# Text Analysis of Political Risk Data

COMPSS 224B: Quantitative Political Risk

---

**Iris Malone, PhD and Mark Rosenberg, PhD**

April 25, 2025

# Announcements

- Problem Set 2 Released: Due May 7
- Final Response Released: Due May 16
- PCA follow-up:
    - Scaling
    - Index Polarity

# Recap

Where We've Been:

- Data engineering means knowing your data
- There is no uniform solution to missing data
- PCA and KNN useful tools for exploration and index creation

# Recap

Where We've Been:

- Data engineering means knowing your data
- There is no uniform solution to missing data
- PCA and KNN useful tools for exploration and index creation

New Terminology:

- Missing At Random
- PCA
- Biplot
- Euclidean Distance

# Agenda

1. Motivation
2. Dictionary Methods
3. Distinctive Words
4. Embeddings
5. Clustering and Topic Modeling

# Motivation

# Computational Text Analysis Growing Popularity

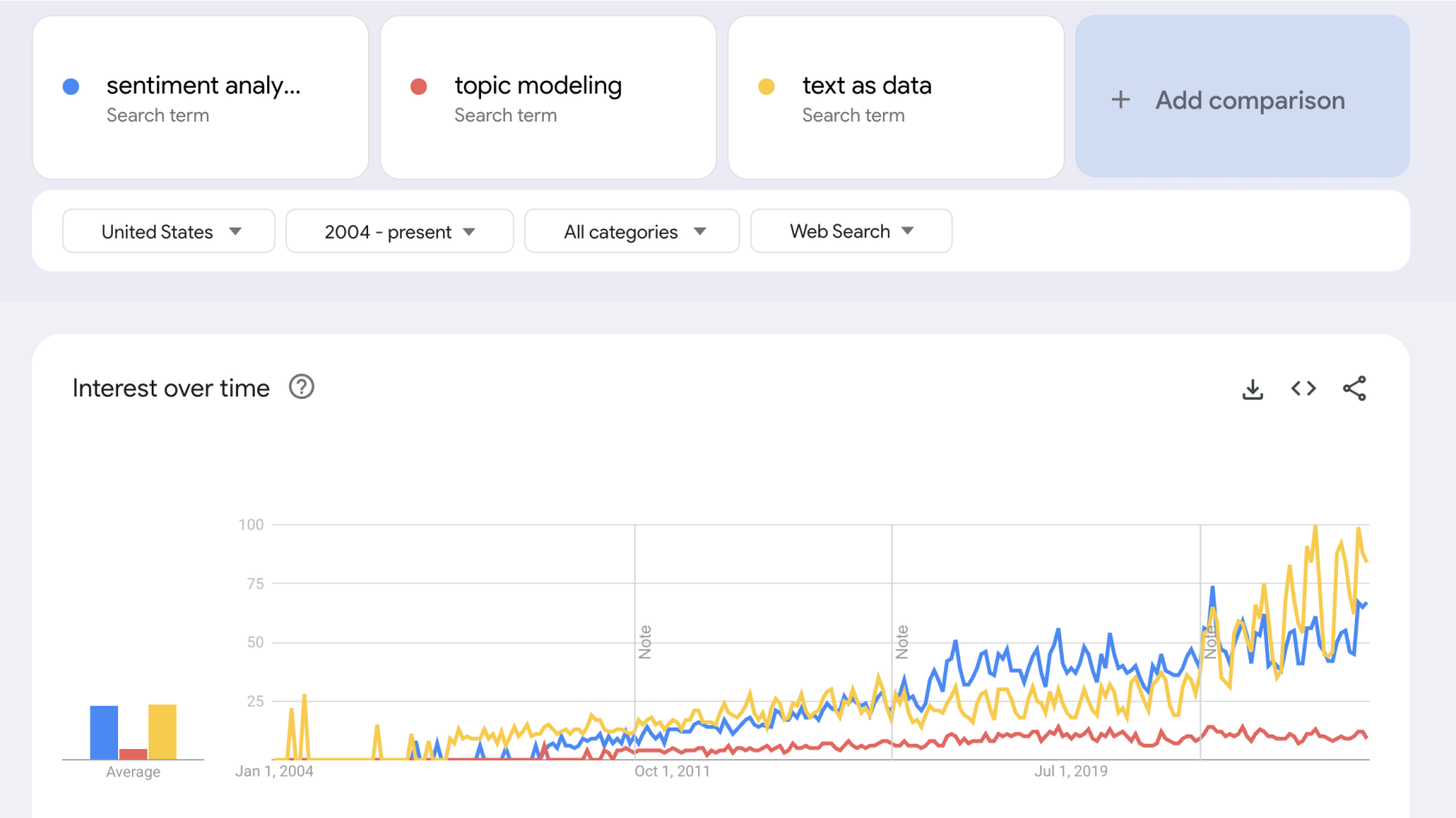



**Figure 1:** Trends, 2004-2025

# Political Risk Applications

- Political Science
  - Classify treaty provisions and language (Spirling 2012)
  - Measure political events in real-time (GeoQuant 2016-)
  - Identify foreign policy biases (Atalan, Reynolds and Jensen 2025)

# Political Risk Applications

- Political Science
  - Classify treaty provisions and language (Spirling 2012)
  - Measure political events in real-time (GeoQuant 2016-)
  - Identify foreign policy biases (Atalan, Reynolds and Jensen 2025)
- Finance
  - Analyze 10k statements for political risks item Measure effect of Trump speeches on markets
  - Predict Fed Reserve interest rate cuts

# Political Risk Applications

- Political Science
  - Classify treaty provisions and language (Spirling 2012)
  - Measure political events in real-time (GeoQuant 2016-)
  - Identify foreign policy biases (Atalan, Reynolds and Jensen 2025)
- Finance
  - Analyze 10k statements for political risks item Measure effect of Trump speeches on markets
  - Predict Fed Reserve interest rate cuts
- Government and Tech Industry
  - Identify extremist chatter
  - Label coordinated inauthentic behavior
  - Counter disinformation campaigns

# Why use Text as Data?

- We care about language
- Text as (qualitative) data is well-established norm
- Analyzing large texts is time-consuming, but computers can lower these costs

# Problems with Text Analysis

Speech is:

- Ironic

```
Break a leg.
```

# Problems with Text Analysis

Speech is:

- Ironic

  ```
  Break a leg.
  ```

- Coded

  ```
  Let's go Brandon.
  ```

# Problems with Text Analysis

Speech is:

- Ironic

  ```
  Break a leg.
  ```

- Coded

  ```
  Let's go Brandon.
  ```

- Informal

  ```
  Alright, bet.
  ```

# Problems with Text Analysis

Speech is:

- Ironic

  ```
  Break a leg.
  ```

- Coded

  ```
  Let's go Brandon.
  ```

- Informal

  ```
  Alright, bet.
  ```

- Order Dependent

  ```
  Peace, no more war. War, no more peace.
  ```

# Problems with Text Analysis

Speech is:

- Ironic

  ```
  Break a leg.
  ```

- Coded

  ```
  Let's go Brandon.
  ```

- Informal

  ```
  Alright, bet.
  ```

- Order Dependent

  ```
  Peace, no more war. War, no more peace.
  ```

- Multi-Language

# Problems with Text Analysis

Speech is:

- Ironic

  ```
  Break a leg.
  ```

- Coded

  ```
  Let's go Brandon.
  ```

- Informal

  ```
  Alright, bet.
  ```

- Order Dependent

  ```
  Peace, no more war. War, no more peace.
  ```

- Multi-Language

**Overall:** Validate, validate, validate.

# Text Analysis Procedures

Supervised:

Unsupervised:

Supervised + Unsupervised:

# Text Analysis Procedures

Supervised:

- Hand code set of documents
- Train model on handcoded documents
- Predict content of unlabeled documents

Unsupervised:

Supervised + Unsupervised:

# Text Analysis Procedures

Supervised:

Unsupervised:

- **Sentiment Analysis:** Measure content of documents
- **TF-IDF:** Identify distinctive words
- **Topic Modeling:** Cluster text into categories

Supervised + Unsupervised:

# Text Analysis Procedures

Supervised:

Unsupervised:

Supervised + Unsupervised:

- **Transformers:** BERT, GPT, LlLamA
- **Transfer Learning**: Zero Shot or Few-Shot learning
- **LLM Fine Tuning**: Re-learn for specific tasks

# Recall: Bag of Words Assumption

**Assumption:** We assume that word order doesn't matter in order to facilitate **tokenization**

# Recall: Bag of Words Assumption

**Assumption:** We assume that word order doesn't matter in order to facilitate **tokenization**

Tokenization:

# Recall: Bag of Words Assumption

**Assumption:** We assume that word order doesn't matter in order to facilitate **tokenization**

Tokenization:

- Treats words in a document as a **"bag of words"**
- Ignores long sequencing and context (otherwise NN...)
- Transforms words into word vector
- Different word lengths:
  - Unigram
  - Bigram
  - Trigram
  - ...

# Recall: Pre-Processing

**Motivation:** Prepare texts for computational analysis by removing ‘noise’ in the documents to create our bag of words

# Recall: Pre-Processing

**Motivation:** Prepare texts for computational analysis by removing 'noise' in the documents to create our bag of words

1. Acquire Text
2. Assemble into a **corpus**
3. Remove capitalization and punctuation
4. Discard word order
5. Combine similar terms
    - Stemming
    - Lemmatization
6. Create a count vector

**Outcome:** Create a **Document Term Matrix**

# Acquiring Texts

Popular Sources:

- Online databases, e.g. LexisNexis, Comparative Manifesto Project, Thomas/Congress.gov
- Websites
  - Scraping (Unstructured Data)
  - APIs
- Archives
  - Wayback machine
  - Pre-Digitized, e.g. FRUS or Wilson Center
  - OCR-Compatible, e.g. high quality scanner + optical character recognition
- Cloud Translation (Google)

# Dictionary Methods

# Motivation

- Do people like a product?
- How are people reacting to a policy?
- What's the national mood?

# What is a dictionary?

**Dictionaries:** List of words belonging to a category

Types of dictionaries:

- Encyclopedic
- Sentiment
- Topic-based, e.g. sports, food, locations
- User-defined content, e.g. GPR/geopolitics

# Sentiment Dictionaries

**Main Idea:** Sentiment dictionaries have a list of words which are scored according to some emotion or opinion content of the words

- Broad Sentiment (Positive vs Negative)
- Specific Emotion (Anger, Joy, Sadness)
- Subset of words (many are neutral)

Word Weights and Scores:

- Polarity: Positive (+1), Neutral (0), or Negative (-1)
- Numerical: $\{-2, -1, 1, 1\}$, $\{-1000, 1000\}$
- Compound: Average or normalized sum

