# CS 333: Natural Language Processing

Fall 2025

Prof. Carolyn Anderson
Wellesley College

# Reminders

- Extension for HW 4: **due Monday at 10pm**
- No quiz next week
- Midterm 1: Oct. 10th
- My next help hours: Monday 4-5:30
- I'm **shifting my Thursday help hours** to 3-4 so they don't conflict with the CS Colloquium (Elena Glassman)

WELLESLEY
COMPUTER SCIENCE

CS COLLOQUIUM

# ELANA GLASSMAN

## CorpusStudio and AbstractExplorer: Reading and Writing Scientific Works at Scale


**Elana Glassman**
**Harvard University**
School of Engineering and Applied Sciences
glassmanlab.seas.harvard.edu/#glassman


**9th October 2025**

4:00 PM | SCI-H105
SNACKS WILL BE SERVED AT 3:45 PM

WELLESLEY

???: SB129
ACCESSIBILITY AND DISABILITY: ACCESSIBILITY@WELLESLEY.EDU

# Midterm 1

- In-class programming midterm
- Bring your own laptop to work on
- I will have starter code and documentation for you to download at the beginning, then ask you to turn off wifi.
- At the end, you will submit your code on Gradescope.
- May bring a 4x6in note card

Try to install PyTorch

# Curiosity Points on HW 3

- Further author identification experiments and analysis
- Visualizations of the partisan dataset and analysis
- Extra research on smoothing techniques
- Implementation of  extra sampling functions
- Additional functions for comparing authors
- Smoothing experiments
- Analysis of author style by investigating token frequencies
- Research and implementation of TF-IDF
- Author comparison on novel dataset

# Key Insight #1: Defining meaning by linguistic distribution

Let's define the meaning of a word by its distribution in language use, meaning its **neighboring words** or grammatical environments.

# Key Insight #2: Meaning as a point in multidimensional space

Each word is represented by a vector (not just "good" or "w45").

Similar words are "nearby in semantic space"

We build this space by seeing which words are nearby in text

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

# Take another look at our Austen word frequencies:

| | Emma | Persuasion | Sense & Sensibility |
|---|---|---|---|
| *admiral* | 0 | 69 | 0 |
| *dance* | 49 | 11 | 21 |
| *admire* | 31 | 14 | 18 |
| *horse* | 40 | 15 | 24 |

# Raw frequency is a bad representation

- Word counts for Emma are generally higher because it is a longer novel.
- Another issue: some words are so frequent that they aren't very informative: *the, it,* or *they*

# Solution 1: tf-idf

tf-idf: Term Frequency - Inverse Document Frequency

Term Frequency:

Inverse Document Frequency:

# Term Frequency

$$\mathrm{tf}_{t,d} = \mathrm{count}(t,d)$$

tf(admiral,Persuasion) = 69

tf(horse,Persuasion) = 15

| | Emma | Persuasion | Sense & Sensibility |
|---|---|---|---|
| *admiral* | 0 | 69 | 0 |
| *dance* | 49 | 11 | 21 |
| *admire* | 31 | 14 | 18 |
| *horse* | 40 | 15 | 24 |

# Inverse Document Frequency

$$\text{idf}_t = \frac{N}{\text{df}_t}$$

# of documents

N # of docc that contain term

idf(admiral) = $\frac{3}{1}$

idf(horse) = $\frac{3}{3}$

| | Emma | Persuasion | Sense & Sensibility |
|---|---|---|---|
| *admiral* | 0 | 69 | 0 |
| *dance* | 49 | 11 | 21 |
| *admire* | 31 | 14 | 18 |
| *horse* | 40 | 15 | 24 |

# TF-IDF

$$w_{t,d} = \text{tf}_{t,d} \times \text{idf}_t$$

tf-idf(admiral,Persuasion) = $69 \cdot 3 = 207$

tf-idf(horse,Persuasion) = $15 \cdot 1 = 15$

| | Emma | Persuasion | Sense & Sensibility |
|---|---|---|---|
| *admiral* | 0 | ~~69~~ 207 | 0 |
| *dance* | 49 | 11 | 21 |
| *admire* | 31 | 14 | 18 |
| *horse* | 40 | 15 | 24 |

