorial
  - https://github.com/chkla/NLP2CSS-Tutorial

References

Grimmer, Justin, Margaret E. Roberts, and Brandon M. Stewart. 2022. *Text as Data: A New Framework for Machine Learning and the Social Sciences*. Princeton University Press.

James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2013. *An Introduction to Statistical Learning: With Applications in R*. Springer.

Kraft, Patrick W. 2024. "Women Also Know Stuff: Challenging the Gender Gap in Political Sophistication." *American Political Science Review* 118(2):903–21.

Kraft, Patrick W., and Kathleen Dolan. 2023. "Asking the Right Questions: A Framework for Developing Gender-Balanced Political Knowledge Batteries." *Political Research Quarterly* 76(1):393–406. doi: 10.1177/10659129221092473.

Laver, Michael, Kenneth Benoit, and John Garry. 2003. "Extracting Policy Positions from Political Texts Using Words as Data." *American Political Science Review* 97(2):311–31.

Mikolov, T., I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. 2013. "Distributed Representations of Words and Phrases and Their Compositionality." Pp. 3111–19 in *Advances in neural information processing systems*.

Roberts, Margaret E., Brandon M. Stewart, and Dustin Tingley. 2014. "Stm: R Package for Structural Topic Models." *Journal of Statistical Software* 1:1–49.

Roberts, Margaret E., Brandon M. Stewart, Dustin Tingley, Christopher Lucas, Jetson Leder-Luis, Shana Kushner Gadarian, Bethany Albertson, and David G. Rand. 2014. "Structural Topic Models for Open-Ended Survey Responses." *American Journal of Political Science* 58(4):1064–82. doi: 10.1111/ajps.12103.

Rodman, Emma. 2020. "A Timely Intervention: Tracking the Changing Meanings of Political Concepts with Word Vectors." *Political Analysis* 28(1):87–111.

Rodriguez, Pedro L., Arthur Spirling, and Brandon M. Stewart. 2023. "Embedding Regression: Models for Context-Specific Description and Inference." *American Political Science Review* forthcoming:1–20.