# Sentiment Dictionaries

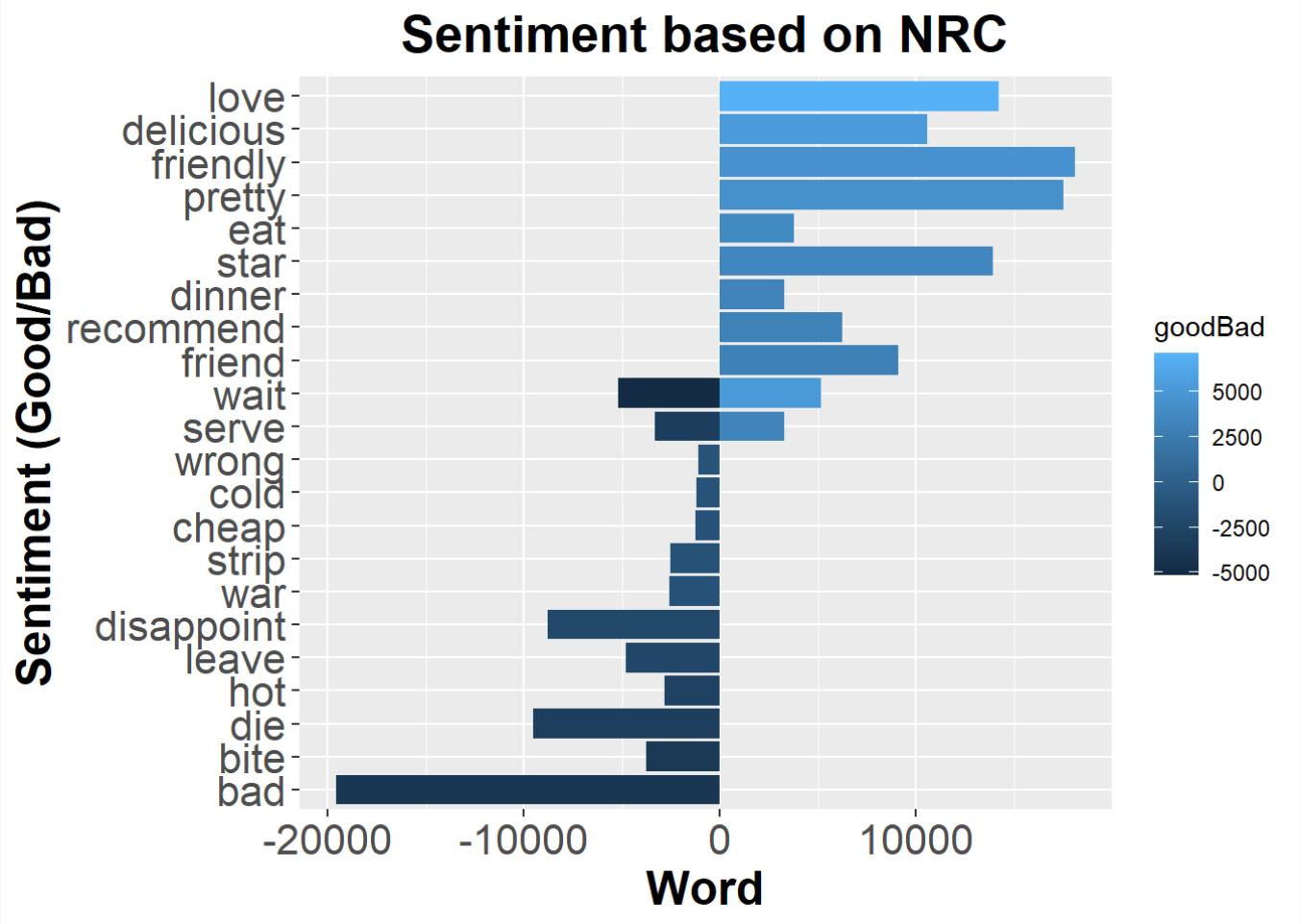



**Figure 2:** Source: NRC Emotional Lexicon

# Off the Shelf Dictionaries

1. NRC Emotional Lexicon
2. General Inquirer Database/Harvard IV-4
3. Linguistic Inquiry Word Count (LIWC)
4. Lasswell (e.g. power, uncertainty, desires)
5. Affective Norms for English Words (ANEW)
6. Valence Aware Dictionary and sEntiment Reasoner model (Vader)
7. ...

# Generating Custom Dictionaries

1. Manual Generation

2. Crowd-Sourcing

3. Quantitative methods

# Generating Custom Dictionaries

1. Manual Generation
    - Careful thought (epiphanies?) about useful words
    - Ex. LIWC: “We drew on common emotion rating scales…Roger’s Thesarus, standard English dictionaries, etc…[then] brain-storming sessions among 3-6 judges were held” to generate other words
2. Crowd-Sourcing

3. Quantitative methods

# Generating Custom Dictionaries

1. Manual Generation

2. Crowd-Sourcing
    - Surveys
    - Digital Labor (e.g. Appen)
3. Quantitative methods

# Generating Custom Dictionaries

1. Manual Generation

2. Crowd-Sourcing

3. Quantitative methods
    - Identify distinctive or separating words
    - Embeddings
    - LLMs

# Applying dictionaries to documents

- Identify relevant dictionary
- Map words in DTM to dictionary values
- Get aggregate tone based on weighted sentiment score

# Applying dictionaries to documents

- Vector of word counts: $\mathbf{X} = (X_{i1}, X_{i2}, \ldots, X_{iP}$, $i = (1, 2, \ldots, N)$
- Values attached to words $\theta = (\theta_1, \theta_2, \ldots, \theta_p)$
  - $\theta_p \in \{0, 1\}$
  - $\theta_p \in \{-1, 0, 1\}$
  - $\theta_p \in \{-1000, 1000\}$

# Applying dictionaries to documents

For each document $i$ calculate sentiment score $Y$

$$Y_i = \frac{\sum_{p=1}^{P} \theta_p X_{ip}}{\sum_{p=1}^{P} X_{ip}}$$

$Y_i$ is continuous:

- $Y_i > 0 \rightarrow$ positive category
- $Y_i < 0 \rightarrow$ negative category
- $Y_i \approx 0 \rightarrow$ ambiguous (neutral)

# Application: Quantifying Happiness

# Application: Quantifying Happiness

- Use presidential speeches to measure historical mood
- Use Tweets to find happiest place in the US
- Use IG Captions to Measure DST Mood Changes

# Application: Quantifying Happiness

- Use presidential speeches to measure historical mood
  - Reagan: “Morning Again in America”
  - Trump: “American Carnage”
- Use Tweets to find happiest place in the US

- Use IG Captions to Measure DST Mood Changes

# Application: Quantifying Happiness

- Use presidential speeches to measure historical mood

- Use Tweets to find happiest place in the US
  - Happiest States: Hawaii, Maine, Nevada, Utah
  - Saddest States: Louisiana, Mississippi, Maryland, Delaware
- Use IG Captions to Measure DST Mood Changes

# Application: Quantifying Happiness

- Use presidential speeches to measure historical mood
- Use Tweets to find happiest place in the US
- Use IG Captions to Measure DST Mood Changes
  - 25% increase in complaints of fatigue
  - 20% increase in positive sentiment

# Application: Quantifying Happiness

## Study: Dodds and Danforth (2009)

- Apply dictionary to measure happiness in songs, blogs, and State of the Unions

J Happiness Stud (2010) 11:441–456
DOI 10.1007/s10902-009-9150-9

RESEARCH PAPER

**Measuring the Happiness of Large-Scale Written Expression: Songs, Blogs, and Presidents**

**Peter Sheridan Dodds · Christopher M. Danforth**

Published online: 17 July 2009
© The Author(s) 2009. This article is published with open access at Springerlink.com

**Abstract** The importance of quantifying the nature and intensity of emotional states at the level of populations is evident: we would like to know how, when, and why individuals feel as they do if we wish, for example, to better construct public policy, build more successful organizations, and, from a scientific perspective, more fully understand economic and social phenomena. Here, by incorporating direct human assessment of words, we quantify happiness levels on a continuous scale for a diverse set of large-scale texts: song titles and lyrics, weblogs, and State of the Union addresses. Our method is transparent, improvable, capable of rapidly processing Web-scale texts, and moves beyond approaches based on coarse categorization. Among a number of observations, we find that the happiness of song lyrics trends downward from the 1960s to the mid 1990s while remaining stable within genres, and that the happiness of blogs has steadily increased from 2005 to 2009, exhibiting a striking rise and fall with blogger age and distance from the Earth's equator.

**Keywords** Happiness · Hedonometer · Measurement · Emotion · Written expression · Remote sensing · Blogs · Song lyrics · State of the Union addresses

# Application: Quantifying Happiness

**Study: Dodds and Danforth (2009)**

- Use Affective Norms for English Words (ANEW) dictionary
- Dictionary has 1034 words and measures affective reaction:
  - On a scale of 1-9 how happy does this word make you?
  - Happy: triumphant (8.82), paradise (8.72), love (8.72)
  - Neutral: street (5.22), paper (5.20), engine (5.20)
  - Unhappy: cancer (1.5), funeral (1.39), rape (1.25)

# Application: Quantifying Happiness

**Study: Dodds and Danforth (2009)**

- Valence Score: Happiness for text $i$ score based on word p having happiness value score $\theta_p$ and document frequency $X_{ip}$

$$\text{Happiness}_i = \frac{\sum_{p=1}^{P} \theta_p X_{ip}}{\sum_{p=1}^{P} X_{ip}}$$

# Application: Quantifying Happiness

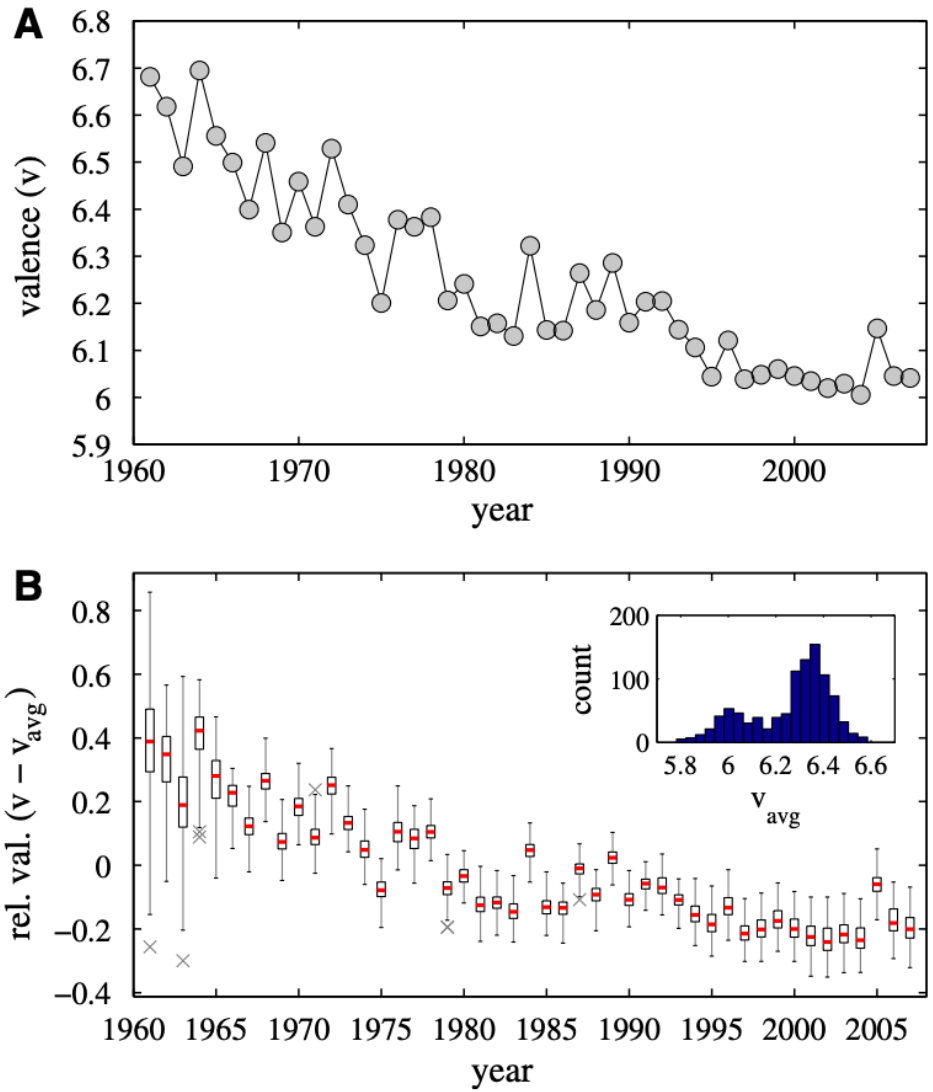



**Fig. 4** **a** Valence time series for song lyrics, showing a clear downward trend over the 47 year period starting in 1961. Valence is measured by averaging over the valences of individual words from the ANEW study (Bradley and Lang 1999) found in songs released in each year. **b** Box and whisker plot of relative