# Solution 1: tf-idf

tf-idf: Term Frequency - Inverse Document Frequency

Term Frequency:

$$tf_{t,d} = count(t,d)$$

$$tf_{t,d} = \log_{10}(count(t,d)+1)$$

Inverse Document Frequency:

$$idf_t = \frac{N}{df_t}$$

$$idf_t = \log_{10}\left(\frac{N}{df_t}\right)$$

$$tf\text{-}idf: \; w_{t,d} = tf_{t,d} \times idf_t$$

Repeat for every term & every document

# What is a document?

Could be a play or a Wikipedia article.

But for the purposes of tf-idf, documents can be anything; we often call each paragraph a document!

# Word2Vec

# Sparse versus dense vectors

positive mutual information

tf-idf (or PMI) vectors are:

- **long** (length |V|= 20,000 to 50,000)
- **sparse** (most elements are zero)

Alternative: learn vectors that are:

- **short** (length 50-1000)
- **dense** (most elements are non-zero)

# Sparse versus dense vectors

Why dense vectors?

- Short vectors may be easier to use as **features** in machine learning (fewer weights to tune)
- Dense vectors may **generalize** better than explicit counts
- In practice, they work better

# Simple static embeddings you can download!

Word2vec (Mikolov et al)

https://code.google.com/archive/p/word2vec/

GloVe (Pennington, Socher, Manning)

http://nlp.stanford.edu/projects/glove/

Static : each word has a unique vector representation

Contextual : words have different representations depending on the sentence they appear in.

# Word2Vec

Word2Vec is a popular embedding method that is very fast to train.

Idea: **predict** rather than **count**

Word2Vec provides various options for how to learn embeddings. We'll discuss:

**skip-gram with negative sampling (SGNS)**

# Word2Vec

Instead of **counting** how often each word *w* occurs near "*apricot*"

- Train a classifier on a binary prediction task:
  - Is *w* likely to show up near "*apricot*"?

We don't actually care about this task

- We'll take the **learned classifier weights** as the word embeddings

Big idea: **self-supervision**:

- A word c that occurs near apricot in the corpus ~~cats~~ as the gold "correct answer" for supervised learning
- **No need for human labels**
- Bengio et al. (2003); Collobert et al. (2011)

# Approach: predict if candidate word *c* is a "neighbor"

1. Treat the target word t and a neighboring context word c as positive examples.

# Approach: predict if candidate word *c* is a "neighbor"

1. Treat the target word t and a neighboring context word c as positive examples.
2. Randomly sample other words in the lexicon to get negative examples

# Approach: predict if candidate word *c* is a "neighbor"

1. Treat the target word t and a neighboring context word c as positive examples.
2. Randomly sample other words in the lexicon to get negative examples
3. Use logistic regression to train a classifier to distinguish those two cases

# Approach: predict if candidate word *c* is a "neighbor"

1. Treat the target word t and a neighboring context word c as positive examples.
2. Randomly sample other words in the lexicon to get negative examples
3. Use logistic regression to train a classifier to distinguish those two cases
4. Use the learned weights as the embeddings

# Skip-Gram Training Data

Assume a +/- 2 word window, given training sentence:

…lemon, a [tablespoon of apricot jam, a] pinch…

c1 c2 [target] c3 c4

# Skip-Gram Classifier

(assuming a +/- 2 word window)

…lemon, a [tablespoon of apricot jam, a] pinch…

c1 c2 [target] c3 c4

Goal: train a classifier that is given a candidate (**w**ord, **c**ontext) pair

(apricot, jam) positive

(apricot, aardvark) negative

…

And assigns each pair a probability:

$P(+|w,c)$

$P(-|w,c) = 1 - P(+|w,c)$

# Similarity is computed from dot product

Remember: two vectors are similar if they have a high dot product

- Cosine is just a normalized dot product

So:

