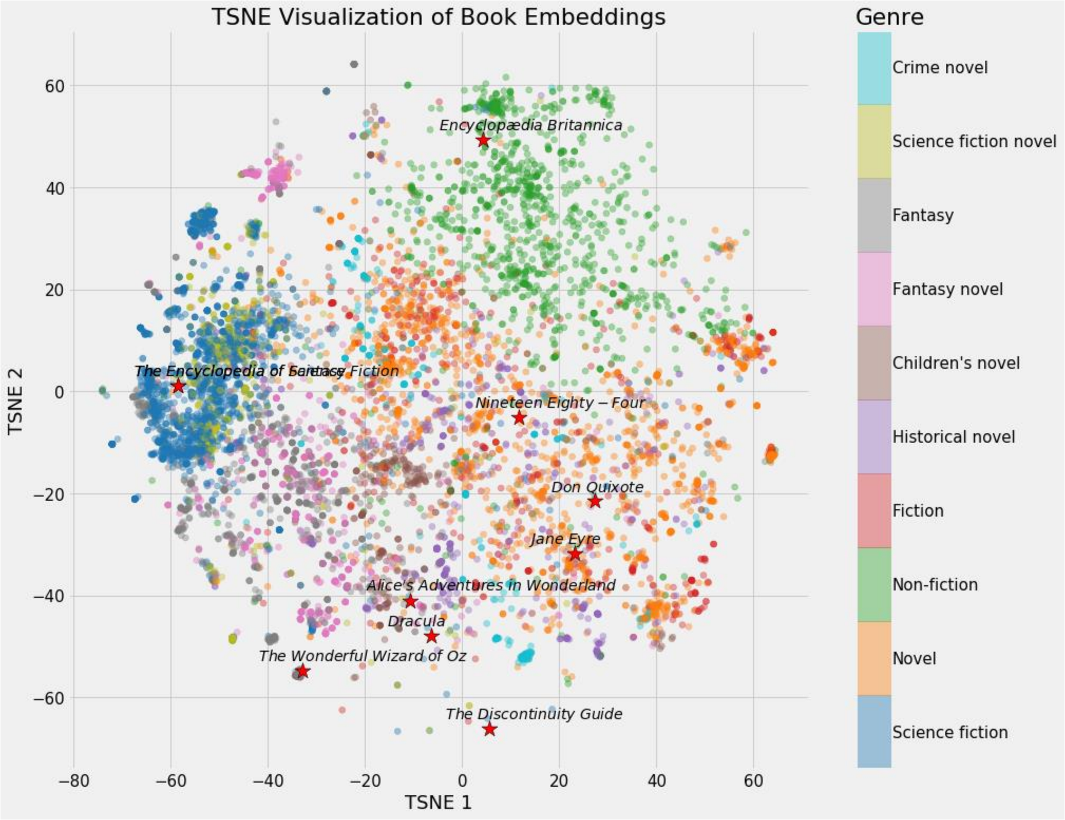
University of Ljubljana, Faculty of Computer and Information Science

# Neural embeddings



Prof Dr Marko Robnik-Šikonja

Natural Language Processing, Edition 2022

# Contents

- Neural language models
- word2vec word representation
- cross-lingual embeddings
- properties of dense embeddings

partially based on Chapter 6.8 to 6.12  in Jurafsky & Martin, 3rd edition,

# Vector Representation of Words

- Vector space models represent (embed) words in a continuous vector space
  - Theoretical foundation in Linguistics: Distributional Hypothesis
    - Words with similar meanings will occur with similar neighbors if enough text material is available (Rubenstein et al. 1967).
- Approaches that leverage embeddings can be divided into two categories

| Approach | Example | Description |
|---|---|---|
| Count-based methods | Latent semantic analysis | Compute how often some word co-occurs with its neighbor words in a large text corpus, and then map these count-statistics down to a small, dense vector for each word |
| Predictive methods | Neural probabilistic language model | Directly predict a word from its neighbors in terms of learned small, dense embedding vectors (considered parameters of the model) |

# Neural embeddings

- neural network is trained to predict the context of words (input: word, output: context of neighboring words)
- Analogy of neural network operations with matrix operations