# Application: Quantifying Happiness

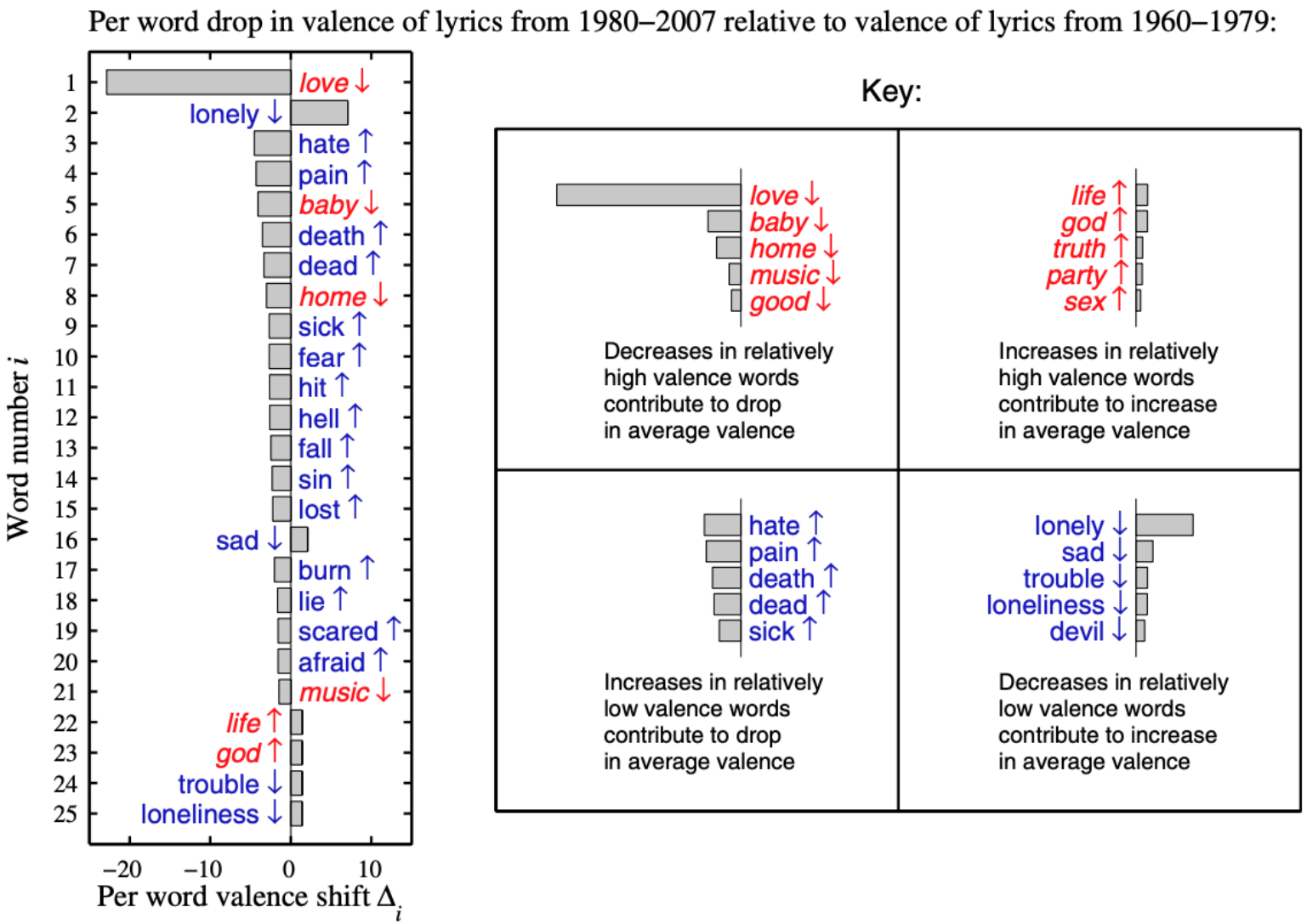



**Fig. 5** Valence Shift Word Graph: Words ranked by their absolute contributions to the drop in average valence of song lyrics from January 1, 1980 onwards relative to song lyrics from before January 1, 1980. The contribution of word *i* is defined in Eq. 3 and explained in the surrounding text

# Application: Quantifying Happiness

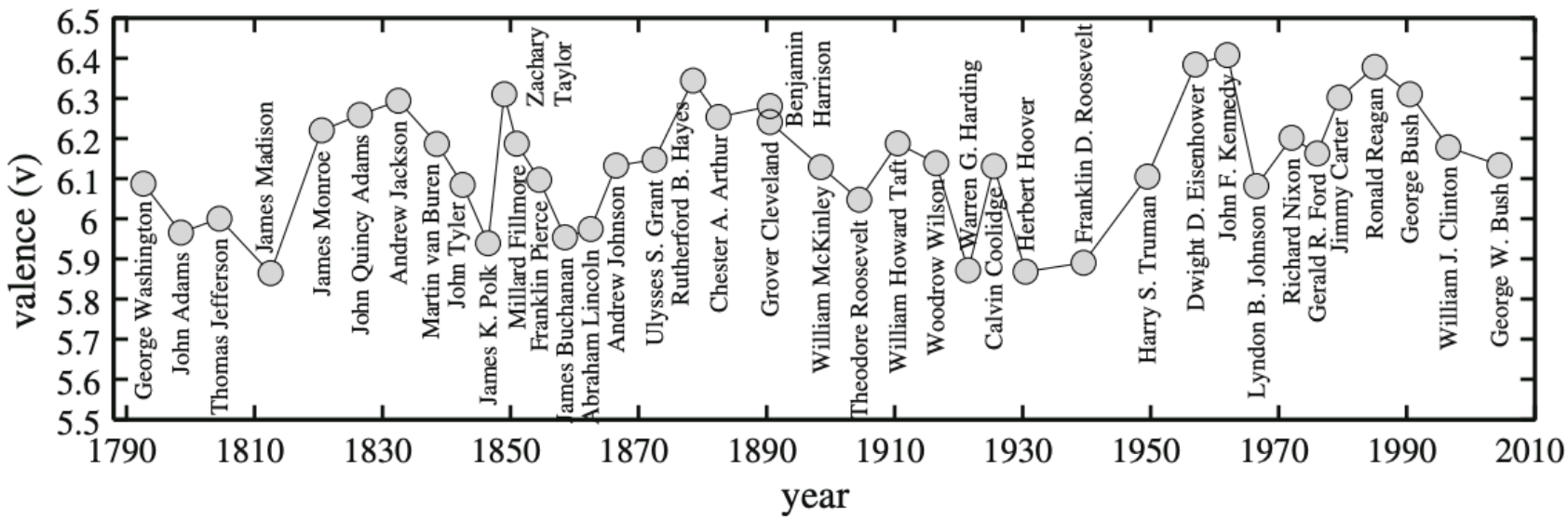



**Fig. 11** Average valence of State of the Union addresses, binned by President and plotted against the average of the years the President was in office

# Limits to Dictionary Methods

- Most dictionary methods do not take into account qualifiers, intensifiers, or negation, e.g. good = no good (Vader claims to be an exception)
- Ignore sarcasm, irony, nuance
- Some words have multiple meanings (polysemes)
- Ignores context dependence

# Limits to Dictionary Methods

- Most dictionary methods do not take into account qualifiers, intensifiers, or negation, e.g. good = no good (Vader claims to be an exception)
- Ignore sarcasm, irony, nuance
- Some words have multiple meanings (polysemes)
- Ignores context dependence

**Word of Warning: Always validate your findings!**

# Warning: Validate Your Findings

**Motivation:** Accounting research measures the tone of 10-K reports on financial performance to measure industry sentiment.
Analysis uses 2 dictionaries:

- Harvard-IV-4 Dictionary
- SME-defined FinNeg (Financial Negative) dictionary

# Warning: Validate Your Findings

**Problem:** Different dictionaries result in different findings.

- Negative Words in Harvard-IV-4 Dictionary, Not Negative in FinNeg: tax, cost, capital, board, liability, foreign, cancer, crude (oil), tire
- Not Negative in Harvard, Negative in FinNeg: felony, litigation, restated, misstatement, unanticipated
- 73% of Harvard negative words not in FinNeg!

# Distinctive Words

# Sometimes we want to identify distinctive words

**Motivation:**

- Non-sentiment related tasks (e.g. themes, goals, uncertainty)
- Feature (variable) selection: identify relevant words for subsequent analysis
- Examine specific types of words, e.g.
  - Partisan Language
    e.g., compare Republican vs Democratic speeches
  - Ideological Language
    e.g., compare liberal vs conservative news stories
  - Gendered Language
    e.g., compare toy advertising

# Finding ‘Discriminating’ Words

**Main Idea:** Discriminating words are the most distinctive words in each corpus, i.e. appear in 1 document, but not another.

**Examples:**

- Bible: ‘Jesus’ is distinct to New Testament
- Shakespeare: ‘Oberon’ distinct to Midsummer’s Night Dream
- Chat-GPT text: ‘delve’ (oddly) distinct to GPT outputs

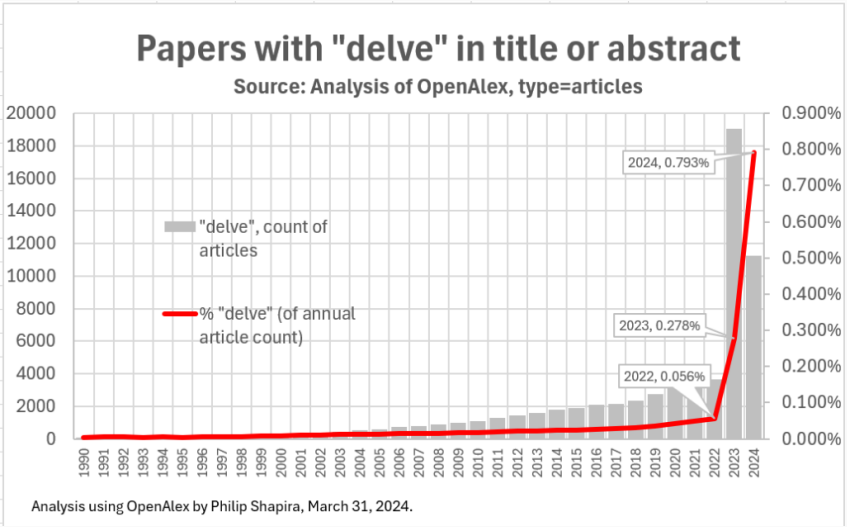



**Figure 3:** Source: Philip Shapira

# Statistical Methods to Find Distinctive Words

1. Unique Usage
2. Difference in Frequencies
3. Difference in Averages
4. Term Frequency-Inverse Document Frequency (tf-idf)

# Motivating Example: Is this PubMed abstract ChatGPT-generated?

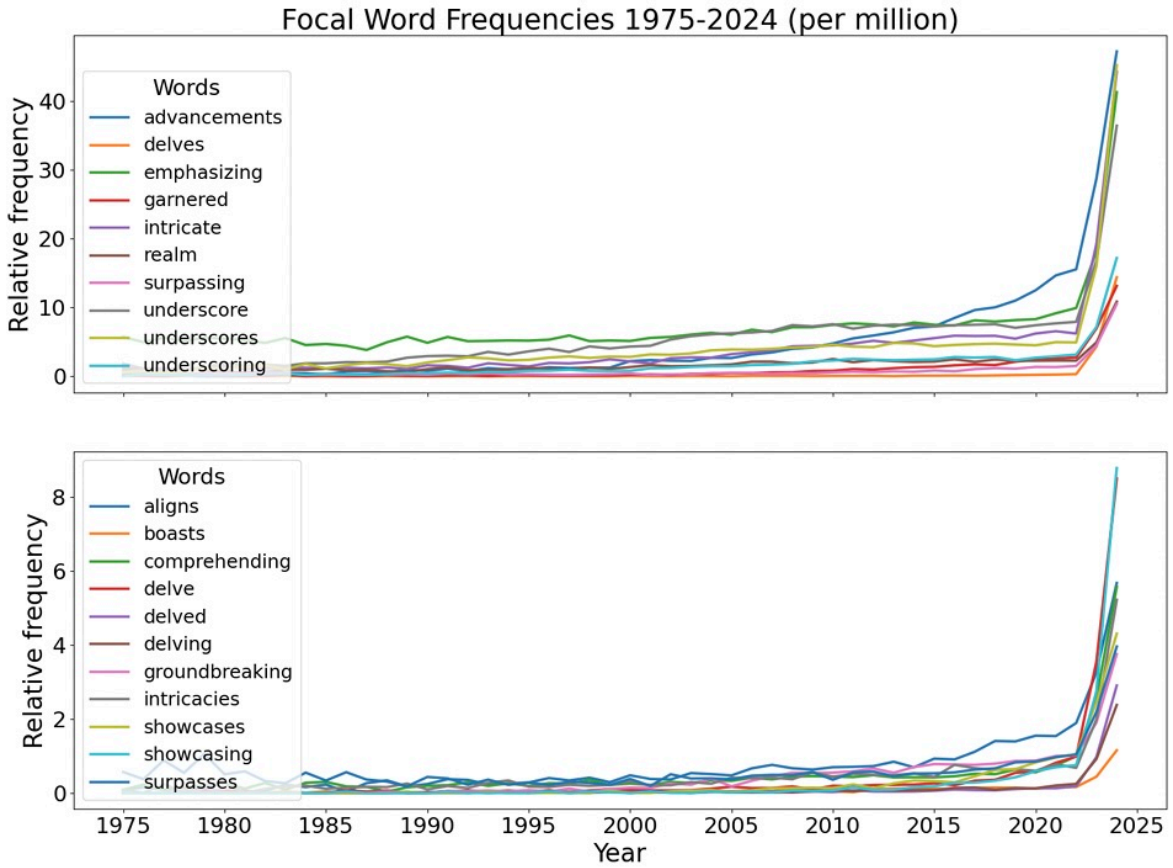



Figure 4: Occurrences per million words in PubMed abstracts for our 21 focal words.