- Similarity(w,c) $\propto w \cdot c$

We'll need to normalize to get a probability

- (cosine isn't a probability either)

# Turning dot products into probabilities

Sim(w,c) ≈ w · $c$

To turn this into a probability, we'll use the *sigmoid function*:

$$\sigma(x) = \frac{1}{1+\exp(-x)}$$

$$P(+|w,c) = \sigma(c \cdot w) = \frac{1}{1+\exp(-c \cdot w)}$$

$$P(-|w,c) = 1 - P(+|w,c)$$

$$= \sigma(-c \cdot w) = \frac{1}{1+\exp(c \cdot w)}$$

# Turning dot products into probabilities

Sim(w,c) ≈ w · $c$

To turn this into a probability, we'll use the *sigmoid function*:

$$\sigma(x) = \frac{1}{1+\exp(-x)}$$

$$P(+|w,c) =$$

$$P(-|w,c) =$$

# How Skip-Gram Classifier computes $P(+|w, c)$

$$P(+|w,c) = \sigma(\mathbf{c}\cdot\mathbf{w}) = \frac{1}{1+\exp(-\mathbf{c}\cdot\mathbf{w})}$$

This is for one context word, but we have lots of context words.

We'll **assume independence** and multiply them:

$$P(+|w, c_{1:L}) = \prod_{i=1}^{L} \sigma(c_i \cdot w)$$

$$\log P(+|w, c_{1:L}) = \sum_{i=1}^{L} \log \sigma(c_i \cdot w)$$

# Skip-gram classifier: summary

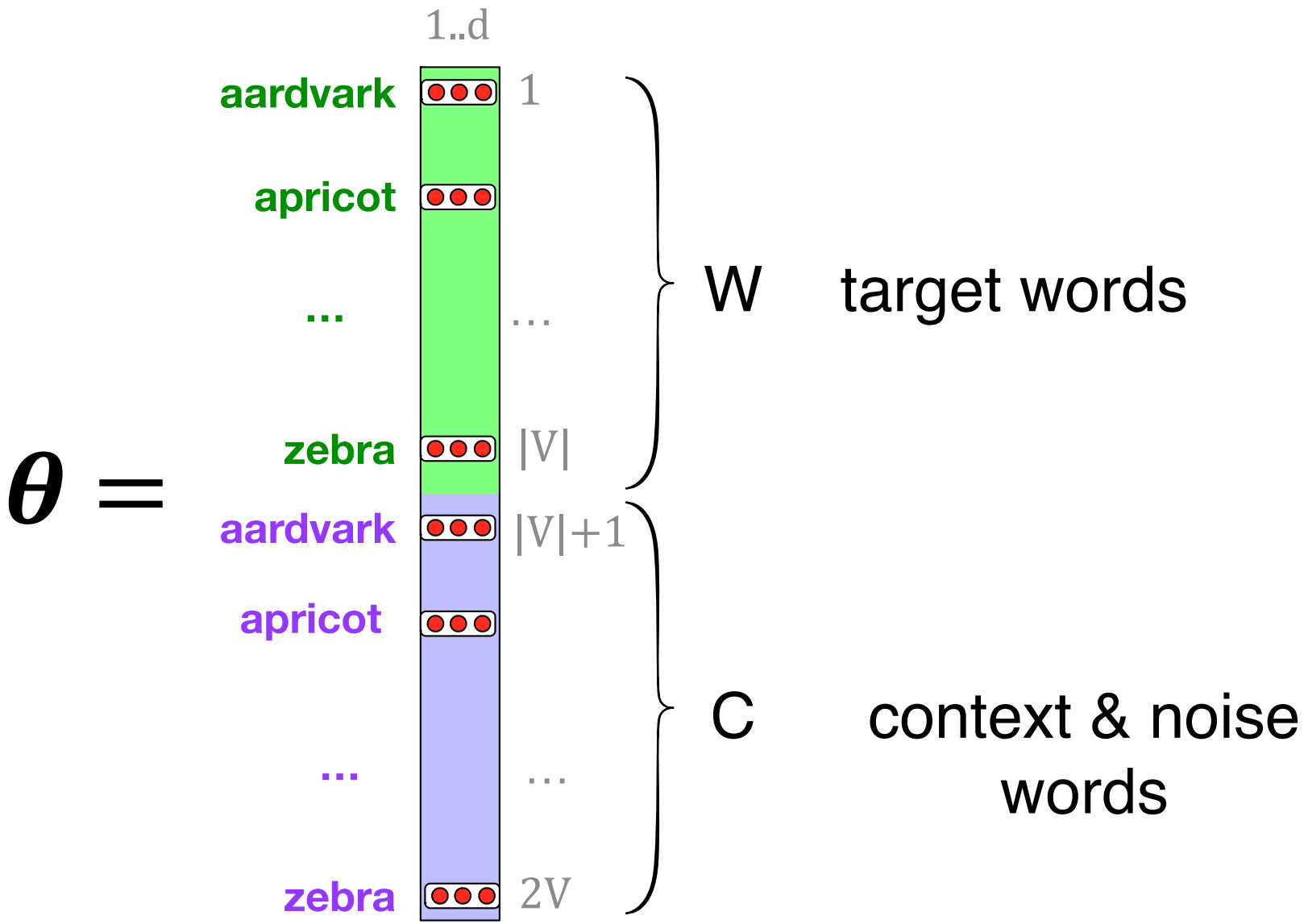
A probabilistic classifier, given

- a test target word $w$
- its context window of $L$ words $c_{1:L}$

Estimates probability that w occurs in this window based on similarity of w (embeddings) to $c_{1:L}$ (embeddings).

To compute this, we just need embeddings for all the words.

# Embeddings we'll need: a set for w, a set for c



# Word2Vec: Learning embeddings

# Skip-Gram Training data

…lemon, a [tablespoon of apricot jam, a] pinch…

c1 c2 [target] c3 c4

**positive examples +**

| $w$ | $c_{pos}$ |
|---|---|
| apricot | tablespoon |
| apricot | of |
| apricot | jam |
| apricot | a |

For each positive example we take k negative examples, sampled by word frequency.

