ORTA DOĞU TEKNİK ÜNİVERSİTESİ
MIDDLE EAST TECHNICAL UNIVERSITY

CENG 463: Introduction to Natural Language Processing
Word Vectors: Word Embeddings

Asst. Prof. Cagri Toraman
Computer Engineering Department
ctoraman@ceng.metu.edu.tr

04.11.2025

Text Semantics

Sparse vs. dense vectors

tf-idf (or PMI) vectors

- **long** (length |V|= 10,000 to 50,000)
- **sparse** (most elements are zero)

Alternative: learn vectors

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

Text Semantics

Why dense vectors?

Short vectors may be easier to use as **features** in deep learning (fewer weights to tune)
Dense vectors may **generalize** better than explicit counts
Dense vectors may do better at capturing synonymy:
car and *automobile* are synonyms; but are distinct dimensions
In practice, they work better

Text Semantics

Common methods for getting short dense vectors

Static embeddings

- Word2vec (skipgram, CBOW), GloVe

Singular Value Decomposition (SVD)

- A special case of this is called LSA – Latent Semantic Analysis

Dynamic embeddings:

- Contextual Embeddings (ELMo, BERT)
- Compute distinct embeddings for a word in its context
- Separate embeddings for each token of a word

Text Semantics

Static embeddings that you can download (no training needed)

Word2vec (Google, 2013)

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

GloVe (Stanford, 2014)

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

fastText (Facebook/Meta, 2015)

https://fasttext.cc

Word2Vec: skip-gram with negative sampling

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

- But we'll take the learned classifier weights as the word embeddings

Word2Vec: skip-gram with negative sampling

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

Word2Vec: skip-gram with negative sampling

(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)
(apricot, aardvark)

And assigns each pair a probability:
$P(+|w, c)$
$P(-|w, c) = 1 - P(+|w, c)$

$\text{Sim}(w,c) \approx w \cdot c$

To turn this into a probability, use the sigmoid:

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

Word2Vec: skip-gram with negative sampling

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

Assume independence and just multiply them:

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

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

Word2Vec: skip-gram with negative sampling

Skip-gram classifier: summary

A probabilistic classifier, given
a test target word *w*
and 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.

Word2Vec: skip-gram with negative sampling

Skip-Gram Training data

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

c1 c2 [target] c3 c4

positive examples +

t	c
apricot	tablespoon
apricot	of
apricot	jam
apricot	a

negative examples -

t	c	t	c
apricot	aardvark	apricot	seven
apricot	my	apricot	forever
apricot	where	apricot	dear
apricot	coaxial	apricot	if

Word2vec: How to Learn Vectors

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

The goal of learning is to adjust those word vectors such that:

Maximize the similarity of the target word, context word pairs (w , c_{pos}) drawn from the positive data

Minimize the similarity of the (w , c_{neg}) pairs drawn from the negative data.