**Figure 4:** Source: FSU/ Juzek and Ward (2025)

# Option 1: Unique Usage (Focal Words)

- **Main Idea:** Identify distinctive words based on their exclusivity to a text.

# Option 1: Unique Usage (Focal Words)

- **Main Idea:** Identify distinctive words based on their exclusivity to a text.
- **Example:** If we suspect GPT-written abstract, we should count certain focal words (e.g. “groundbreaking”) as distinctive bread crumbs

# Option 1: Unique Usage (Focal Words)

- **Main Idea:** Identify distinctive words based on their exclusivity to a text.
- **Example:** If we suspect GPT-written abstract, we should count certain focal words (e.g. “groundbreaking”) as distinctive bread crumbs
- **Limit:** These words tend to not be terribly interesting or informative

# Option 2: Difference in Frequencies

- **Main Idea:** Identify distinctive words based on their difference in frequency

# Option 2: Difference in Frequencies

- **Main Idea:** Identify distinctive words based on their difference in frequency
- **Method:**
  - Count the number of time each document uses a word
  - Find the words with the largest absolute difference

# Option 2: Difference in Frequencies

- **Main Idea:** Identify distinctive words based on their difference in frequency
- **Method:**
  - Count the number of time each document uses a word
  - Find the words with the largest absolute difference
- **Example:** If post-2022 abstract mentions groundbreaking 100x and pre-2022 abstract mentions 'groundbreaking' 1x, then consider post-2022 abstracts relatively distinct to pre-2022

# Option 2: Difference in Frequencies

- **Main Idea:** Identify distinctive words based on their difference in frequency
- **Method:**
  - Count the number of time each document uses a word
  - Find the words with the largest absolute difference
- **Example:** If post-2022 abstract mentions groundbreaking 100x and pre-2022 abstract mentions 'groundbreaking' 1x, then consider post-2022 abstracts relatively distinct to pre-2022
- **Limit:** Doesn't account for difference in total words–are post-2022 documents just longer (vibe writing)?

# Option 3: Difference in Averages

**Main Idea:** Identify distinctive words based on difference in rates

# Option 3: Difference in Averages

**Main Idea:** Identify distinctive words based on difference in rates

**Method:**

- Normalize DTM from count to proportions. For each word $p$ in corpus $c$:

$$\mu_p = \frac{\sum_{i=1}^{N} p_i}{T}$$

  - $p_i$ is the number of times a word $p$ appears in document $i$,
  - N is the total number of documents in $c$
  - $T$ is the total number of words in c
- Take the difference between one author's proportion of a word and another's proportion of the same word:

$$\theta_p = \mu_{p,\text{Post-2022 Document}} - \mu_{p,\text{Pre-2022 Document}}$$

- Find words with highest absolute difference

# Option 3: Difference in Averages

**Limits:**

1. Favors more frequent words, e.g.
   Word 1: 30/1000 (Pre-2022); 25/1000 (Post-2022) → score 5/1000
   Word 2: 5/1000 (Pre-2022); 1/1000 (Post-2022) → score 4/1000

# Option 3: Difference in Averages

**Limits:**

1. Favors more frequent words, e.g.
   Word 1: 30/1000 (Pre-2022); 25/1000 (Post-2022) → score 5/1000
   Word 2: 5/1000 (Pre-2022); 1/1000 (Post-2022) → score 4/1000
2. Ignores cases where one text uses a word frequently and another text barely uses it, e.g. ‘omicron’
   Word 1: 1/1000 (Pre-2022) and 990/1000 (Post-2022)

# Option 3: Difference in Averages

**Limits:**

1. Favors more frequent words, e.g.
   Word 1: 30/1000 (Pre-2022); 25/1000 (Post-2022) → score 5/1000
   Word 2: 5/1000 (Pre-2022); 1/1000 (Post-2022) → score 4/1000
2. Ignores cases where one text uses a word frequently and another text barely uses it, e.g. ‘omicron’
   Word 1: 1/1000 (Pre-2022) and 990/1000 (Post-2022)
3. Biased towards differences in rates of frequent words > differences in rates of rare words

# Option 3: Difference in Averages

**Limits:**

1. Favors more frequent words, e.g.
   Word 1: 30/1000 (Pre-2022); 25/1000 (Post-2022) → score 5/1000
   Word 2: 5/1000 (Pre-2022); 1/1000 (Post-2022) → score 4/1000
2. Ignores cases where one text uses a word frequently and another text barely uses it, e.g. 'omicron'
   Word 1: 1/1000 (Pre-2022) and 990/1000 (Post-2022)
3. Biased towards differences in rates of frequent words > differences in rates of rare words

**Solution:** Divide the difference in texts' average rates by the pooled average rates

# Option 4: TF-IDF

**Main Idea:** Identify distinctive words based on their relative frequency and unique usage.

# Option 4: TF-IDF

**Main Idea:** Identify distinctive words based on their relative frequency and unique usage.

- Term Frequency (TF): number of times word $p$ appears in document $i$
- Inverse Document Frequency (IDF): measure of exclusive usage across documents $i$

# Option 4: TF-IDF

**Main Idea:** Identify distinctive words based on their relative frequency and unique usage.

- Term Frequency (TF): number of times word $p$ appears in document $i$
- Inverse Document Frequency (IDF): measure of exclusive usage across documents $i$

**TF-IDF Equation:**

$$w_{ip} = tf_{ip} \times log\frac{N}{df_p}$$

- tf: number of occurrences of word $p$ in document i
- $df_p$: number of documents containing word p
- N: total number of documents

# Option 4: TF-IDF

**Example:**

- 'Medicine', 'study', and 'research' appear in all PubMed abstracts so tf is high and IDF score is approximately zero ($log\frac{n}{n} = log(1) = 0$) → low TF-IDF scores
- 'advancements', 'delves', and 'groundbreaking' are relatively unique to post-2022 abstracts so will have **higher IDF scores**
- 'Covid' is mentioned more in post-2022 abstracts so it will have a higher TF-IDF score than 'ebola' or 'measles'
- Ranking TF-IDF scores:
  delves > groundbreaking > covid > ...> ebola > research

# Option 4: TF-IDF

**Main Takeaways:**

- Common words have lower TF-IDF scores
- Higher TF-IDF scores reflect more exclusive usage adjusting for frequency
- GPT text has a very distinctive voice.

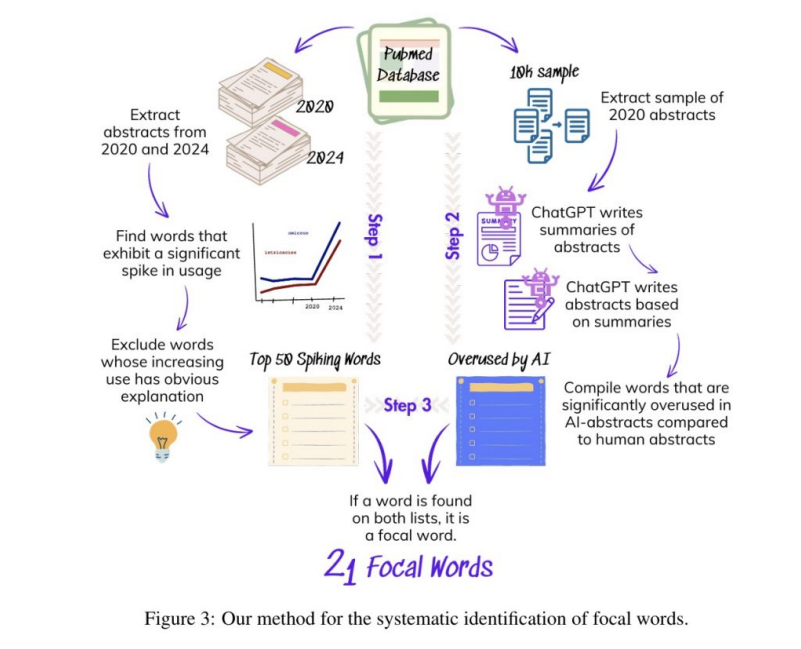



Figure 3: Our method for the systematic identification of focal words.

**Figure 5:** Juzek and Ward (2025)

# Embeddings

# Motivation

We just learned what makes documents distinct ...
Now, what make documents similar?

- Simple Answer: Similar word count vectors
- Complicated Answer: Similar use of language

# Motivating Example: Turn It In

Germany was the first to employ area bombing tactics during its assault on Poland in September 1939. In 1940, during the Battle of Britain, the Luftwaffe failed to bring Britain to its knees by targeting London and other heavily populated areas with area bombing attacks. Stung but unbowed, the Royal Air Force (RAF) avenged the bombings of London and Coventry in 1942 when it launched the first of many saturation bombing attacks against Germany. In 1944, Hitler named the world's first long-range offensive missile V-1, after "vergeltung," the German word for "vengeance" and an expression of his desire to repay Britain for its devastating bombardment of Germany.

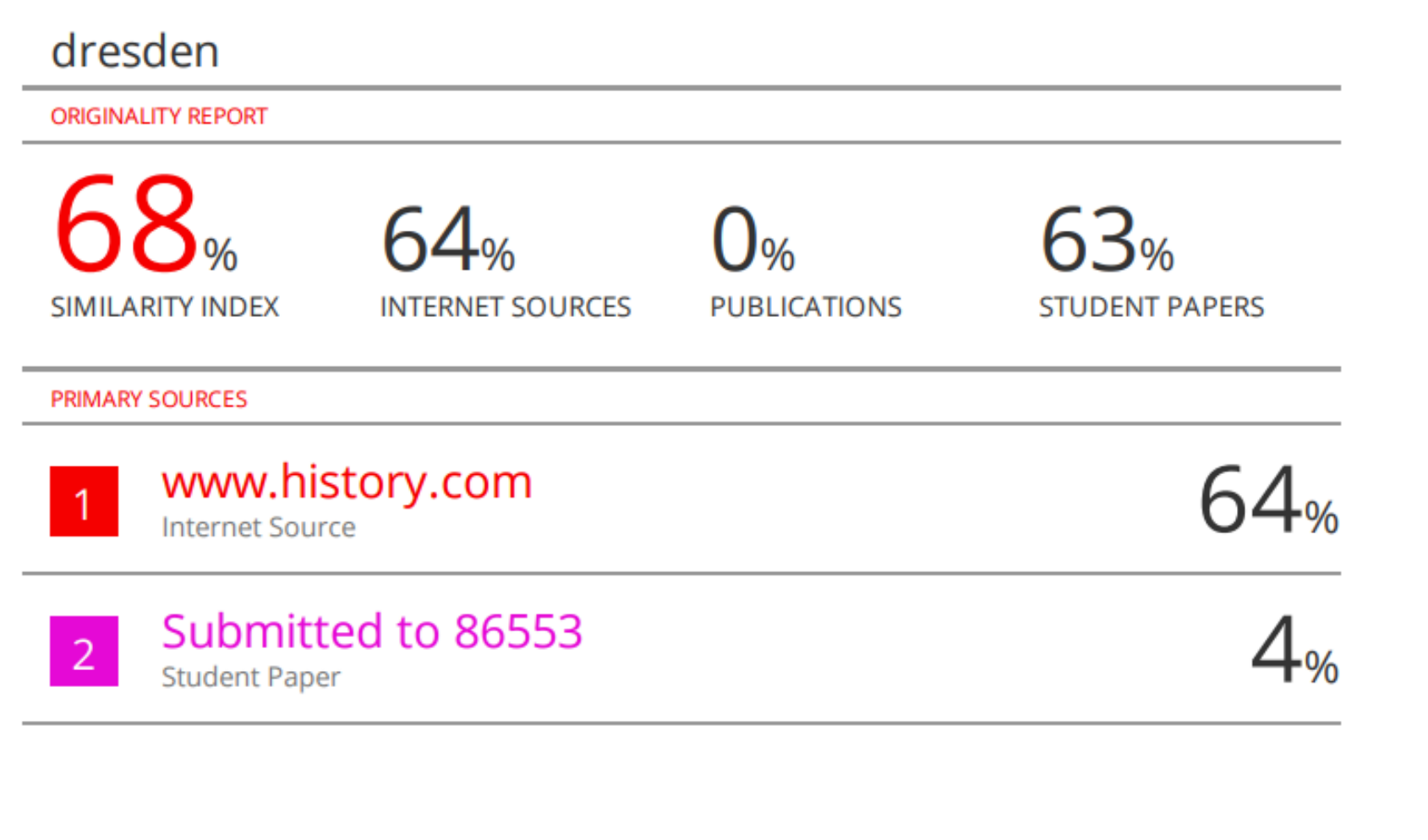
The Allies never overtly admitted that they were engaged in saturation bombing; specific military targets were announced in relation to every attack. However, it was but a veneer, and few mourned the destruction of German cities that built the weapons and bred the soldiers that by 1945 had killed more than 10 million Allied soldiers and even more civilians. The firebombing of Dresden would prove the exception to this rule.