# Skip-Gram Training data

…lemon, a [tablespoon of apricot jam, a] pinch…

c1 c2 [target] c3 c4

**positive examples +**

| $w$ | $c_{pos}$ |
|---|---|
| apricot | tablespoon |
| apricot | of |
| apricot | jam |
| apricot | a |

For each positive example we'll grab k negative examples, sampling by frequency.

# Skip-Gram Training data

…lemon, a [tablespoon of apricot jam, a] pinch…

c1 c2 [target] c3 c4

**positive examples +**

| $w$ | $c_{pos}$ |
|---|---|
| apricot | tablespoon |
| apricot | of |
| apricot | jam |
| apricot | a |

OBSERVED

**negative examples -**

| $w$ | $c_{neg}$ | $w$ | $c_{neg}$ |
|---|---|---|---|
| apricot | aardvark | apricot | seven |
| apricot | my | apricot | forever |
| apricot | where | apricot | dear |
| apricot | coaxial | apricot | if |

MADE UP

# Word2vec: how to learn vectors

Given the set of positive and negative training instances, and an initial set of embedding vectors

Goal: adjust the word vectors so they:

- Maximize the similarity of target & context word pairs $(w, c_{pos})$

- Minimize the similarity of the negative examples & target word pairs $(w, c_{neg})$

# Word2vec: how to learn vectors

Given the set of positive and negative training instances, and an initial set of embedding vectors

Goal: adjust the word vectors so they:

- Maximize the similarity of the target word, context word pairs (w , cpos) drawn from the positive data

# Word2vec: how to learn vectors

Given the set of positive and negative training instances, and an initial set of embedding vectors

Goal: adjust the word vectors so they:

- Maximize the similarity of the target word, context word pairs (w , cpos) drawn from the positive data
- Minimize the similarity of the (w , cneg) pairs drawn from the negative data.