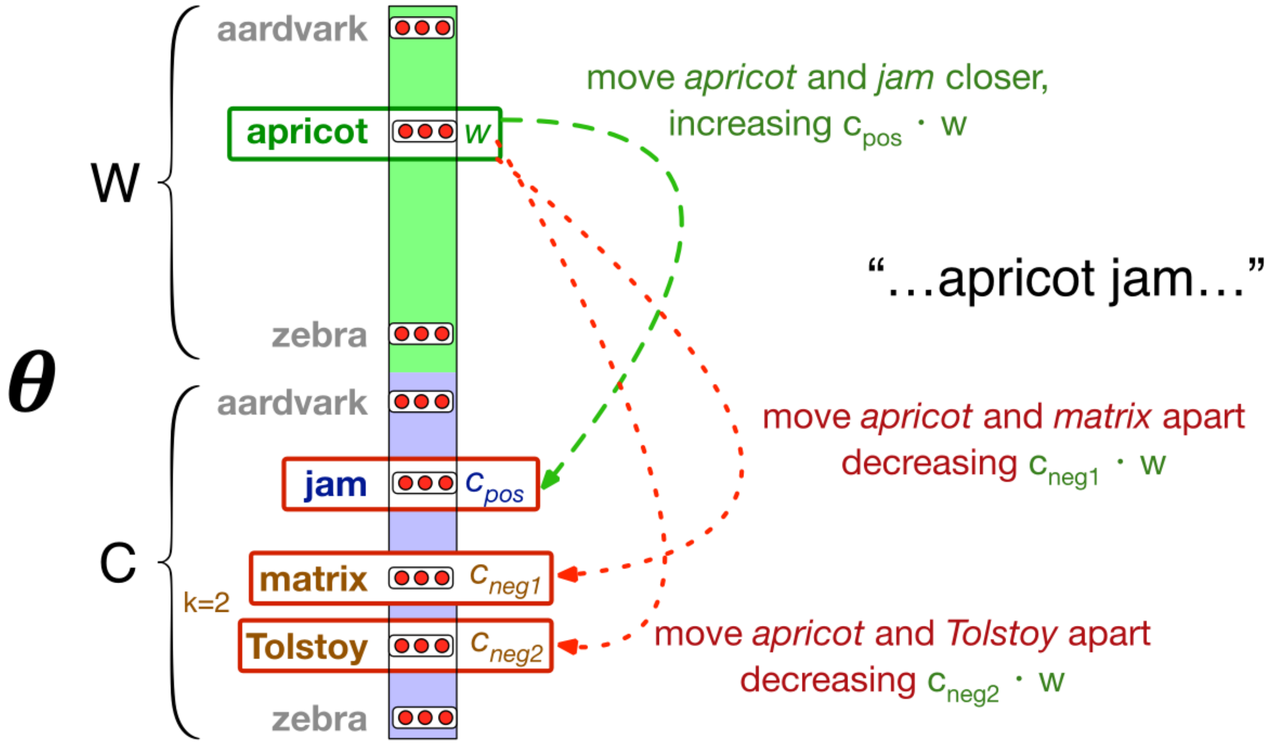
Word2vec: How to Learn Vectors

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.

Use Stochastic Gradient Descent!

$$
\begin{aligned}
L_{CE} &= -\log\left[P(+|w,c_{pos})\prod_{i=1}^{k}P(-|w,c_{neg_i})\right] \\
&= -\left[\log P(+|w,c_{pos}) + \sum_{i=1}^{k}\log P(-|w,c_{neg_i})\right] \\
&= -\left[\log P(+|w,c_{pos}) + \sum_{i=1}^{k}\log\left(1-P(+|w,c_{neg_i})\right)\right] \\
&= -\left[\log \sigma(c_{pos}\cdot w) + \sum_{i=1}^{k}\log\sigma(-c_{neg_i}\cdot w)\right]
\end{aligned}
$$

Intuition of one step of gradient descent



Reminder

At each step:

We move in the reverse direction from the gradient of the loss function.

We move the value of this gradient

The derivatives of the loss function

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

$$\frac{\partial L_{CE}}{\partial c_{pos}} = [\sigma(c_{pos} \cdot w) - 1]w$$

$$\frac{\partial L_{CE}}{\partial c_{neg}} = [\sigma(c_{neg} \cdot w)]w$$

$$\frac{\partial L_{CE}}{\partial w} = [\sigma(c_{pos} \cdot w) - 1]c_{pos} + \sum_{i=1}^{k} [\sigma(c_{\mathrm{neg}_i} \cdot w)]c_{neg_i}$$

Start with randomly initialized C and W matrices, then incrementally do updates

$$
\begin{aligned}
c_{pos}^{t+1} &= c_{pos}^{t} - \eta[\sigma(c_{pos}^{t} \cdot w^{t}) - 1]w^{t} \\
c_{neg}^{t+1} &= c_{neg}^{t} - \eta[\sigma(c_{neg}^{t} \cdot w^{t})]w^{t} \\
w^{t+1} &= w^{t} - \eta\left[[\sigma(c_{pos} \cdot w^{t}) - 1]c_{pos} + \sum_{i=1}^{k}[\sigma(c_{\mathrm{neg}_i} \cdot w^{t})]c_{neg_i}\right]
\end{aligned}
$$

Learns two sets of embeddings:

- Target embeddings matrix W
- Context embedding matrix C

Add them together, representing word *i* as the vector $w_i + c_i$

GloVe: Global Vector Embeddings

Words with similar distributions have similar meanings*

X_{ij} tabulate the number of times word j occurs in the context of word i

$X_i = \sum_k X_{ik}$

$P_{ij} = P(j|i) = X_{ij}/X_i$

$w \in \mathbb{R}^d$ are word vectors

probe word

$$F(w_i, w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}$$

co-relations between the word w*i* and w*j*

co-occurrence probabilities for the word w*j* and w*k*

*Pennington, J., Socher, R., & Manning, C. D. (2014, October). Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).