**Figure 6:** History.com Essay on Dresden Bombing

I think the bombing of Dresden is controversial, because it was not important to German wartime or industrial center. All enemy industries, not just munitions, were targeted. It is different because the bombs are not just aimed at military bases nor military related places. Instead it bombs an entire place including civilians and troop areas. For an example all enemy industry, not just war mutants, were attacked, and civilians proportions of cities are obliterated along troop areas. Causing wreaking on the German economy it then betraying the German people's moreal, and forcing an early surrender. The purpose was to terrorize the German population by forcing and early surrender, but they were disrupting important lines of communication that would have hired the Soviet offensive. In 1944, Hitler named the world's first long range offensive missile V-1 after vergeltung. The German word for "vengeance" and an expression of his desire to repay Britain for devastating bombardment of Germany. Dresden's contribution to the war effort was minimal compared with other german cities. In February 1945, refugees fleeing the Russian advance in the east took refuge there. As Hitler had thrown much of his surviving forces into defense of Berlin in the north, city defences were minimal, and the Russians would have had little trouble capturing Dresden. It seemed an unlikely target for a major Allied air attack. There is no disputing that the British incendiary attack on the night of February 13 to February 14 was conducted, if not primarily, for the purpose of terrorizing the German population and forcing early surrender.

**Figure 7:** Student Essay on Dresden

# Motivating Example: Turn It In



dresden

ORIGINALITY REPORT

| 68% | 64% | 0% | 63% |
|---|---|---|---|
| SIMILARITY INDEX | INTERNET SOURCES | PUBLICATIONS | STUDENT PAPERS |

PRIMARY SOURCES

| | | |
|---|---|---|
| 1 | www.history.com<br>Internet Source | 64% |
| 2 | Submitted to 86553<br>Student Paper | 4% |

**Figure 8:** Turnitin Similarity Report

# Texts and Geometry

Consider a document term matrix:

$$X = \begin{pmatrix} & Word1 & Word2 & Word3 & \dots & WordP \\ Doc1 & 1 & 0 & 0 & \dots & 0 \\ Doc2 & 0 & 2 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ DocN & 0 & 0 & 1 & \dots & 3 \end{pmatrix}$$

# Texts and Geometry

Consider a document term matrix:

$$X = \begin{pmatrix} & Word1 & Word2 & Word3 & \dots & WordP \\ Doc1 & 1 & 0 & 0 & \dots & 0 \\ Doc2 & 0 & 2 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ DocN & 0 & 0 & 1 & \dots & 3 \end{pmatrix}$$

By transforming text into a word count matrix, we represent each document as point in multi-dimensional space:

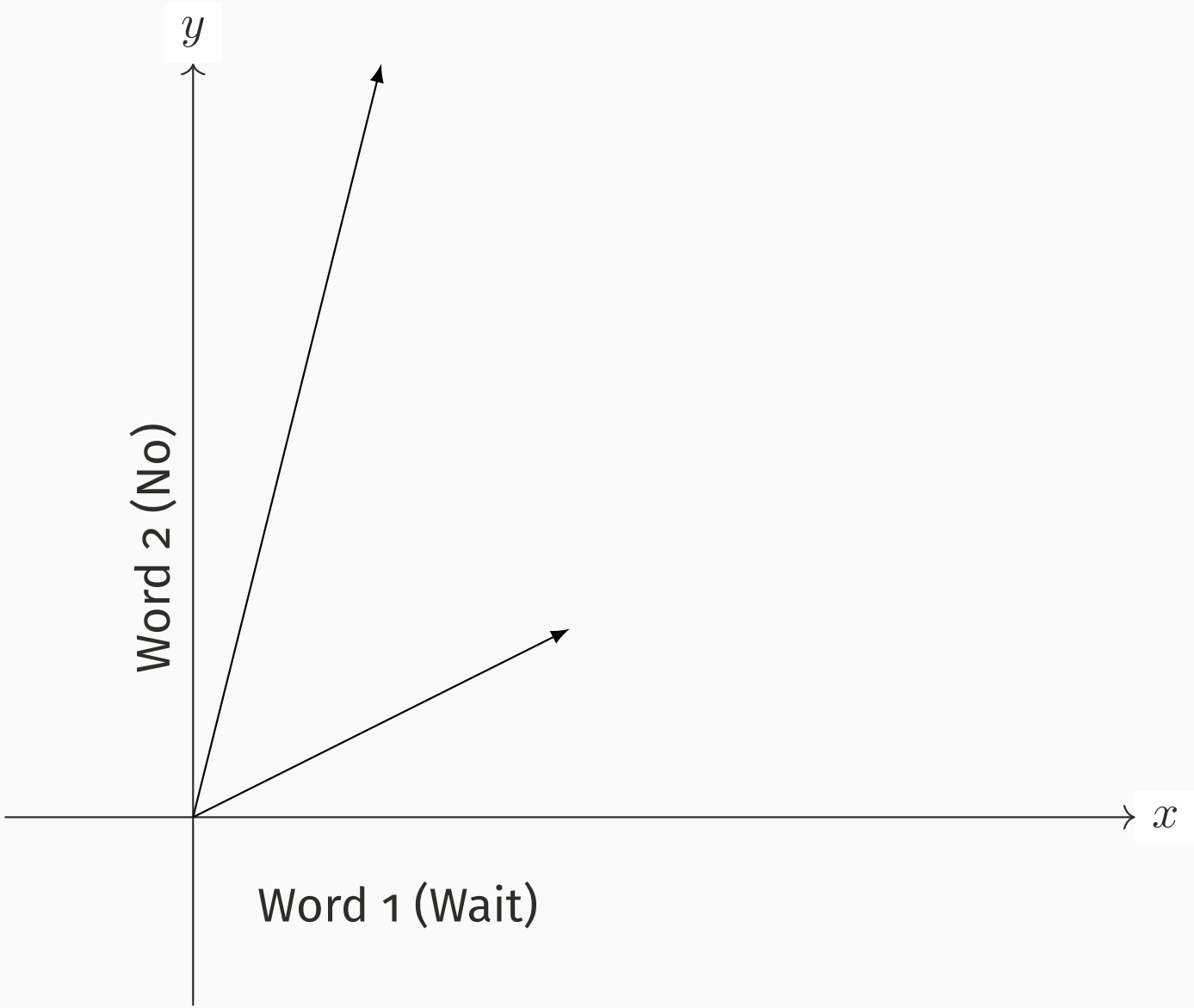
- Provides a geometry
- natural notions of distance and similarity
- apply linear algebra to calculate distance mathematically

# Example: Comparing documents

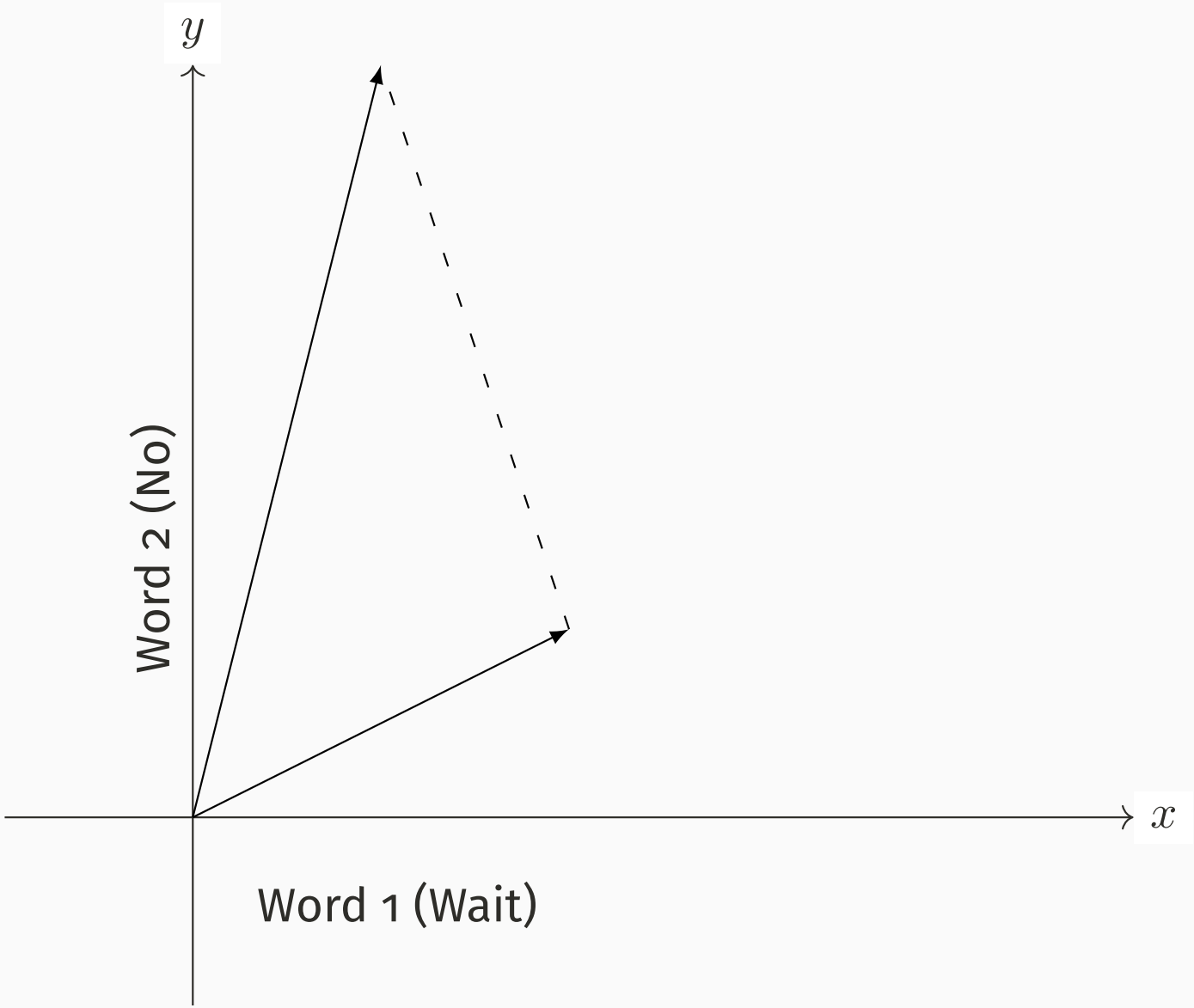
Document 1: “Wait? No wait.” → (2 Waits, 1 No) → (2, 1)

Document 2: “No, wait! No, no, no! → (1 Wait, 4 No) → (1,4)

# Example: Comparing documents



# Example: Comparing documents



# Recap: Euclidean Distance

We can measure the similarity of language based on Euclidean distance:

$$d\left(x_{i'j}, x_{ij}\right) = \sqrt{\sum_{i=1}^{n}\left(x_{ij} - x_{i'j}\right)^2}$$

- When distance is small, observation pairs (language vectors) are more similar
- When distance is larger, observation pairs (language vectors) are more dissimilar

# Recap: Euclidean Distance

Document 1: $\mathbf{X}_1 = (2, 1)$

Document 2: $\mathbf{X}_2 = (1, 4)$

$$d(\text{doc1, doc2})) = \sqrt{(1-2)^2 + (4-1)^2} = \sqrt{10} = 3.16$$

# Limits to Euclidean Distance

Euclidean distance depends on document length.