# Loss function for one $w$ with $c_{pos}$, $c_{neg1}$ ...$c_{negk}$

Maximize the similarity of the target with the actual context words, and minimize the similarity of the target with the k negative sampled non-neighbor words.

$$L_{CE} = -\log \left[ p(+|w, c_{pos}) \prod_{i=1}^{k} p(-|w, c_{neg_i}) \right]$$

$$= -\left[ \log p(+|w, c_{pos}) + \sum_{i=1}^{k} \log p(-|w, c_{neg_i}) \right]$$

$$= -\left[ \log p(+|w, c_{pos}) + \sum_{i=1}^{k} \log \left(1 - p(+|w, c_{neg_i})\right) \right]$$

How wrong the model is

$$= -\left[ \log \sigma(c_{pos} \cdot w) + \sum_{i=1}^{k} \log \sigma(-c_{neg_i} \cdot w) \right]$$

# Learning the classifier

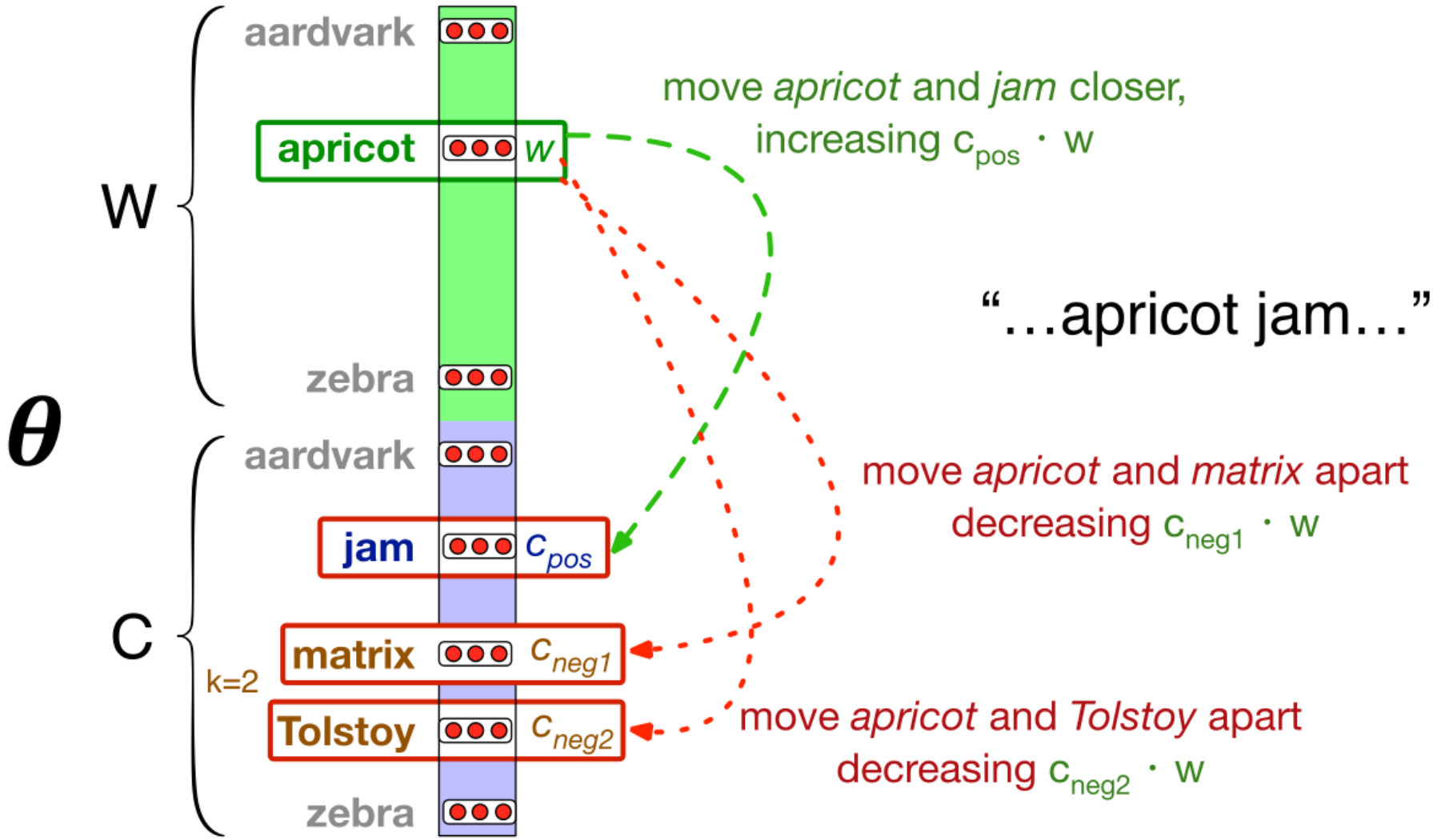
How to learn?