GloVe: Global Vector Embeddings

Words with similar distributions have similar meanings*

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

Very small or large:

solid is related to ice but not steam, or
gas is related to steam but not ice

close to 1:

water is highly related to ice and steam, or
fashion is not related to ice or steam.

*Pennington, J., Socher, R., & Manning, C. D. (2014, October). Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).

GloVe: Global Vector Embeddings

Words with similar distributions have similar meanings*

$w \in \mathbb{R}^d$ are word vectors

probe word

$$F(w_i, w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}$$

co-relations between the word w*i* and w*j*

co-occurrence probabilities for the word w*j* and w*k*

$w_i{}^T \tilde{w}_k$ relate to (high probability if they are similar)

$$F\left((w_i - w_j)^T \tilde{w}_k\right) = \frac{P_{ik}}{P_{jk}}$$

$w_j{}^T \tilde{w}_k$

*Pennington, J., Socher, R., & Manning, C. D. (2014, October). Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).

GloVe: Global Vector Embeddings

Cost (objective) function:

Mean Square Error to calculate the error in the predicted co-occurrence counts and the ground truth.

A weight to readjust the cost for each word pair

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2$$

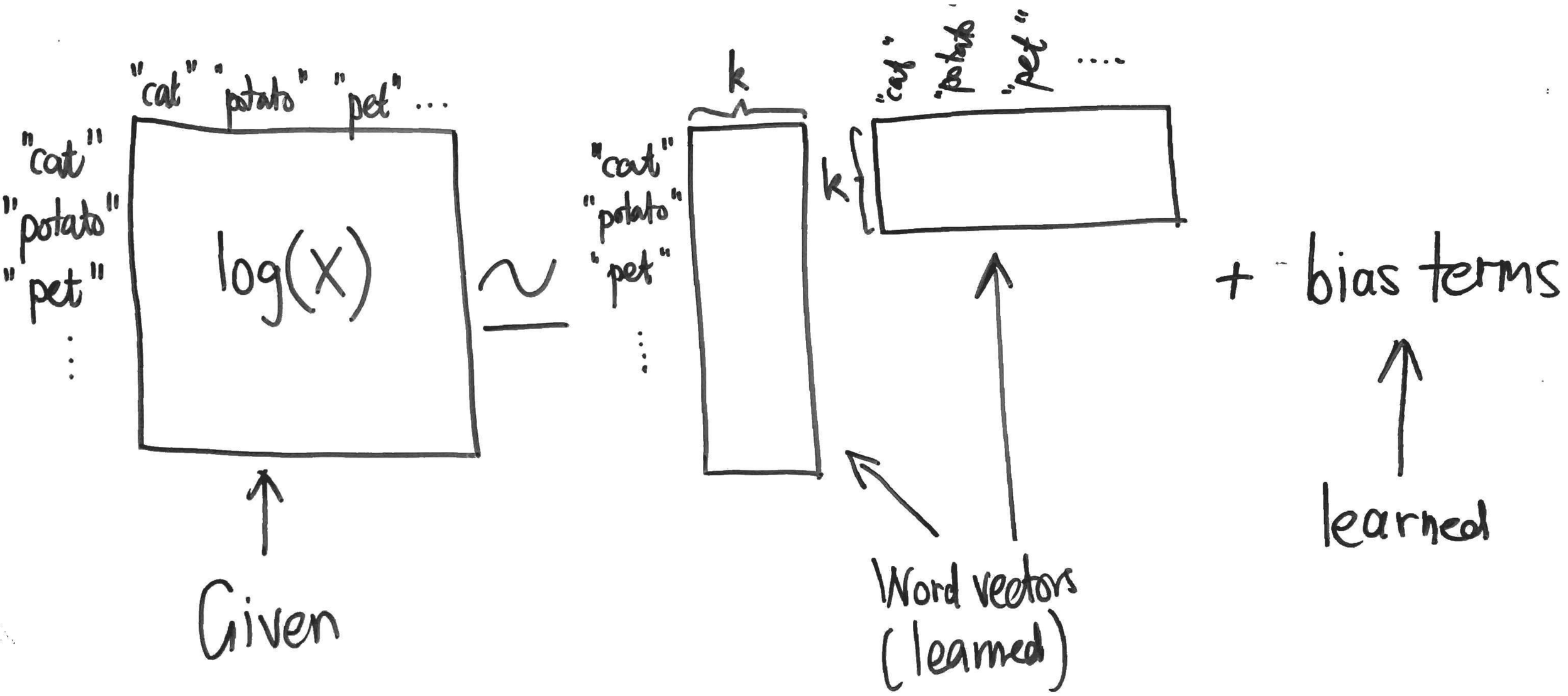
100 3/4

$$f(x) = \begin{cases} (x/x_{\max})^{\alpha} & \text{if } x < x_{\max} \\ 1 & \text{otherwise} \end{cases}$$

Learn word vectors such that their dot product equals the logarithm of the words' probability of co-occurrence