**Example:**
Document 1: “Wait? No wait.” → (2, 1)

Document 2: “No, wait! No, no, no! → (1,4)

Document 3: “Wait? No wait. Wait, wait, no! …” → (4, 2)

$$d(\text{doc2}, \text{doc3})) = \sqrt{(4-1)^2 + (2-4)^2} = \sqrt{13} = 3.60$$

# Limits to Euclidean Distance

Euclidean distance depends on document length.

**Example:**
Document 1: “Wait? No wait.” → (2, 1)

Document 2: “No, wait! No, no, no! → (1,4)

Document 3: “Wait? No wait. Wait, wait, no! …” → (4, 2)

$$d(\text{doc2}, \text{doc3})) = \sqrt{(4-1)^2 + (2-4)^2} = \sqrt{13} = 3.60$$

This makes document 3 seem more different than document 1. But it’s the same content!

# Limits to Euclidean Distance

Euclidean distance depends on document length.

**Example:**
Document 1: “Wait? No wait.” → (2, 1)

Document 2: “No, wait! No, no, no! → (1,4)

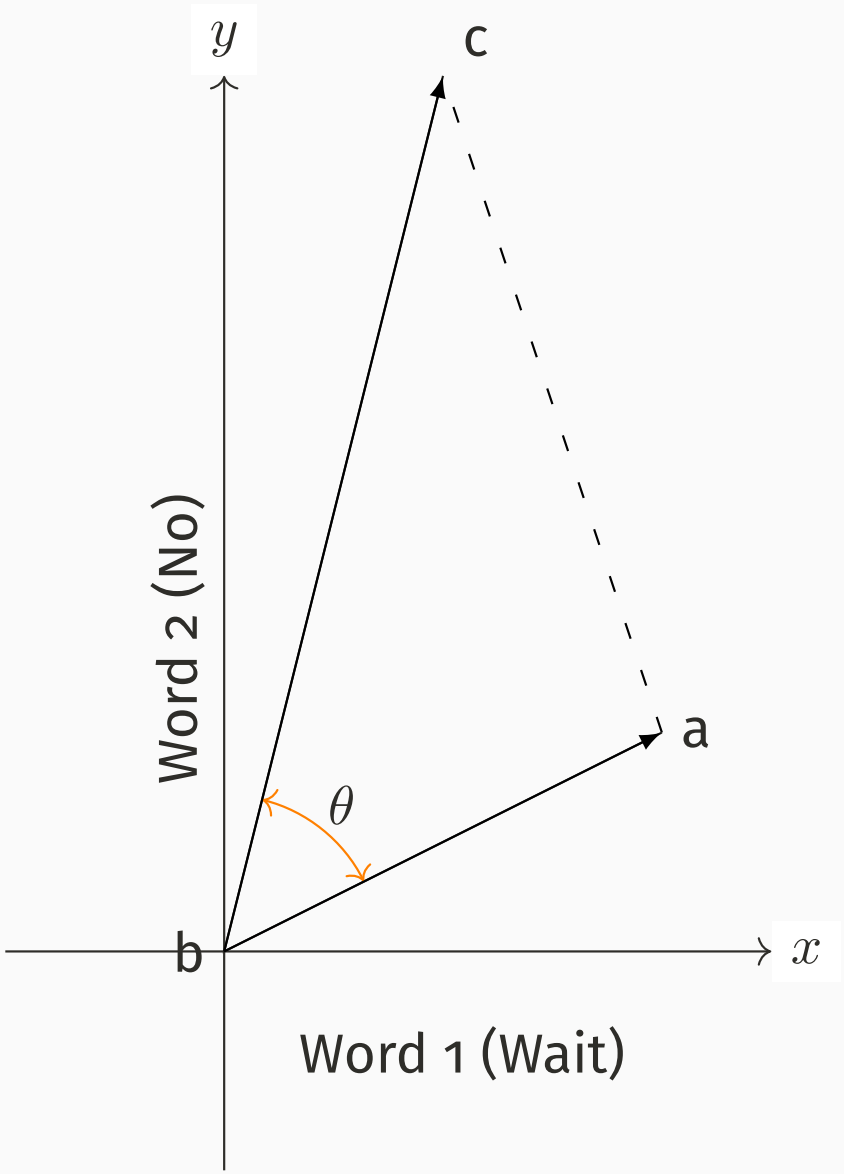
Document 3: “Wait? No wait. Wait, wait, no! …” → (4, 2)

$$d(\text{doc2}, \text{doc3})) = \sqrt{(4-1)^2 + (2-4)^2} = \sqrt{13} = 3.60$$

This makes document 3 seem more different than document 1. But it’s the same content!

**Solution:** **Cosine similarity**

# Cosine Similarity




- Cosine Similarity
- Adjusts for document length
- Measures **cosine of the angle** $\theta$ between vectors
- Measure of similarity rather than distance between 0, 1
- Convert to distance (or dissimilarity): $1 - cos(\theta)$

# Documents vs Words

Document Embeddings:

- Compare similarities/dissimilarities across documents
- Represent entire documents as a single vector
- **Tool:** doc2vec

Word Embeddings:

- Compare similarities/dissimilarities across words
- Represent single words as a single vector
- **Tool:** word2vec

# What are word embeddings?

**Word embeddings:** vectors of numbers that represent the meaning of a word.

| Word | Doc1 | Doc2 | Doc3 | Doc4 | ... | Doc$_n$ |
|---|---|---|---|---|---|---|
| cat | 0.015 | 0.018 | 0.083 | 0.009 | ... | -0.012 |
| sleep | 0.060 | 0.046 | 0.063 | 0.002 | ... | -0.093 |
| fluffy | -0.042 | -0.031 | -0.054 | 0.033 | ... | 0.056 |

# What are word embeddings?

Assumption: words only have meaning in relation to other words.

Implication: Vectors lend themselves to **mathematical operations**, e.g. adding and subtracting.

**Characteristics**:

- each word is represented as a vector in a high-dimensional space (typically 100-300 dimensions)
- each continuous value (or “weight”) captures a dimension of the word’s meaning
- semantically similar words are close together in this space
- dissimilar words far away
- Ex: “cat” is close to “fluffy”, while distant from “banana”.

# Why represent words as vectors?

1. Find similar words - helpful for stemming, lemmatizing, etc

Most similar vectors to the `talk` vector (using cosine similarity):

- `talking`, 0.6718707084655762
- `speak`, 0.6052849292755127
- `talked`, 0.5983412265777588
- `tell`, 0.5425027012825012
- `conversation`, 0.5362381935119629
- `hear`, 0.5258073806762695
- `know`, 0.5096373558044434
- `think`, 0.4957665205001831
- `listen`, 0.4826739430427551
- `discuss`, 0.4741832017898596

# Why represent words as vectors?

2. Solve problems of **disambiguation**

Example: ‘bank’

- financial institution?
- side of a river?
- hedge against?

To isolate the financial meaning, remove the river meaning dimensions from the bank vector, keeping the financial bank meaning dimensions:

> bank - `river` ≈ the financial institution

Or, add vectors to approximate and capture their conjoint meaning.

> `united` + `states` ≈ the country `United States`

# Why represent words as vectors?

3. Analogical Reasoning - Identify similarities using a simple vector offset method based on cosine distance.

Example: (with apologies to Judith Butler)

$$\texttt{king} - \texttt{man} + \texttt{woman} \approx \texttt{queen}$$

Subtract one meaning from the word vector for `king` (maleness), add another meaning (femaleness), and show that this new word vector (`king - man + woman`) maps most closely to the word vector for `queen`.

# Analogical reasoning is just high-dimensional mapping

| Query (a is to b as c is to d?) | Answer (d) |
|---|---|
| king: queen → man: ? | woman |
| smart: smarter → strong: ? | stronger |
| Tokyo: Japan → Paris:? | France |
| Google: Larry Page → Tesla: ? | Elon Musk |

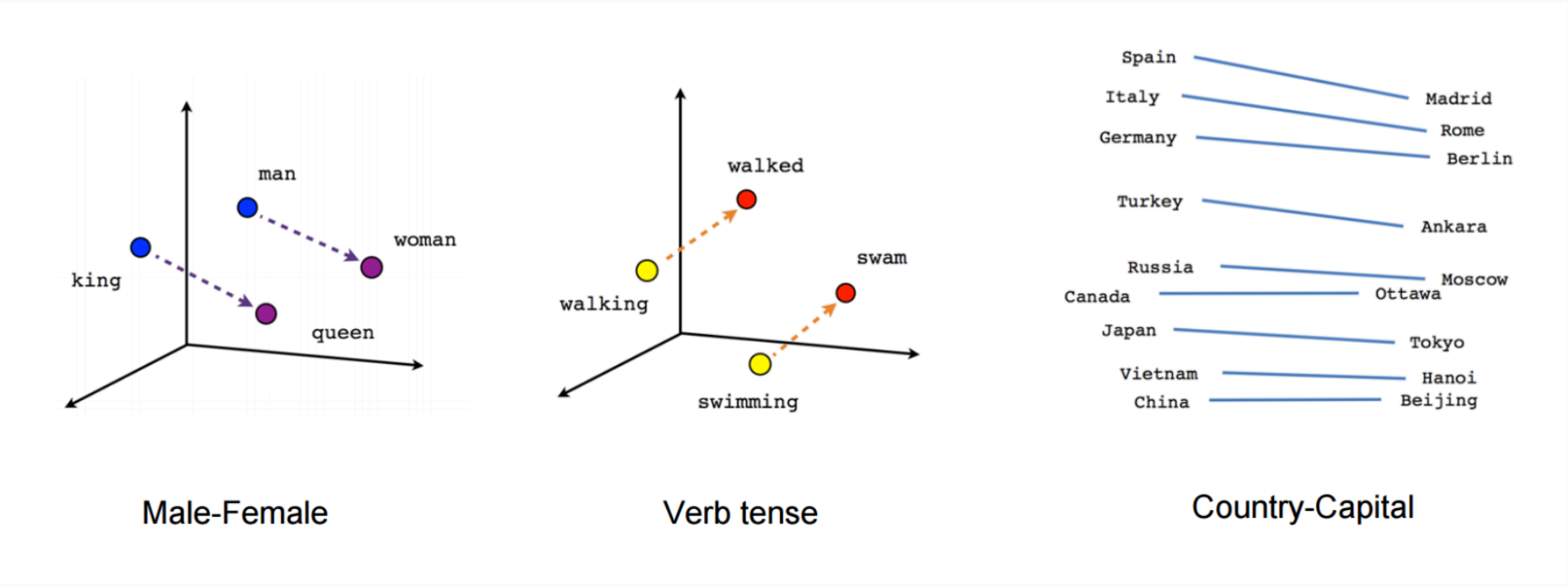



**Figure 9:** Dimensional Mapping

# Where do word vectors come from?

Firth's (1957) distributional hypothesis:

