# Distributional Semantics and Embeddings

Marion Di Marco

December 3, 2024

# Outline

# Distributional Hypothesis

- Humans infer meaning from the context McDonald & Ramscar (2001)

- What is a *wampimuk*?

  *"He filled the* **wampimuk**, *passed it around and we all drunk some."*

  A container for drinks?

  *"We found a little, hairy* **wampimuk** *sleeping behind the tree."*

  A little animal?

# Distributional Hypothesis

- What other words could occur in this context?

*"He filled the ___, passed it around and we all drunk some."*

  - cup
  - goblet
  - bottle

*"We found a little, hairy ___ sleeping behind the tree."*

  - monkey
  - squirrel
  - rabbit

# Distributional Hypothesis

- Words that occur in similar *contexts* tend to have similar *meanings*
- Firth (1957):
  *"You shall know a word by the company it keeps."*
- Vectors as word representations to describe the properties of each word
- Distributional statistics: core part of language technology
  leverage large amounts of unlabeled data to learn about (rare) words

# Constructing a Distributional Semantic Model

understand the causes of these climate anomalies is important not only for historical information on how Earth 's climate has changed over thousands and thousands of oceans are also threatened by climate change , with temperature rises of

- Context: words next to **w** in a fixed-size window
- Use contexts of all observations of **w** to build a representation of **w**
- Collecting context words of **w** (content words only):

| | |
|---|---|
| causes | 1 |
| anomalies | 1 |
| important | 1 |
| Earth | 1 |
| changed | 1 |
| threatened | 1 |
| change | 1 |

# Context Design

- Size of the context window
- Differentiate: left-side context and right-side context
- All words in a windos vs. selected words in a window,
  lemma vs. inflected forms
- POS-coded: $\text{change}_{NOUN}$ vs. $\text{change}_{VERB}$
- Structured information: dependency information
  (e.g. SUBJ, OBJ, PP relations)
  - dependency-filtered: consider only particular relations
    *postman: bite*, *dog: bite*
  - dependency-linked: also keep information of the depency path
    *postman: obj-of-bite*, *dog: subj-of-bite*

$\Rightarrow$ Task-specific

# Words as Vectors: Word Dimensions and Similarity

| | aardvark | ... | computer | data | result | pie | sugar | ... |
|---|---|---|---|---|---|---|---|---|
| **cherry** | 0 | ... | 2 | 8 | 9 | 442 | 25 | ... |
| **strawberry** | 0 | ... | 0 | 0 | 1 | 60 | 19 | ... |
| **digital** | 0 | ... | 1670 | 1683 | 85 | 5 | 4 | ... |
| **information** | 0 | ... | 3325 | 3982 | 378 | 5 | 13 | ... |

**Figure 6.6** Co-occurrence vectors for four words in the Wikipedia corpus, showing six of the dimensions (hand-picked for pedagogical purposes). The vector for *digital* is outlined in red. Note that a real vector would have vastly more dimensions and thus be much sparser.

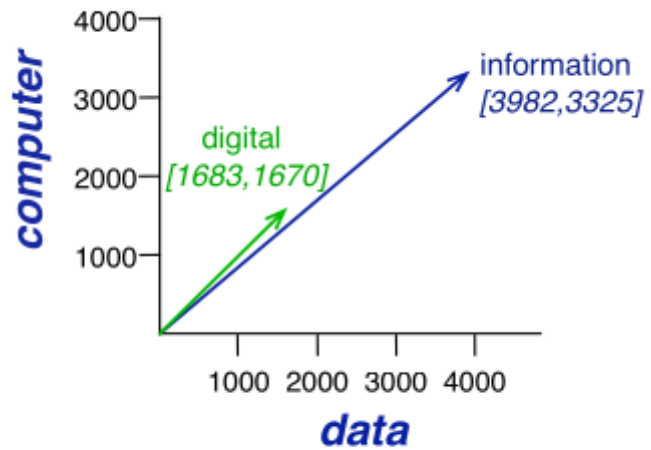



**Figure 6.7** A spatial visualization of word vectors for *digital* and *information*, showing just two of the dimensions, corresponding to the words *data* and *computer*.

Figures from Jurafsky and Martin

# Cosine to Measure Similarity

- Measure similarity between vectors of the words *v* and *w*
- Common similarity measure: *cosine* of the angle between the vectors

$$\text{cosine}(\mathbf{v},\mathbf{w}) = \frac{\mathbf{v}\cdot\mathbf{w}}{|\mathbf{v}||\mathbf{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}} \qquad (6.10)$$

- Values range from 1 (= same direction) to 0 (for orthogonal vectors)

  -1 for opposite directions, but we don't get this with non-negative frequencies

Figure from Jurafsky and Martin

# Cosine to Measure Similarity

| | pie | data | computer |
|---|---|---|---|
| **cherry** | 442 | 8 | 2 |
| **digital** | 5 | 1683 | 1670 |
| **information** | 5 | 3982 | 3325 |

$$\cos(\text{cherry}, \text{information}) = \frac{442*5+8*3982+2*3325}{\sqrt{442^2+8^2+2^2}\sqrt{5^2+3982^2+3325^2}} = .018$$

$$\cos(\text{digital}, \text{information}) = \frac{5*5+1683*3982+1670*3325}{\sqrt{5^2+1683^2+1670^2}\sqrt{5^2+3982^2+3325^2}} = .996$$