**Pennington, J., Socher, R., & Manning, C. D. (2014, October). Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).*

GloVe: Global Vector Embeddings



GloVe: Global Vector Embeddings

Example Calculation

Using the example co-occurrence matrix, let's calculate for the pair **("i", "love")** with a co-occurrence count $X_{ij} = 2$.

1. **Co-occurrence Value**: $\log(X_{ij}) = \log(2) = 0.693$.
2. **Dot Product**: Assume we initialize the word vectors $\mathbf{w}_i = [w_{i1}, w_{i2}]$ and $\mathbf{w}_j = [w_{j1}, w_{j2}]$, each with 2 dimensions.
   The dot product:

$$\mathbf{w}_i \cdot \mathbf{w}_j = w_{i1} \cdot w_{j1} + w_{i2} \cdot w_{j2}$$

3. **Error for the Pair**: Substituting into the loss function for this pair:

$$\text{Loss} = f(X_{ij}) \cdot (\mathbf{w}_i \cdot \mathbf{w}_j + b_i + b_j - \log(X_{ij}))^2$$

GloVe: Global Vector Embeddings

4. **Weighting Function $f(X_{ij})$**: $f(X_{ij})$ ensures the algorithm focuses on meaningful co-occurrence pairs. For example:

$$f(X_{ij}) = \min\left(\left(\frac{X_{ij}}{X_{\max}}\right)^{\alpha}, 1\right)$$

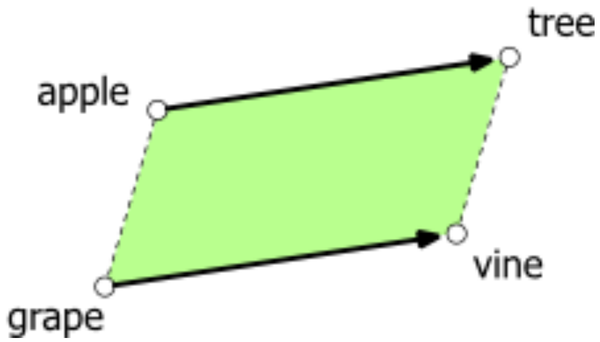
Where $X_{\max}$ and α are hyperparameters. For simplicity, let's assume $f(X_{ij}) = 1$.

5. **Substitute Values**: For word pair ("i", "love"):
   - Assume initial vectors $\mathbf{w}_i = [0.5, 0.8]$ and $\mathbf{w}_j = [0.6, 0.7]$.
   - Dot product:
   $$\mathbf{w}_i \cdot \mathbf{w}_j = (0.5 \cdot 0.6) + (0.8 \cdot 0.7) = 0.3 + 0.56 = 0.86$$
   - Bias terms $b_i = 0.1, b_j = 0.2$.
   - Loss for this pair:
   $$\text{Loss} = (0.86 + 0.1 + 0.2 - 0.693)^2 = (1.16 - 0.693)^2 = (0.467)^2 = 0.218$$