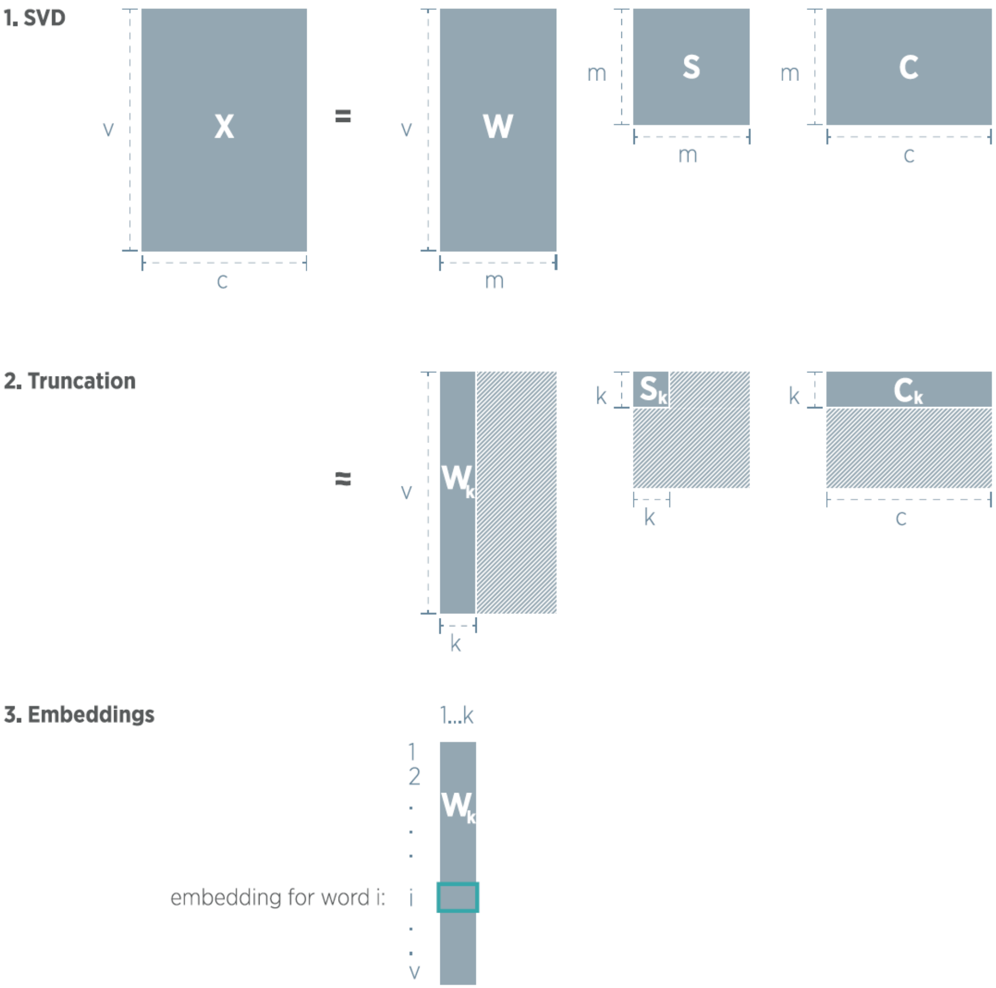
# Word-word matrix (or "term-context matrix")

- Two **words** are similar in meaning if their context vectors are similar.