$\Rightarrow$ *information* is closer to *digital* than to *cherry*

Figure from Jurafsky and Martin

# Cosine to Measure Similarity

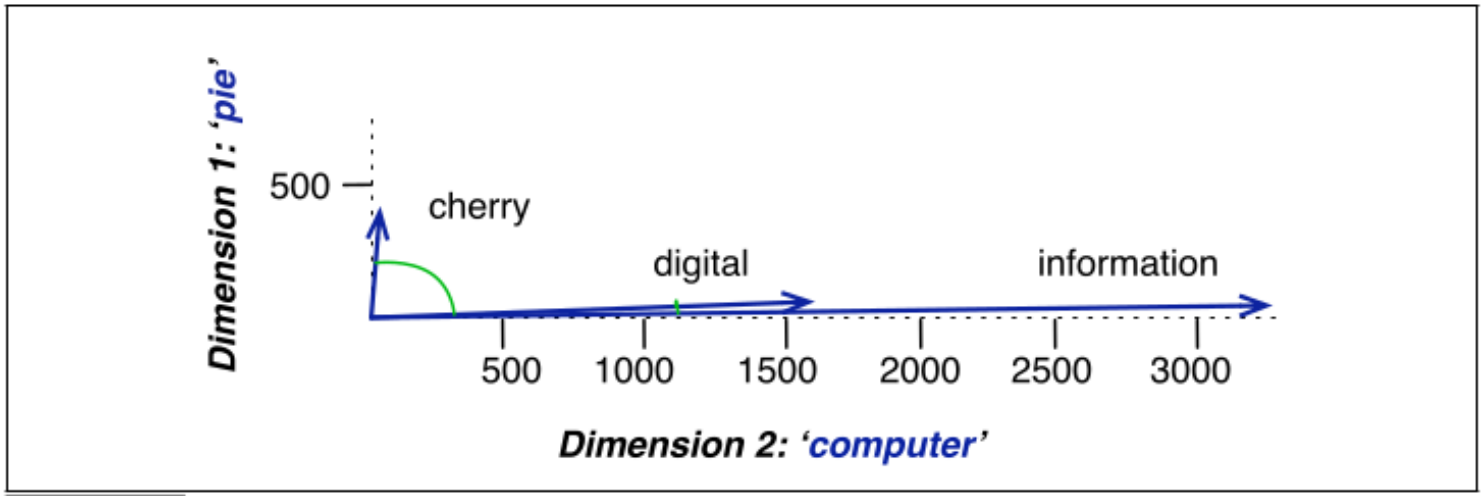



**Figure 6.8** A (rough) graphical demonstration of cosine similarity, showing vectors for three words (*cherry*, *digital*, and *information*) in the two dimensional space defined by counts of the words *computer* and *pie* nearby. The figure doesn't show the cosine, but it highlights the angles; note that the angle between *digital* and *information* is smaller than the angle between *cherry* and *information*. When two vectors are more similar, the cosine is larger but the angle is smaller; the cosine has its maximum (1) when the angle between two vectors is smallest ($0^\circ$); the cosine of all other angles is less than 1.

Figure from Jurafsky and Martin

# Outline

# Context Weighting

- Raw context counts → scores taking into account the relevance of an observed context
- Association measures: higher weight to context words highly associated with target word
- Co-occurrence with a frequent context element is less informative: the less frequent a context element is → the higher the weight should be for the observed co-occurrence
- Different measures, for example point-wise mutual information

# Point-wise Mutual Information

- Intuition: how much more do two words occur in a corpus than we would expect at chance?

$$\text{PMI}(w,c) = \log_2 \frac{P(w,c)}{P(w)P(c)} \qquad (6.17)$$

- Numerator: how often we observed the word and the context together
- Denominator: how often we expect the two words to co-occur
  Probability of two independent events both occurring: product of the probabilities of the events
- Ratio: estimate of how much more the two words co-occur than we expect by chance

Figure from Jurafsky and Martin

# Point-wise Mutual Information

- PMI scores lie between $-\infty$ and $+\infty$

- Negative PMI means that words are co-occurring less than we expect by chance
- Negative PMI values tend to be unreliable unless the corpora are enormous

  Unclear whether scores of "unrelatedness" are meaningful

- Positive point-wise mutual information: replace negative values with 0

$$\text{PPMI}(w,c) = \max(\log_2 \frac{P(w,c)}{P(w)P(c)}, 0) \qquad (6.18)$$

Figure from Jurafsky and Martin

# Point-wise Mutual Information: Example

| | computer | data | result | pie | sugar | count(w) |
|---|---|---|---|---|---|---|
| **cherry** | 2 | 8 | 9 | 442 | 25 | 486 |
| **strawberry** | 0 | 0 | 1 | 60 | 19 | 80 |
| **digital** | 1670 | 1683 | 85 | 5 | 4 | 3447 |
| **information** | 3325 | 3982 | 378 | 5 | 13 | 7703 |
| **count(context)** | 4997 | 5673 | 473 | 512 | 61 | 11716 |

**Figure 6.10** Co-occurrence counts for four words in 5 contexts in the Wikipedia corpus, together with the marginals, pretending for the purpose of this calculation that no other words/contexts matter.

Thus for example we could compute PPMI(information,data), assuming we pretended that Fig. 6.6 encompassed all the relevant word contexts/dimensions, as follows:

$$P(\text{w=information, c=data}) = \frac{3982}{11716} = .3399$$

$$P(\text{w=information}) = \frac{7703}{11716} = .6575$$

$$P(\text{c=data}) = \frac{5673}{11716} = .4842$$

$$\text{PPMI(information,data)} = \log_2(.3399/(.6575 * .4842)) = .0944$$

Figure from Jurafsky and Martin

# Point-wise Mutual Information: Example

| | p(w,context) | | | | | p(w) |
|---|---|---|---|---|---|---|
| | **computer** | **data** | **result** | **pie** | **sugar** | **p(w)** |
| **cherry** | 0.0002 | 0.0007 | 0.0008 | 0.0377 | 0.0021 | 0.0415 |
| **strawberry** | 0.0000 | 0.0000 | 0.0001 | 0.0051 | 0.0016 | 0.0068 |
| **digital** | 0.1425 | 0.1436 | 0.0073 | 0.0004 | 0.0003 | 0.2942 |
| **information** | 0.2838 | 0.3399 | 0.0323 | 0.0004 | 0.0011 | 0.6575 |
| **p(context)** | 0.4265 | 0.4842 | 0.0404 | 0.0437 | 0.0052 | |

**Figure 6.11** Replacing the counts in Fig. 6.6 with joint probabilities, showing the marginals in the right column and the bottom row.

| | **computer** | **data** | **result** | **pie** | **sugar** |
|---|---|---|---|---|---|
| **cherry** | 0 | 0 | 0 | 4.38 | 3.30 |
| **strawberry** | 0 | 0 | 0 | 4.10 | 5.51 |
| **digital** | 0.18 | 0.01 | 0 | 0 | 0 |
| **information** | 0.02 | 0.09 | 0.28 | 0 | 0 |

**Figure 6.12** The PPMI matrix showing the association between words and context words, computed from the counts in Fig. 6.11. Note that most of the 0 PPMI values are ones that had a negative PMI; for example PMI(*cherry,computer*) = -6.7, meaning that *cherry* and *computer* co-occur on Wikipedia less often than we would expect by chance, and with PPMI we replace negative values by zero.