> *"You shall know a word by the company it keeps"*

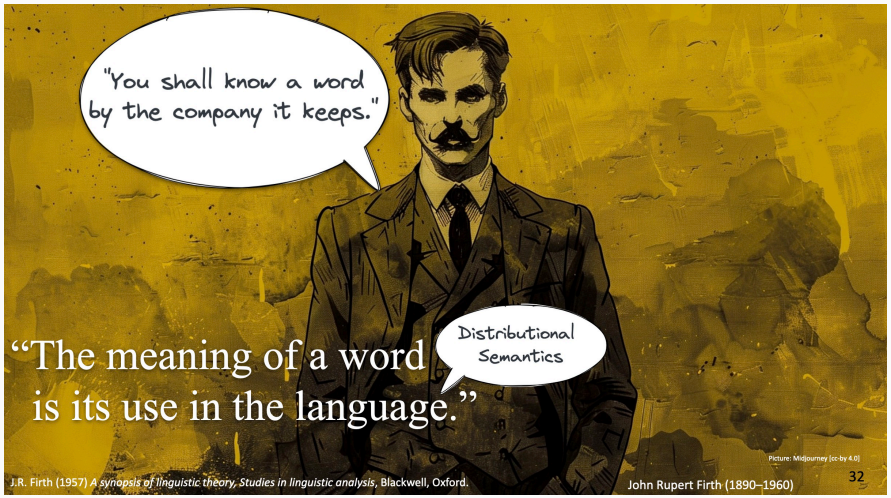



**Figure 10:** Words that share similar **contexts** tend to have similar meanings.

# Context Windows

**Contexts** refers to surrounding words.

**Context Window**: N words before and after target/center word.

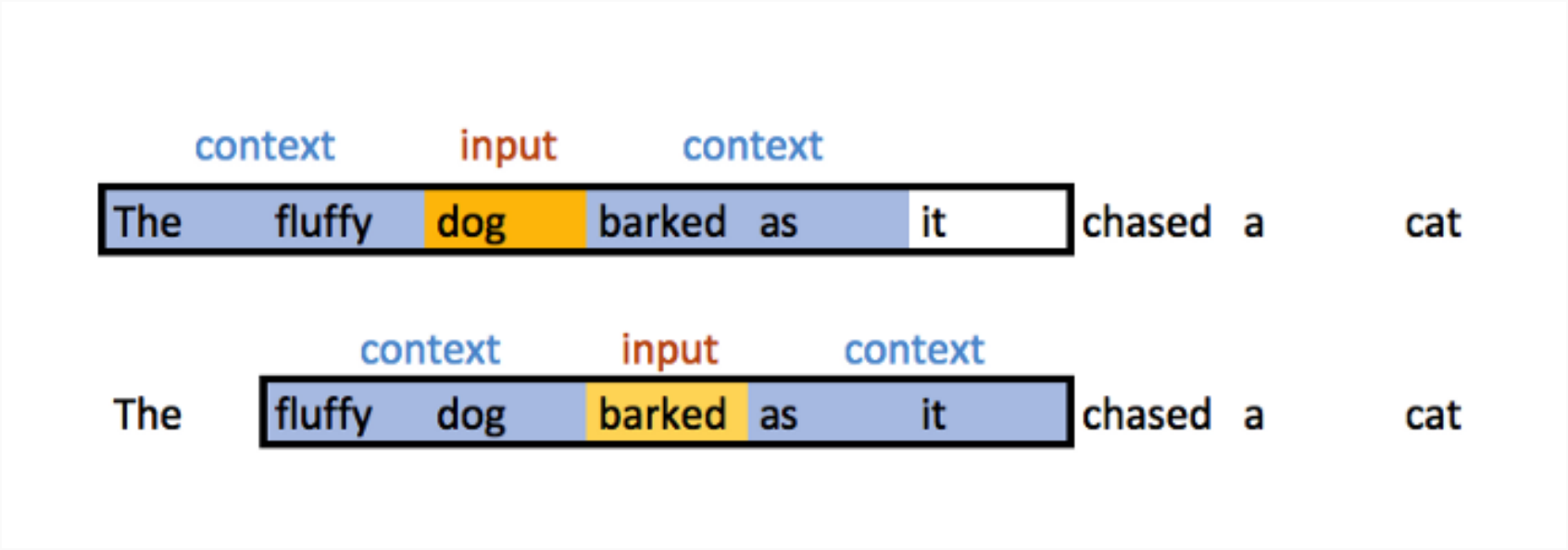



**Figure 11:** Caption

# Word2Vec

**Supervised task**: for any given word, predict whether another word is likely to appear nearby (i.e., in the context window).

The model is **trained** on word, context pairings for every word in the corpus.

Example: small window size of 2.

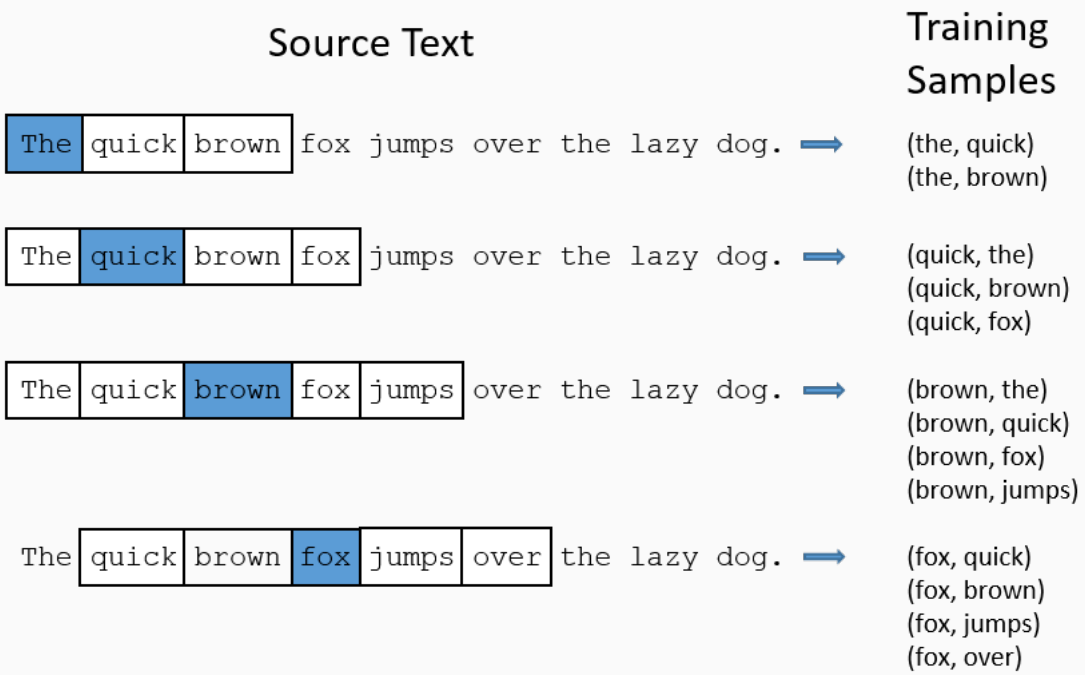



**Figure 12:** Source: Rochelle Terman

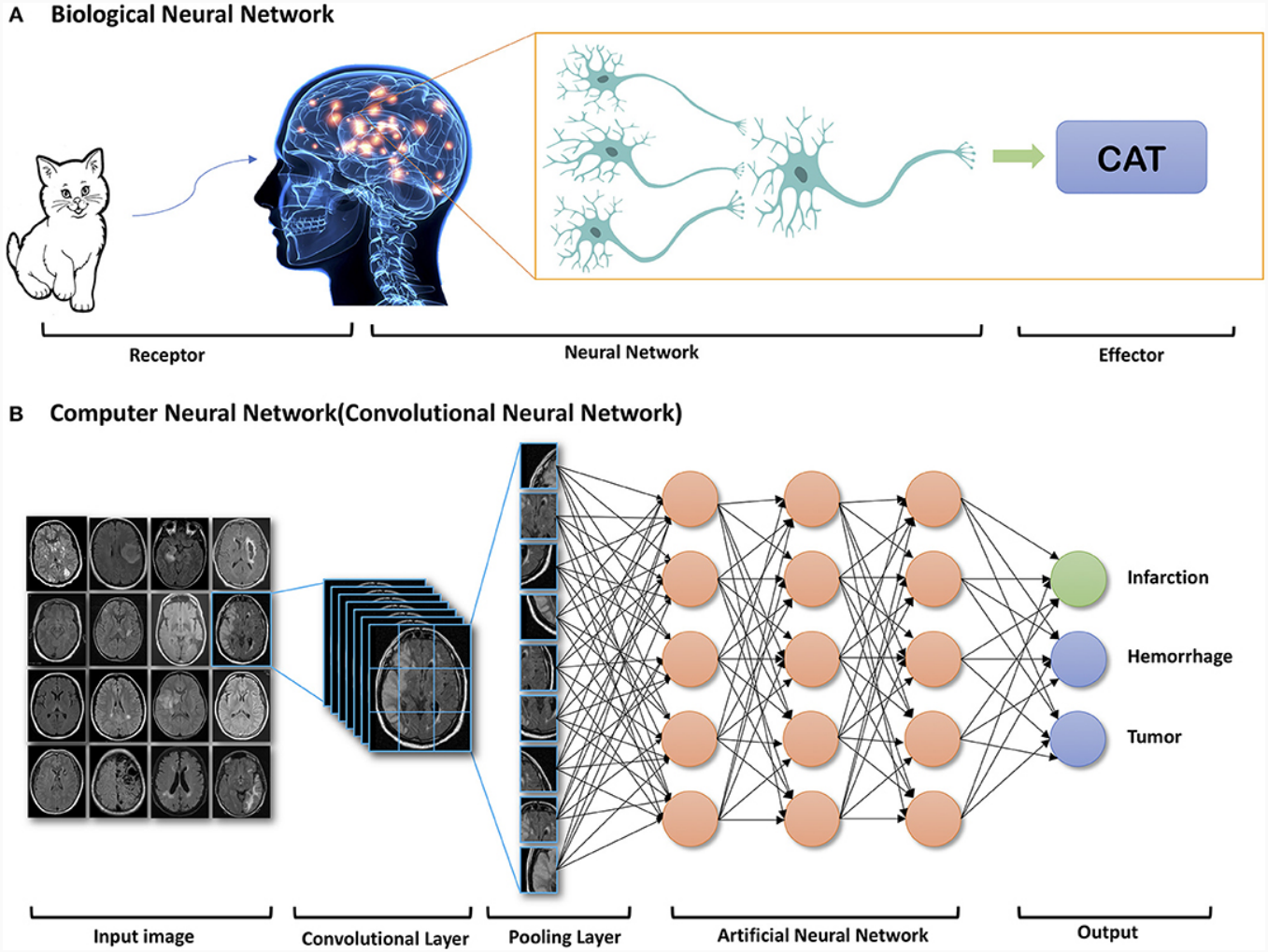
# Word2Vec

**Intuition:** Algorithm learns patterns based on the number of times each pairing shows up around a given context window.

- **Training**: more samples of `(Soviet, Union)` than `(Soviet, Sasquatch)`.
- **Learning**: if you give it the word `Soviet` as input, then it will output a much higher probability for `Union` or `Russia` than it will for `Sasquatch.`
- **How?** Neural networks! 🙂

# Neural networks for text analysis

Many (most?) word embedding models use **neural networks**, especially transformers architecture.



# Neural Networks for Text Analysis

**Main Idea:** Neural networks mimic human decision-making in pattern recognition and classification. It makes predictions by identifying patterns using **"hidden layers"** between inputs and then translating that into a predicted output.

A **word2vec** model is a neural network with a single hidden layer that was trained to predict the context of words. It finds patterns of word co-occurrences within a context window.

# Word2Vec architecture

**Example:** Consider a corpus with 10,000 unique words.

Key Terminology:

- **Task**: Learning goal is to try to predict the probability that other words are proximate to the input word.
- **Input**: target word represented as a one-hot vector (0-1 encoding) (length = 10,000).
- **Hidden Layer**: weight matrix explaining the multi-dimensional word vector, i.e. a learned representation of the input in order to make the best predictions (This is the secret sauce.)
- **Output**: a vector (length = 10,000) of probabilities that a given vocabulary word is in the context window of the target.