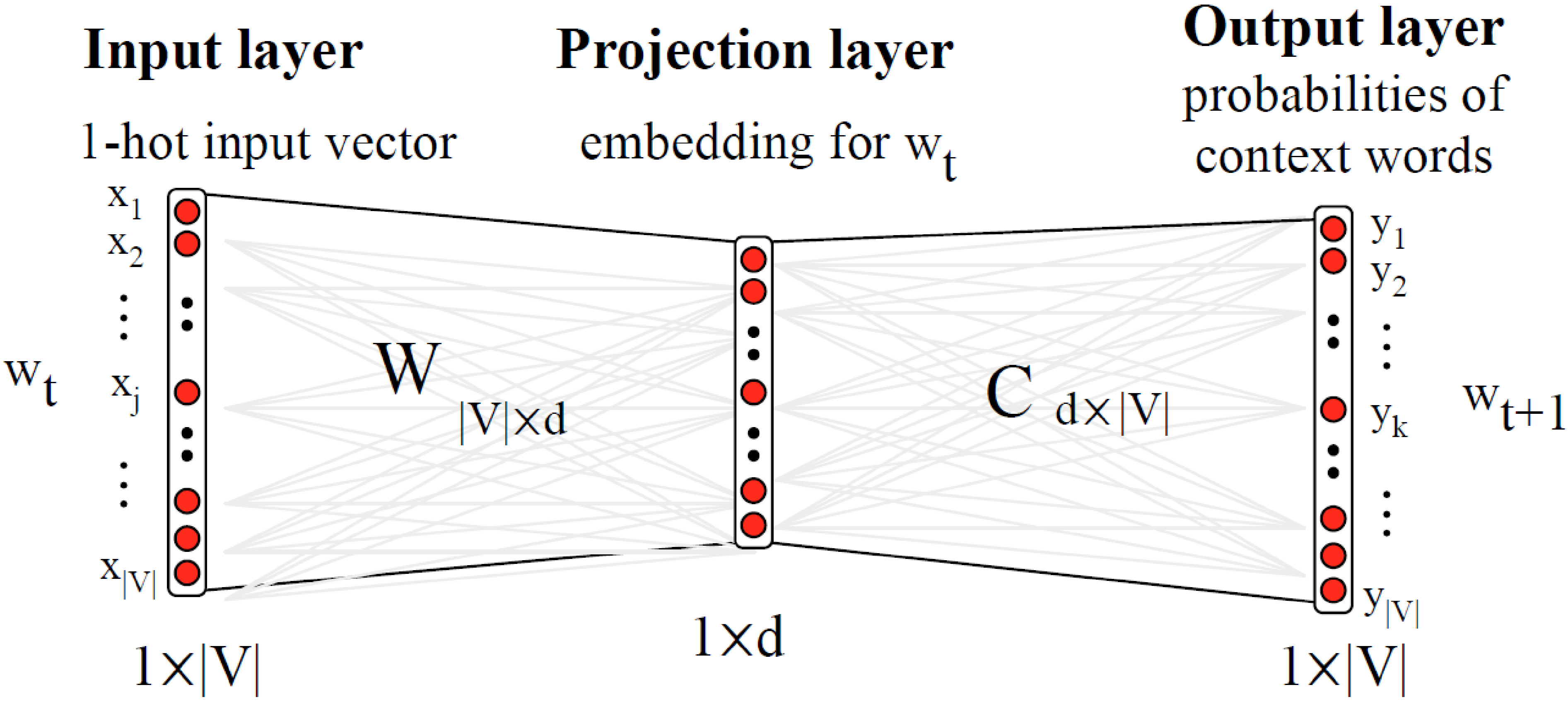
| | | |
|---|---|---|
| sugar, a sliced lemon, a tablespoonful of | **apricot** | jam, a pinch each of, |
| their enjoyment. Cautiously she sampled her first | **pineapple** | and another fruit whose taste she likened |
| well suited to programming on the digital | **computer**. | In finding the optimal R-stage policy from |
| for the purpose of gathering data and | **information** | necessary for the study authorized in the |

| | aardvark | computer | data | pinch | result | sugar | … |
|---|---|---|---|---|---|---|---|
| apricot | 0 | 0 | 0 | 1 | 0 | 1 | |
| pineapple | 0 | 0 | 0 | 1 | 0 | 1 | |
| digital | 0 | 2 | 1 | 0 | 1 | 0 | |
| information | 0 | 1 | 6 | 0 | 4 | 0 | |

# SVD for embeddings



# Simple neural network based embedding



# word2vec method

- 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

- Words near apricot acts as ‘correct answers’ to the question
  “Is word w likely to show up near apricot?”
- No need for hand-labeled supervision

# Main Idea of word2vec

- Instead of capturing co-occurrence counts directly, predict surrounding words of every word

- Faster and can easily incorporate a new sentence/document or add a word to the vocabulary

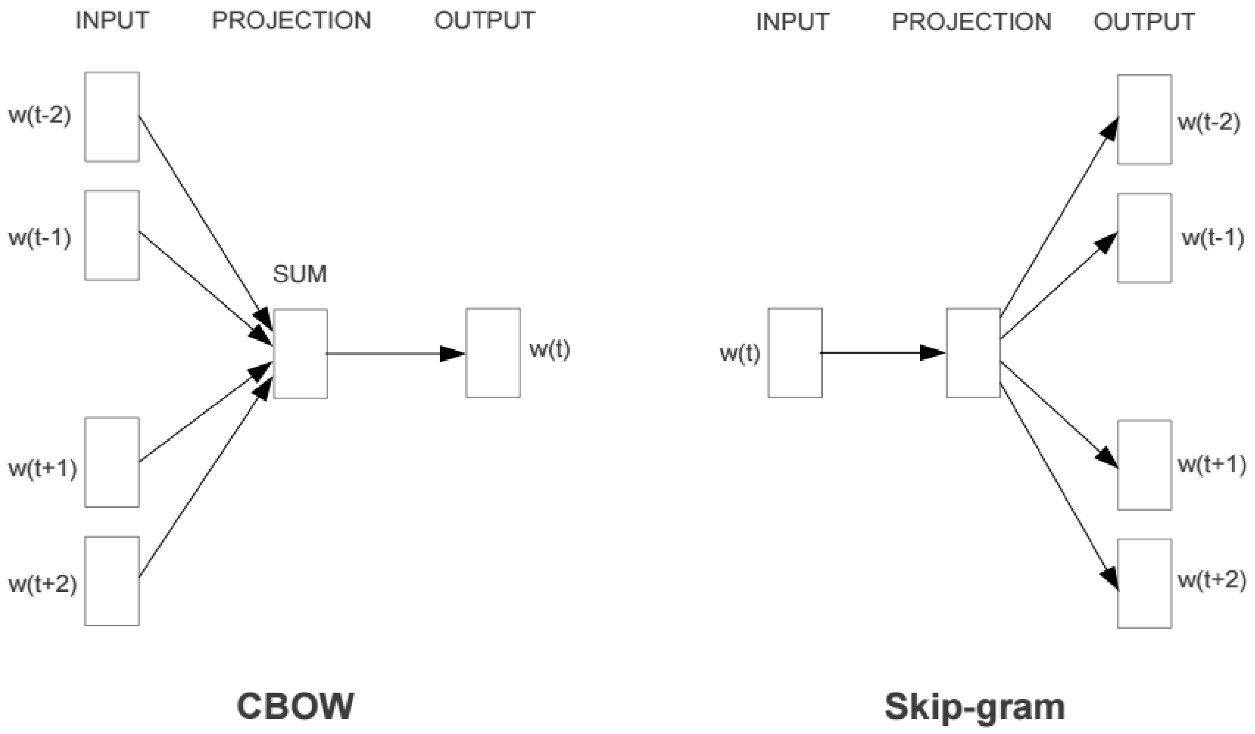
- Two variants:
  - CBOW: Predict target from bag of words context
  - Skipgram: Predict context words from target (position-independent)