Figure from Jurafsky and Martin

# Point-wise Mutual Information: Rare Words

- PMI has the problem of being biased toward infrequent events
  very rare words tend to have very high PMI values
- Reduce bias toward low frequency: change the computation for P(c)
- Function $P_\alpha(c)$: raise the probability of $c$ to the power of $\alpha$

$$\text{PPMI}_\alpha(w,c) = \max(\log_2 \frac{P(w,c)}{P(w)P_\alpha(c)}, 0) \qquad (6.21)$$

$$P_\alpha(c) = \frac{count(c)^\alpha}{\sum_c count(c)^\alpha} \qquad (6.22)$$

- In practice: $\alpha = 0.75$ leads to good results

Figure from Jurafsky and Martin

# Outline

# Word2Vec

- What we want: put information about contexts into vectors

  count-based methods: long, sparse vectors; dimensions corresponding to $|V|$

- Word2Vec: Learn vectors by teaching them to predict contexts

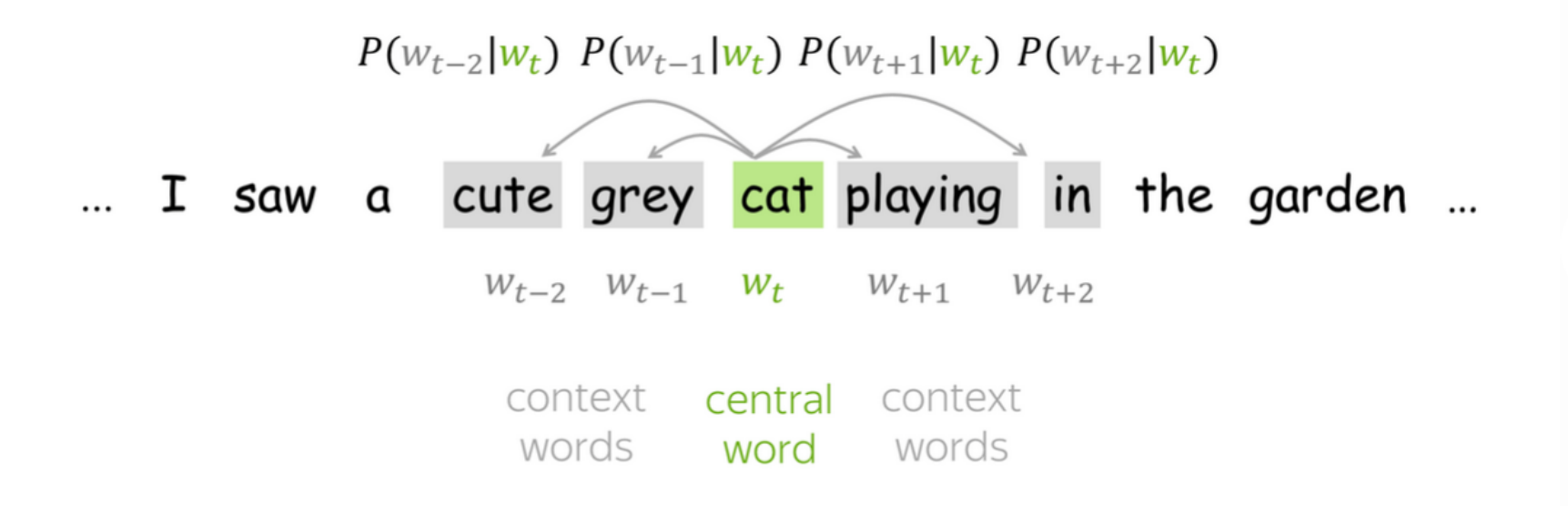
  Mikolov et al. 2013(a); Mikolov et al. 2013(b)

- Word2Vec's parameters are word vectors that are optimized for a certain objective

- Objective: make vectors "know" about the context of it word
  train vectors to predict possible contexts

- Hypothesis: "know" about context → "know" about meaning

# Word2Vec: Idea




- Main idea
  - all words in the vocabulary $V$ are represented by a vector
  - go over corpus with a sliding window: one central word and context words to the left and right
  - calculate the probability of a context word given the center word using the similarity of word the vectors
  - adjust vectors to maximize the probabilities

https://lena-voita.github.io/nlp_course/word_embeddings.html#w2v_idea

# Objective Function: Negative Log-Likelihood

- For each position $t = 1, \ldots, T$, predict context words in a fixed-size window of size $m$ given the central word $w_t$:

$$\text{Likelihood} = L(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \le j \le m \\ j \neq 0}} P(w_{t+j} \mid w_t; \theta)$$

$\theta$ is all variables to be optimized

- The objective function is the average negative log-likelihood

sometimes called a *cost* or *loss* function

The objective function $J(\theta)$ is the (average) negative log likelihood:

$$J(\theta) = -\frac{1}{T} \log L(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \neq 0}} \log P(w_{t+j} \mid w_t; \theta)$$

Minimizing objective function ⇔ Maximizing predictive accuracy

Figures from web.stanford.edu/class/cs224n/slides/cs224n-spr2024-lecture01-wordvecs1.pdf

# Calculate the Probabilities

- How to calculate $P(w_{t+j}|w_t, \theta)$ ?
- For each word $w$ we have two vectors
  - $v_w$ when it is a central word
  - $u_w$ when it is a context word

- *Softmax* to map arbitrary values $x_i$ to a probability distribution $p_i$

$$softmax(x_i) = \frac{exp(x_i)}{\sum_{j=i}^{n} exp(x_j)}$$

*max*: the largest $x_i$ will have the largest probability $p_i$
*soft*: all probabilities are non-zero

- $P(o|c) = \frac{exp(u_o^T v_c)}{\sum_{w \epsilon V} exp(u_w^T v_c)}$

dot product: measure similarity of o and c
larger dot product = larger probability