# Word2Vec architecture

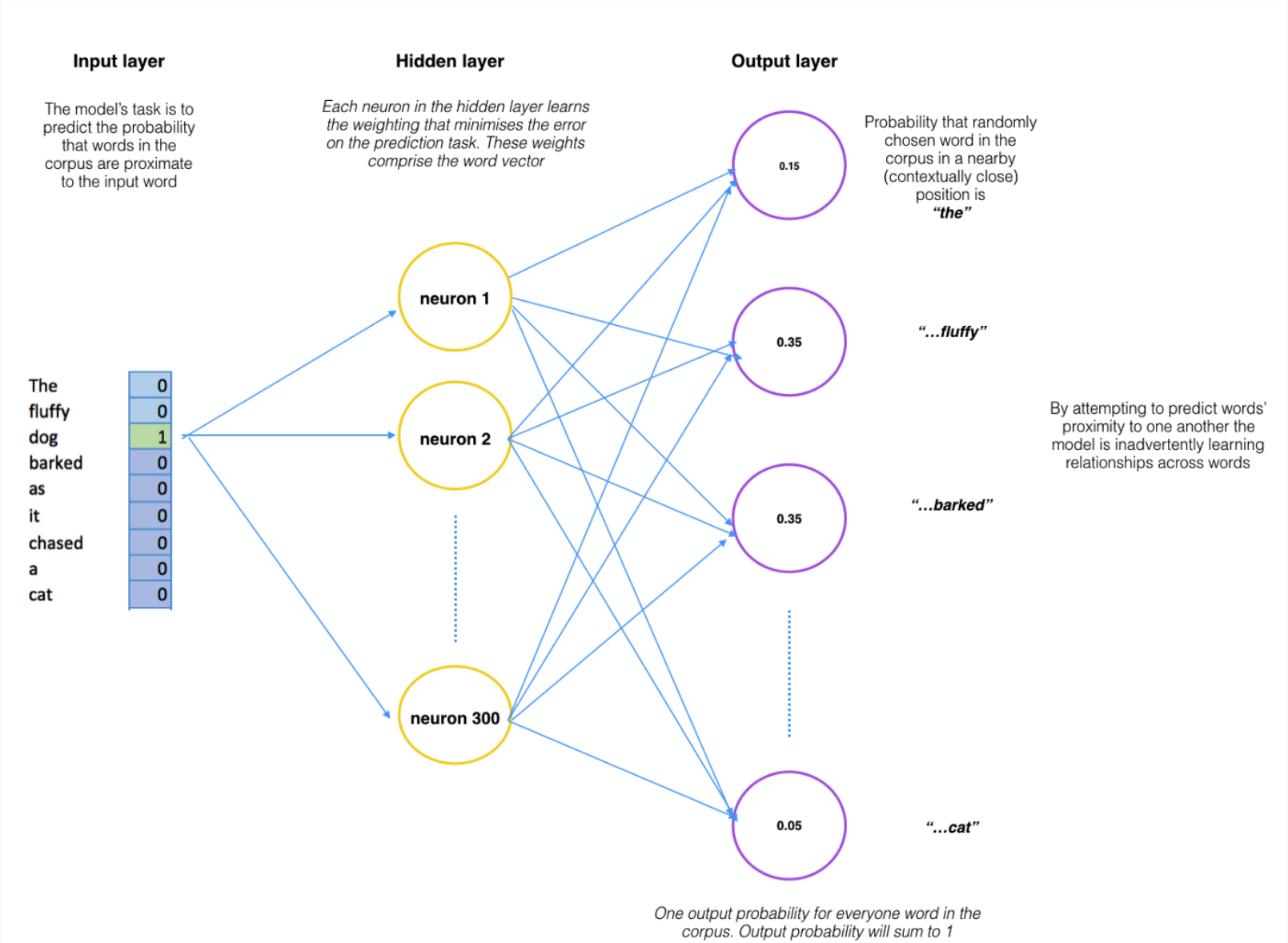



**Figure 13:** Source: Rochelle Terman

# Main Word2Vec Models

1. **Skip-Gram Model**
    - predicts the context words, given the center word
    - most useful for people who want to identify patterns within texts
    - limit: requires lots of data for stable representations
2. **Continuous Bag of Words (CBOW)**
    - tries to predict the center word, given context
    - faster to train
    - more useful in practical applications such as predictive web search

Alternatives to Word2Vec: GloVe, FastText, BERT ...

# Clustering and Topic Modeling

# Clustering Texts

Motivation: Suppose we want to group texts into meaningful themes, dimensions, topics, etc. How?

1. Embeddings → Qualitative Overlay
2. K-Means Clustering
3. Topic Modeling

# Types of Text Clustering

1. **Single Membership Models**:

2. **Mixed Membership Models**:

# Types of Text Clustering

1. **Single Membership Models**:
   - Each document assigned to one cluster
   - Example: Headline1 is about terror attack in Afghanistan; Headline4 is about monetary policy in Japan
2. **Mixed Membership Models**:

# Types of Text Clustering

1. **Single Membership Models**:

2. **Mixed Membership Models**:
   - Each document assigned to multiple cluster
   - Interested in $P(\text{topic} \mid \text{document})$
   - Example: Headline1 and Headline7 are about political violence; Headline4, Headline5, and Headline6 are about macroeconomic policy

# K-Means Clustering

Procedure:

- Measure similarity of different words and documents → meaningful groupings (themes)
- Define number of clusters
- Apply k-means clustering to partition documents

K-means clustering is **single membership** because it considers each document to fall under one topic.

# K-Means Clustering Example

Goal: Cluster the following texts:

1. I like to eat broccoli and bananas
2. I ate a banana smoothie for breakfast
3. Hamsters and kittens are cute
4. My sister adopted a kitten

# K-Means Clustering Example

Inputs

- Document Term Matrix

| Doc | adopt | banana | breakfast | broccoli | cute | ate | hamster | kitten | like | smoothie |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 2 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 3 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| 4 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |

# K-Means Clustering Example

Inputs

- Document Term Matrix

| Doc | adopt | banana | breakfast | broccoli | cute | ate | hamster | kitten | like | smoothie |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 2 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 3 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| 4 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |

- $K = 2$ (food and pets)

# K-Means Clustering Example

Outputs

- Go through each word and assign p(word | topic) for $k$ clusters

# K-Means Clustering Example

Outputs

- Go through each word and assign p(word | topic) for $k$ clusters
- $\mu_k$: Cluster means/centroids
  - Topic A centroid centered around ‘ate’, ‘banana, breakfast’, ‘broccoli’
  - Topic B centroid centered around ‘adopt’, ‘cute’, ‘hamster’
- K-means algorithm will assign each observation to the cluster with the closest mean

# K-Means Clustering Example

Outputs

- Go through each word and assign p(word | topic) for $k$ clusters
- $\mu_k$: Cluster means/centroids
  - Topic A centroid centered around ‘ate’, ‘banana, breakfast’, ‘broccoli’
  - Topic B centroid centered around ‘adopt’, ‘cute’, ‘hamster’
- K-means algorithm will assign each observation to the cluster with the closest mean
- $C_k$ Cluster assignment:
  - $C_1$ : [Doc1, Doc2]
  - $C_2$ : [Doc3, Doc4]

# Topic Modeling

- Mixed membership model: each document is a *mixture* of different topics

# Topic Modeling

- Mixed membership model: each document is a *mixture* of different topics
- Uses **Latent Dirichlet Allocation (LDA)** unsupervised algorithm
- $P(\textbf{word } w \mid \textbf{topic } t)$: proportion of assignments to topic t over all documents that come from this word w. Tries to capture how many documents are in topic t because of word w.
- $P(\textbf{topic } t \mid \textbf{document } i)$: proportion of words in document i that are assigned to topic t. If a lot of words from document belongs to t, it is more probable that word w belongs to t.

# Advanced Clustering: Topic Modeling

- Assigns corpus elements to substantively meaningful categories or topics using the statistical correlations between words

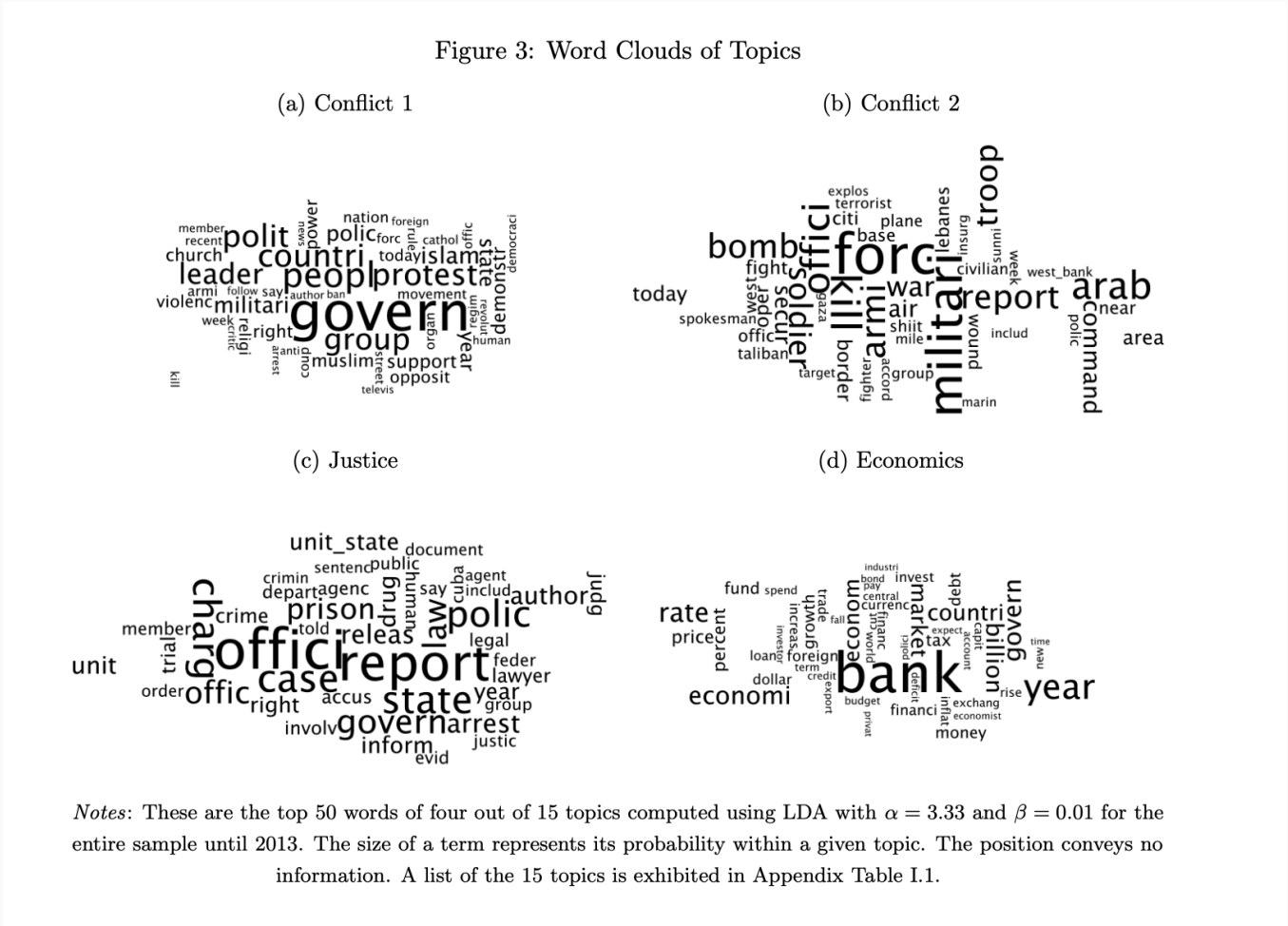



**Figure 14:** Source: Mueller and Rauh (2018)

# Choosing Best Text Metrics

Which topic method best distinguishes texts?

- Depends on context and goal
- Qualitative inference:
  - Face Validity: Do these results make sense?
  - Convergence: Do different metrics lead to the same results?
  - “Gold standard:” Do human inter-reliability tests match up?

# Limits to Clustering and Topic Modeling

- Topic models are super sensitive to feature selection
- Large-n requirement
- Hyperparameter tuning, esp. LDA
- What's the right number of topics?
- Mixed membership: Disambiguation vs interpretability

# Conclusion

- Text is context-dependent, but computational analysis is not
- Dictionary methods count frequency of words and assign weighted values
- Embeddings are the ‘hidden layer’ of a larger neural network to relate words to each other
- K-Means clustering and topic modeling for documents can work but is often noisy
- Validate. Validate. Validate.