- Stochastic gradient descent!

We'll adjust the word weights to

- make the positive pairs more likely
- and the negative pairs less likely,
- over the entire training set.

# Intuition of one step of gradient descent



# Two sets of embeddings

SGNS learns two sets of embeddings

Target embeddings matrix W

Context embedding matrix C

It's common to just **add them together**, representing word i as the vector **wi + ci**

vector for apricot : $w_{apricot} + c_{apricot}$

# Summary: How to learn word2vec (skip-gram) embeddings

Start with V random d-dimensional vectors as initial embeddings

Train a classifier based on embedding similarity

- Take a corpus and take pairs of words that co-occur as positive examples
- Take pairs of words that don't co-occur as negative examples
- Train the classifier to distinguish these by slowly adjusting all the embeddings to improve the classifier performance
- Throw away the classifier code and keep the embeddings.

# Word2Vec

https://semantle.com/

# Properties of Embeddings

# The kinds of neighbors depend on window size

Small windows (C= +/- 2) : nearest words are syntactically similar words in same taxonomy

- Hogwarts nearest neighbors are other fictional schools: *Sunnydale, Evernight, Blandings*

Large windows (C= +/- 5) : nearest words are related words in same semantic field

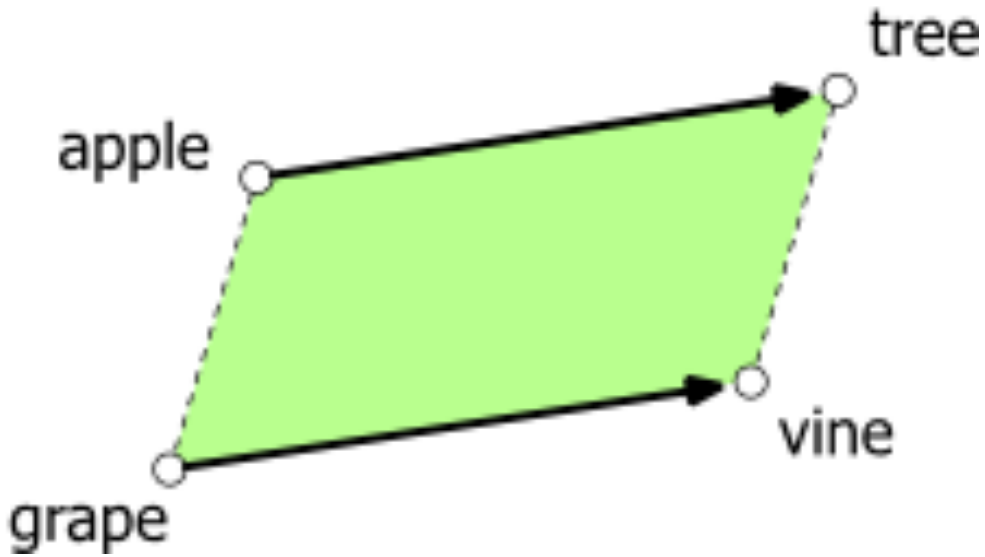
- Hogwarts nearest neighbors are Harry Potter world: *Dumbledore, half-blood, Malfoy*

# Analogical relations

The classic parallelogram model of analogical reasoning (Rumelhart and Abrahamson 1973).