normalize over vocabulary to get probability distribution

where and c = central word and o = outside word

# Word2Vec: Word and Context Vectors

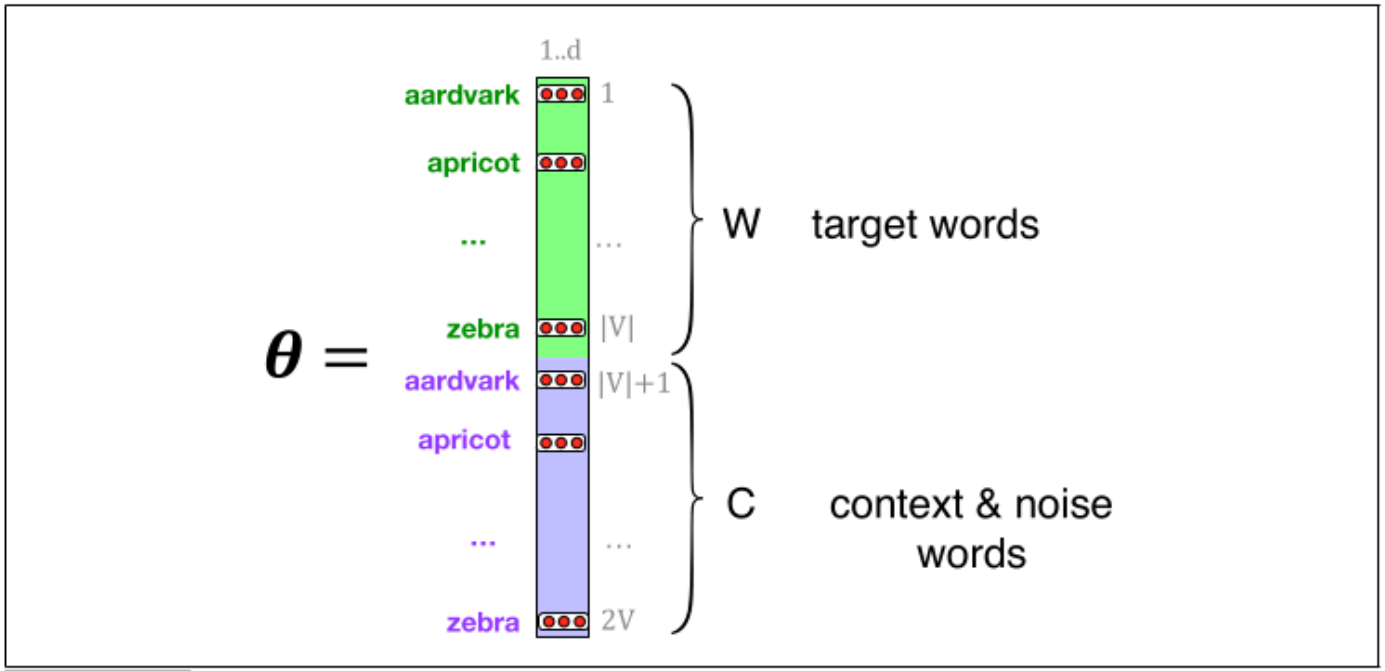



**Figure 6.13** The embeddings learned by the skipgram model. The algorithm stores two embeddings for each word, the target embedding (sometimes called the input embedding) and the context embedding (sometimes called the output embedding). The parameter $\theta$ that the algorithm learns is thus a matrix of $2|V|$ vectors, each of dimension $d$, formed by concatenating two matrices, the target embeddings **W** and the context+noise embeddings **C**.

Figure from Jurafsky and Martin

# Train: Gradient Descent, one Word at a Time

- Parameters $\theta$: vectors $v_w$ and $u_w$ for all words in $|V|$
- Optimize training objective via gradient descent (with learning rate $\alpha$)

$$\theta^{new} = \theta^{old} - \alpha \nabla_\theta J(\theta)$$

- Make updates one at a time: each update is for a single pair of center word and a context word
- Loss function

$$J(\theta) = -\frac{1}{T} logL(\theta) = -\frac{1}{T} \sum_{t}^{T} \sum_{\substack{-m \le j \le m \\ j \neq 0}} logP(w_{t+j}|w_t, \theta) = \frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \neq 0}} J_{t,j}(\theta)$$

- For the center word $w_t$, the loss contains a distinct term $J_{t,j}(\theta) = -logP(w_{t+j}|w_t, \theta)$ for each of its context words $w_{t+j}$

# Example: Gradient Update, one Word at a Time

... I saw a cute grey cat playing in the garden ...

- Loss for the central word *cat* and the context word *cute*:

$$J_{t,j}(\theta) = -\log P(cute|cat) = -\log \frac{\exp u_{cute}^T v_{cat}}{\sum_{w \in Voc} \exp u_w^T v_{cat}} = -u_{cute}^T v_{cat} + \log \sum_{w \in Voc} \exp u_w^T v_{cat}.$$

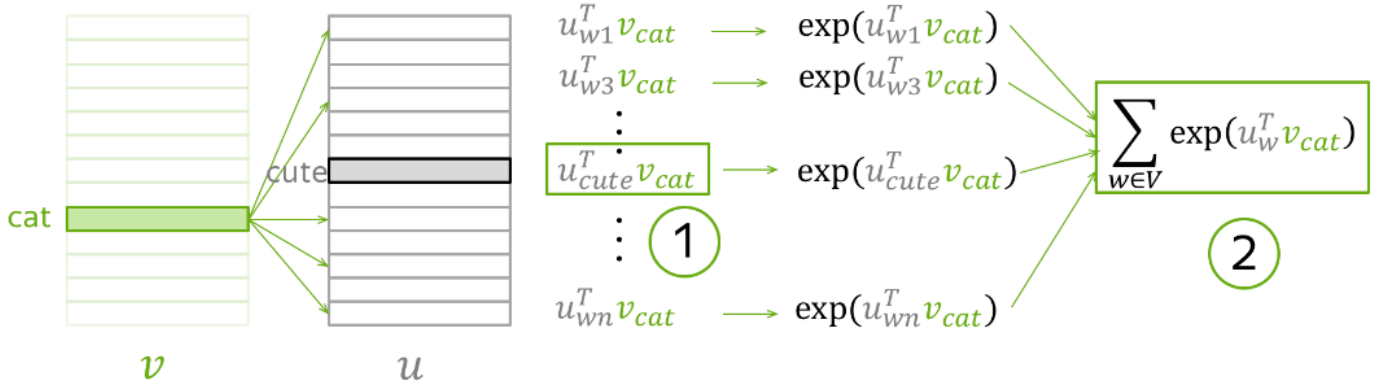
- Parameters at this step:
  - vectors for central words: only $w_{cat}$
  - vectors context words: all $u_w$

$\Rightarrow$ These parameters will be updated

Figure by Lena Voita - NLP Course for You

# Example: Gradient Update

1. Take dot product of $v_{cat}$ with **all** $u$ 2. exp 3. sum all




4. get loss (for this one step)

$$J_{t,j}(\theta) = -\underbrace{u_{cute}^T v_{cat}}_{1} + \underbrace{\log \sum_{w \in V} \exp(u_w^T v_{cat})}_{2}$$