Implications of Word Embeddings

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

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



Implications of Word Embeddings

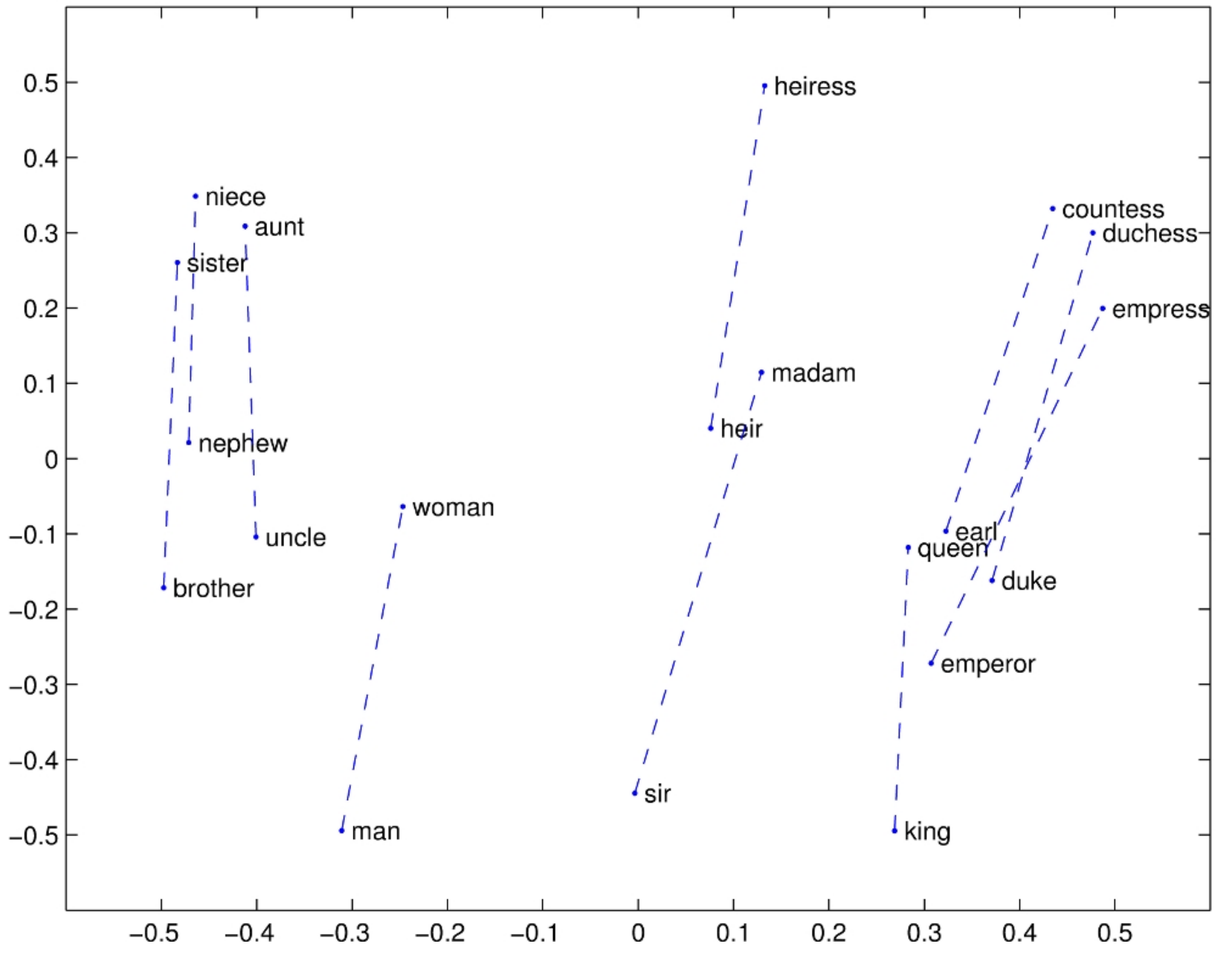
king – man + woman is close to queen

Paris – France + Italy is close to Rome

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

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

Implications of Word Embeddings



Implications of Word Embeddings

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)

Implications of Word Embeddings

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

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

Implications of Word Embeddings

They 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

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.

ORTA DOĞU TEKNİK ÜNİVERSİTESİ
MIDDLE EAST TECHNICAL UNIVERSITY

Thanks for your participation!

Çağrı Toraman
04.11.2025