To solve: *"apple is to tree as grape is to _____"*

Add tree – apple  to grape to get vine



# Analogical relations via parallelogram

The parallelogram method can solve analogies with both sparse and dense embeddings (Turney and Littman 2005, Mikolov et al. 2013b):

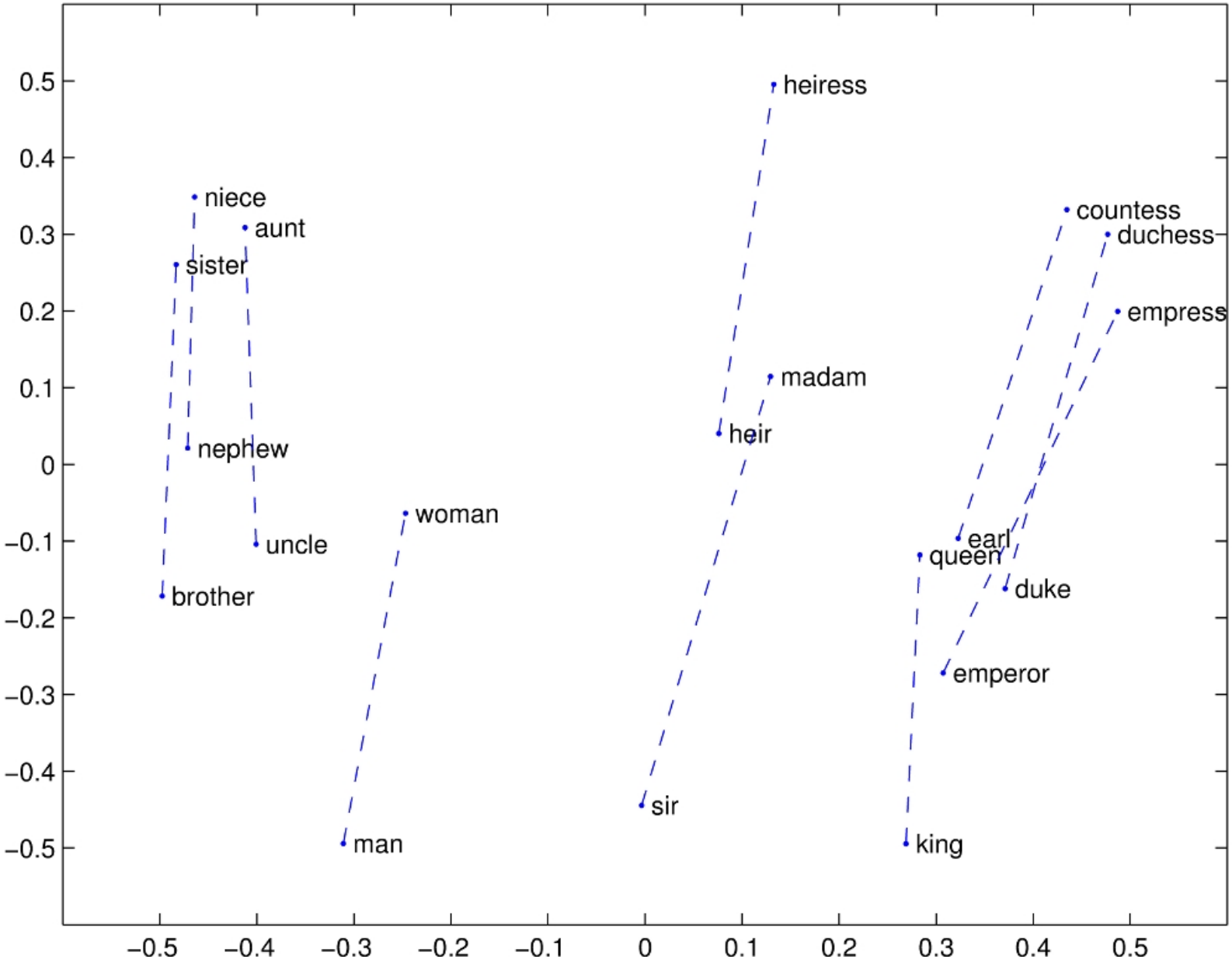
*king* – *man* + *woman* is close to *queen*