5. evaluate the gradient, make an update

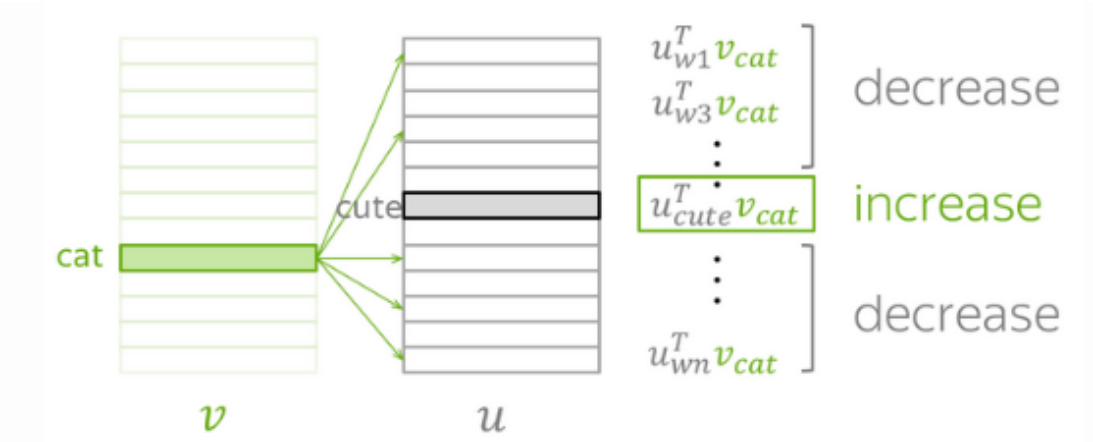
$$v_{cat} := v_{cat} - \alpha \frac{\partial J_{t,j}(\theta)}{\partial v_{cat}}$$

$$u_w := u_w - \alpha \frac{\partial J_{t,j}(\theta)}{\partial u_w} \quad \forall w \in V$$

Figure by Lena Voita - NLP Course for You

# Example: Gradient Update

- Making an update to minimize $J_{t,j}(\theta)$
  - → increase the similarity (i.e. dot product) of $v_{cat}$ and $u_{cute}$
  - → decrease the similarity between $v_{cat}$ and $u_w$ for all other words $w$




- Why decrease similarity between $v_{cat}$ and all other words?
  some of them might be valid contexts as well
- Updates for all context words for all target words
  → averaged over all updates, the vectors will learn the distribution

Figure by Lena Voita - NLP Course for You

# Faster Training: Negative Sampling

- For each pair of central word and its context word: update all vectors for context words
- Highly inefficient: each step is proportional to the vocabulary size

- Don't consider all context vectors in V, but randomly sample just a few negative examples

```
... lemon,  a [tablespoon of apricot jam,      a] pinch ...
               c1         c2    w     c3        c4
```

**positive examples +**

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

**negative examples -**

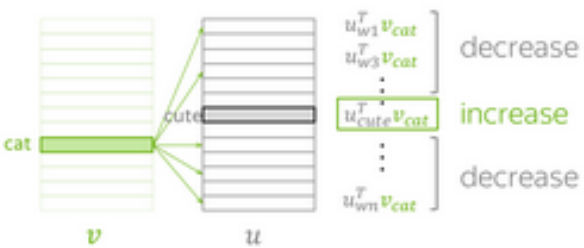
| $w$ | $c_{neg}$ | $w$ | $c_{neg}$ |
|---|---|---|---|
| apricot | aardvark | apricot | seven |
| apricot | my | apricot | forever |
| apricot | where | apricot | dear |
| apricot | coaxial | apricot | if |

- Word2Vec uses more negative than positive examples

Figure from Jurafsky and Martin

# Faster Training: Negative Sampling

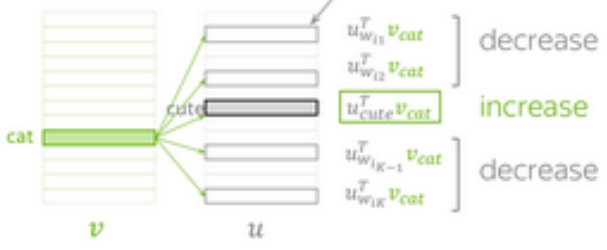
Dot product of $v_{cat}$:
- with $u_{cute}$ - increase,
- with all other $u$ - decrease




Parameters to be updated:
- $v_{cat}$
- $u_w$ for all $w$ in the vocabulary — |V| + 1 vectors

Dot product of $v_{cat}$:
- with $u_{cute}$ - increase,
- with a subset of other $u$ - decrease

Negative samples: randomly selected K words




Parameters to be updated:
- $v_{cat}$
- $u_{cute}$ and $u_w$ for $w$ in K negative examples — K + 2 vectors

- Increase similarity between $u_{cat}$ and $u_{cute}$
- Decrease similarity between $u_{cat}$ and subset of $k$ negative examples
- Large corpus: on average, all updates will update each vector sufficient number of times

# Faster Training: Negative Sampling

- The new loss function:

$$J_{t,j}(\theta) = -\log \sigma(u_{cute}^T v_{cat}) - \sum_{w \in \{w_{i_1}, \ldots, w_{i_K}\}} \log \sigma(-u_w^T v_{cat})$$

- $w_1, \ldots, w_k$ are the $k$ negative examples

- Sigmoid function to map the dot product into a probability:

$$\sigma(x) = \frac{1}{1+exp^{-x}} \qquad \text{Note: } \sigma(-x) = 1 - \sigma(x)$$

- Write the loss also as

$$J_{t,j}(\theta) = -\log \sigma(u_{cute}^T v_{cat}) - \sum_{w \in \{w_{i_1}, \ldots, w_{i_K}\}} \log(1 - \sigma(u_w^T v_{cat})).$$

Figure by Lena Voita - NLP Course for You

# Word2Vec: Word and Context Vectors

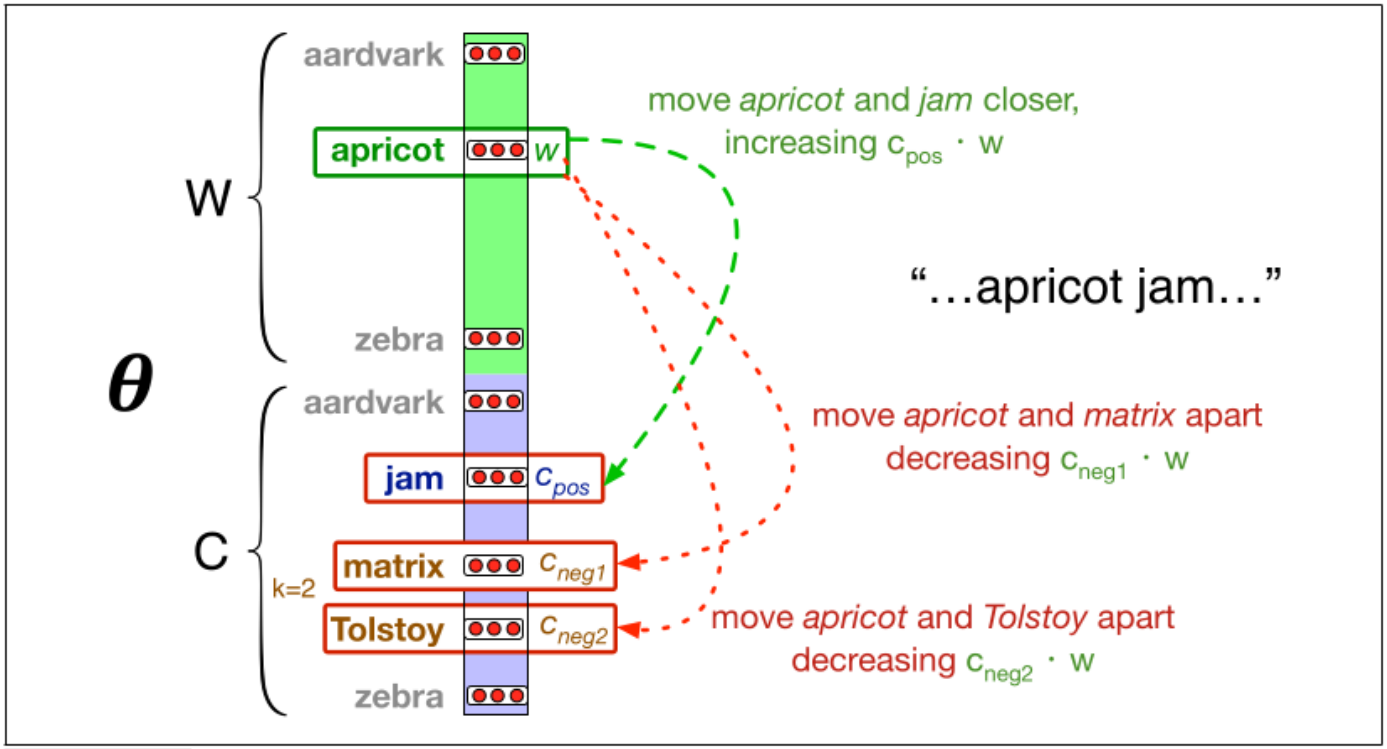



**Figure 6.14** Intuition of one step of gradient descent. The skip-gram model tries to shift embeddings so the target embeddings (here for *apricot*) are closer to (have a higher dot product with) context embeddings for nearby words (here *jam*) and further from (lower dot product with) context embeddings for noise words that don't occur nearby (here *Tolstoy* and *matrix*).

Figure from Jurafsky and Martin

# Choice of Negative Examples

- Each word has only few "true" contexts: randomly chosen words are likely "negative" (= not true) contexts
- Noise words are chosen according to their empirical distribution
- Modified to sample less frequent words more often:
  weighted unigram frequency $p_\alpha(w)$

  sample *the* with probability $p_\alpha(the)$
  sample *aardvark* with probability $p_\alpha(aardvark)$

- In practice, set $\alpha = 0.75$ and use $P_\alpha(w) = \frac{count(w)^\alpha}{\sum_{w'} count(w')^\alpha}$
- We did the same for PPI

# Word2Vec: Variants

- Skip-Gram: predicts context words given the central word.
  Skip-Gram with negative sampling is the most popular approach
- CBOW (Continuous Bag-of-Words): predicts the central word from the sum of context vectors (= bag of words)

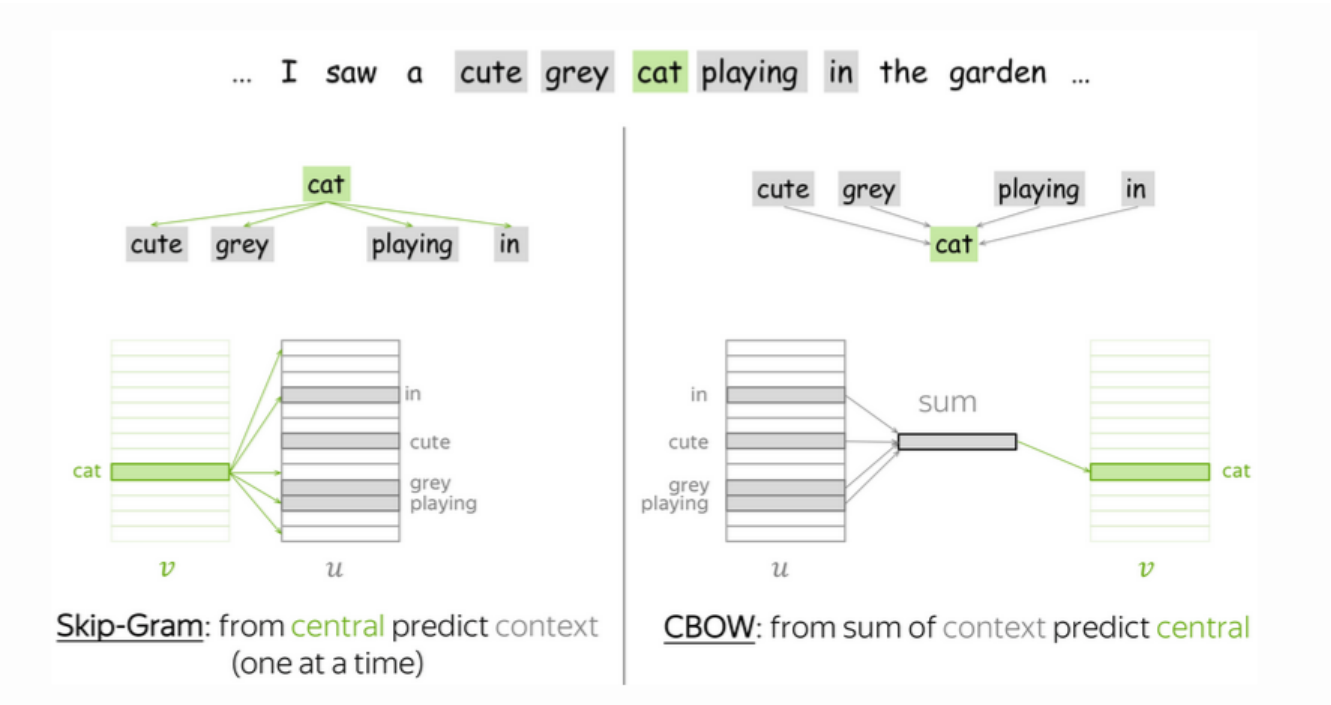



Figure by Lena Voita - NLP Course for You

# Outline

# Fasttext

- Word2Vec: no good way to handle unknown words and sparsity
  - unknown: occurs in test data, but unseen in training data
  - sparsity (for example in morphologically rich languages): some word forms only rarely seen in training data → unreliable representation