*Paris* – *France* + *Italy* is close to *Rome*

For a problem a:a*::b:b*, the parallelogram method is:

$$\hat{\mathbf{b}}^* = \underset{\mathbf{x}}{\text{argmin}}\ \text{distance}(\mathbf{x}, \mathbf{b} - \mathbf{a} + \mathbf{a}^*)$$

# Structure in GloVE Embedding space



# Word Embedding Interactive Demo

https://www.cs.cmu.edu/~dst/WordEmbeddingDemo/index.html

# Embeddings reflect cultural bias!

Ask “Paris : France :: Tokyo : x”

- x = Japan

Ask “father : doctor :: mother : x”

- x = nurse

Ask “man : computer programmer :: woman : x”

- x = homemaker

Algorithms that use embeddings might be biased as a result.

Bolukbasi, Tolga, Kai-Wei Chang, James Y. Zou, Venkatesh Saligrama, and Adam T. Kalai. "Man is to computer programmer as woman is to homemaker? debiasing word embeddings." In *NeurIPS*, pp. 4349-4357. 2016.

# Caveat: Limitations of Gender Bias Approaches

## Harms of Gender Exclusivity and Challenges in Non-Binary Representation in Language Technologies

**Sunipa Dev**
she/her
UCLA

**Masoud Monajatipoor***
he/him
UCLA

**Anaelia Ovalle***
they/he/she
UCLA

**Arjun Subramonian***
they/them
UCLA, Queer in AI

**Jeff M Phillips**
he/him
University of Utah

**Kai-Wei Chang**
he/him
UCLA

### Abstract

*Content Warning:* This paper contains examples of stereotypes and associations, misgendering, erasure, and other harms that could be offensive and triggering to trans and non-binary individuals.

Gender is widely discussed in the context of language tasks and when examining the stereotypes propagated by language models. However, current discussions primarily treat gender as binary, which can perpetuate harms such as the cyclical erasure of non-binary gender

A bulk of social bias studies on language models have focused on binary gender and the stereotypes associated with masculine and feminine attributes (Bolukbasi et al., 2016; Webster et al., 2018; Dev et al., 2020b). Additionally, models often rely on gendered information for decision making, such as in named entity recognition, coreference resolution, and machine translation (Mehrabi et al., 2020; Zhao et al., 2018; Stanovsky et al., 2019), but the purview of gender in these tasks and associated measures of performance focus on binary gender. While discussing binary gender bias

# Caveats with the parallelogram method

It only seems to work for frequent words, small distances and certain relations (relating countries to capitals, or parts of speech), but not others. (Linzen 2016, Gladkova et al. 2016, Ethayarajh et al. 2019a).

Understanding analogy is an open area of research (Peterson et al. 2020)

# Embeddings as a window onto historical semantics

Train embeddings on different decades of historical text to see meanings shift

~30 million books, 1850-1990, Google Books data


William L. Hamilton, Jure Leskovec, and Dan Jurafsky. 2016. Diachronic Word Embeddings Reveal Statistical Laws of Semantic Change. Proceedings of ACL.

# Historical embedding as a tool to study cultural biases

- Compute a **gender or ethnic bias** for each adjective.
  - Embeddings for **competence** adjective (*smart, wise, brilliant, resourceful, thoughtful, logical)* are biased toward men, a bias slowly decreasing 1960-1990
  - Embeddings for **dehumanizing** adjectives (barbaric, monstrous, bizarre) were biased toward Asians in the 1930s, bias decreasing over the 20th century.
- These match the results of old surveys done in the 1930s

Garg, N., Schiebinger, L., Jurafsky, D., and Zou, J. (2018). Word embeddings quantify 100 years of gender and ethnic stereotypes. Proceedings of the National Academy of Sciences 115(16), E3635–E3644.