- Fasttext Bojanowski et al. (2017)
- Enriches word vectors with subword information
- Subword models: represent each word as itself and a bag of constituent n-grams
- For example *word* with $n = 3$: `<word>` and `<wo, wor, ord, rd>`

- Learn a skipgram embedding for each constituent n-gram
- Unknown words: represented by the sum of the constituent n-grams

# GloVe

- GloVe: Gobal Vectors Pennington et al. (2014)
- Based on global corpus statistics: count-based methods to measure association between word $w$ and context $c$
- GloVe controls the influence of rare and frequent words
- Loss function is designed such that
  - rare events are penalized
  - very frequent events are not over-weighted

# Outline

# Visualizing Embeddings

- Visualizing embeddings is important in order to understand and apply such models
- Simplest way to visualize the meaning of a word: list the most similar words according to cosine similarity

0. *frog*
1. frogs
2. toad
3. litoria
4. leptodactylidae
5. rana
6. lizard
7. eleutherodactylus


3. litoria


4. leptodactylidae


5. rana

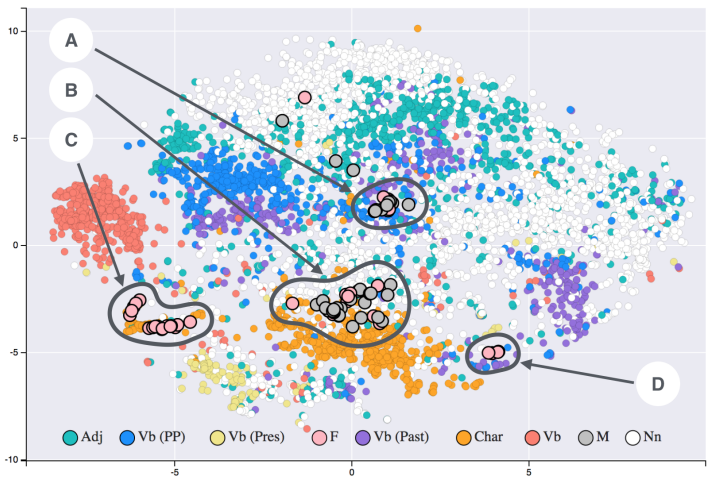

7. eleutherodactylus

- Sometimes, the nearest neighbors according to this metric reveal rare but relevant words that lie outside an average human's vocabulary

Figure from https://nlp.stanford.edu/projects/glove/

# Visualizing Embeddings

- Clustering: show a hierarchical representation of which words are similar to others
- Probably most common method: project $d$ dimensions of a word to two dimensions, for example by means of t-SNE (t-Distributed Stochastic Neighbor Embedding)




T-SNE visualisation of word embeddings generated using 19th century literature

Figure by Siobhán Grayson, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=64541584
https://en.wikipedia.org/wiki/T-distributed_stochastic_neighbor_embedding

# Types of Similarity or Association

- Size of context window: generally between 1 and 10 words on each side of the target word
- Shorter context windows: capture functional and syntactic similarities

  information comes from nearby words → most similar words tend to be semantically similar with the same part-of-speech: *Poodle*, *Pitbull*, *Rottweiler*
- Longer context window: topically related words: *dog*, *bark*, *leash*

# Analogy/Relational Similarity

- Models like word2vec or GloVe can solve analogy problems
  $\overrightarrow{king} - \overrightarrow{man} + \overrightarrow{woman}$ is vector close to $\overrightarrow{queen}$
  $\overrightarrow{Paris} - \overrightarrow{France} + \overrightarrow{Italy}$ is vector close to $\overrightarrow{Rome}$

  The embedding model thus seems to be extracting **representations of relations** like *male – female*, or *capital – city-of*, or even *comparative – superlative*

- But ... it doesn't always work
  - just returning morpological variants:
    $\overrightarrow{cherry} - \overrightarrow{red} + \overrightarrow{potato}$ returns $\overrightarrow{potatoes}$ instead of $\overrightarrow{brown}$
  - embedding spaces perform well if the task involves **frequent words**, **small distances**, and **certain relations** (relating countries/capitals or verbs/nouns with inflected forms), but not so well for other relations

# Analogy/Relational Similarity

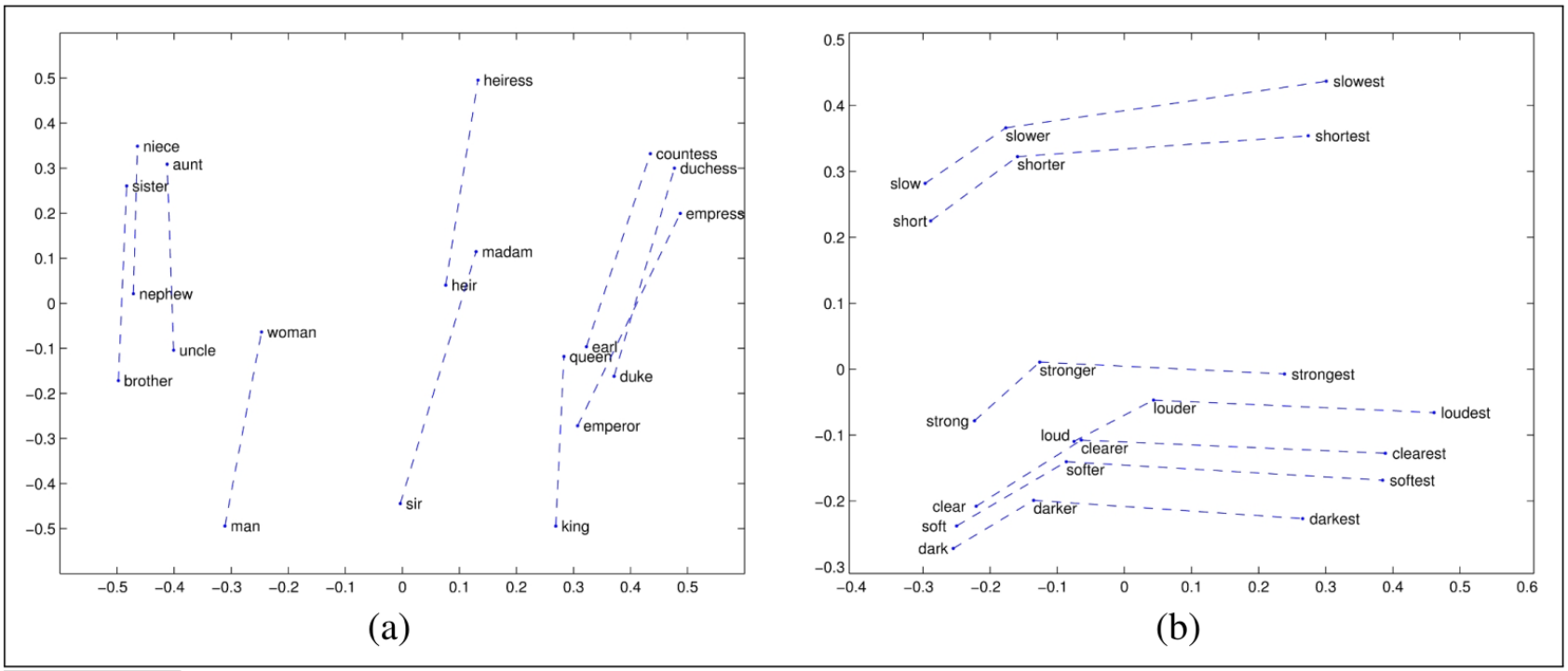



**Figure 6.16** Relational properties of the GloVe vector space, shown by projecting vectors onto two dimensions. (a) $\overrightarrow{\text{king}} - \overrightarrow{\text{man}} + \overrightarrow{\text{woman}}$ is close to $\overrightarrow{\text{queen}}$. (b) offsets seem to capture comparative and superlative morphology (Pennington et al., 2014).

Figure from Jurafsky and Martin

# Embeddings and Historical Semantics

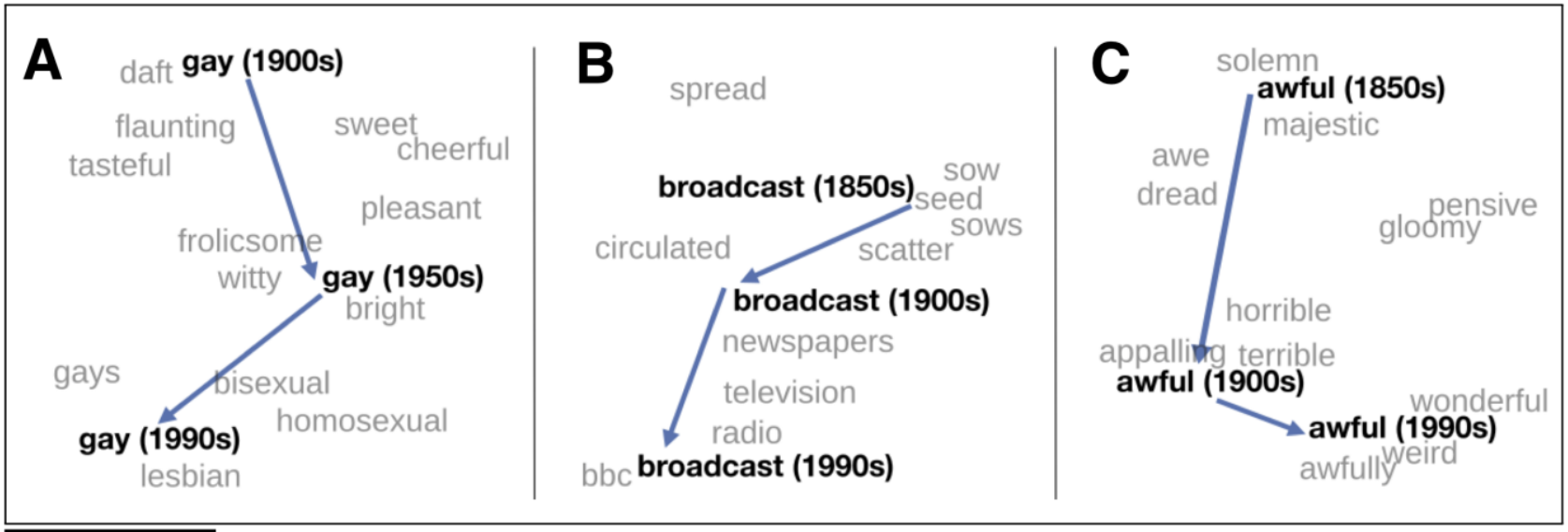



**Figure 6.17** A t-SNE visualization of the semantic change of 3 words in English using word2vec vectors. The modern sense of each word, and the grey context words, are computed from the most recent (modern) time-point embedding space. Earlier points are computed from earlier historical embedding spaces. The visualizations show the changes in the word *gay* from meanings related to "cheerful" or "frolicsome" to referring to homosexuality, the development of the modern "transmission" sense of *broadcast* from its original sense of sowing seeds, and the pejoration of the word *awful* as it shifted from meaning "full of awe" to meaning "terrible or appalling" (Hamilton et al., 2016b).

Figure from Jurafsky and Martin

# Outline

# Bias and Embeddings

- Embeddings reproduce the implicit biases and stereotypes that were latent in the training data
- For example, gender stereotypes :

  *computer programmer - man + woman → homemaker*

  *father:doctor::mother:nurse*

- But also racism, ...
- Bias amplification: gendered terms become *more* gendered in embedding space
- Debiasing: transform the embedding space such that gender stereotypes are removed, but definitional gender is preserved.

  Still an open problem ...

# Outline

# Credits

- This lecture is mostly based on and contains content from
  - Chapter 6 in *Speech and Language Processing*, Jurafsky and Martin (2024)
  - Lena Voita's *NLP Course for You*

    `https://lena-voita.github.io/nlp_course/word_embeddings.html#w2v_idea`

- Some slides took inspiration from a presentation of Marco Baroni and Gemma Boleda

# References


- McDonald, S. and Ramscar, M. (2001): *Testing the distributional hypothesis: The influence of context on judgements of semantic similarity.* Proceedings of the cognitive science society (2001), 611–617.
- Firth, J. R. (1957). *Papers in Linguistics 1934-1951.* Oxford University Press.
- Mikolov, T., K. Chen, G. S. Corrado, and J. Dean (2013a): *Efficient estimation of word representations in vector space.* Proceedings of ICLR 2013
- Mikolov, T., I. Sutskever, K. Chen, G. S. Corrado, and J. Dean (2013b): *Distributed representations of words and phrases and their compositionality.* Proceedings of NeurIPS.
- Jeffrey Pennington, Richard Socher, and Christopher D. Manning (2014): *GloVe: Global Vectors for Word Representation.* Proceedings of EMNLP.
- Bojanowski, P., E. Grave, A. Joulin, and T. Mikolov (2017): *Enriching word vectors with subword information.* Proceedings of TACL, 5:135–146.