- In general, the skipgram variant with negative sampling is somewhat more successful

# Word2vec –Vector Representation of Words (Mikolov et al. 2013)

| Model | Approach | Speed and Performance | Use case |
|---|---|---|---|
| Continuous Bag-of-Words model (CBOW) | The CBOW predicts the current word based on the context. | Faster to train than the skip-gram model | Predicts frequent words better |
| Skip-Gram model | Skip-gram predicts surrounding words given the current word. | Usually performs better than CBOW | Predicts rare words better |

- Word2vec comes with two models:



Note, that this is only a schematic representation, not the actually used neural network architecture.

# Word2vec –Vector Representation of Words (Mikolov et al. 2013)

- Skip-gram learning:
  - Given $w_{0,}$ predict $w_{-2}$, $w_{-1}$, $w_1$, and $w_2$

| $\boldsymbol{w_{-2}}$ | $\boldsymbol{w_{-1}}$ | $\boldsymbol{w_0}$ | $\boldsymbol{w_1}$ | $\boldsymbol{w_2}$ |
|---|---|---|---|---|
| ? | ? | Network | ? | ? |

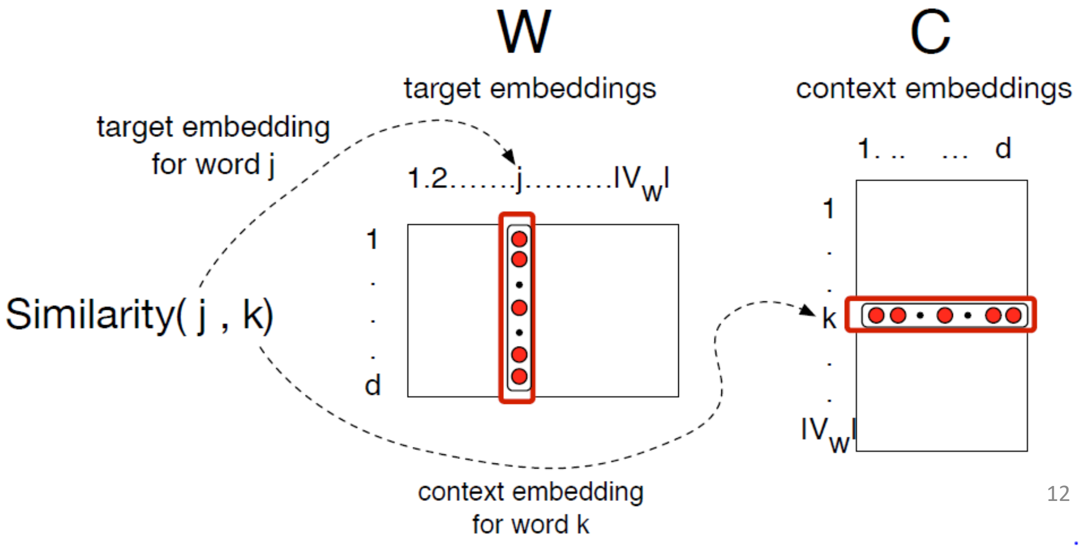
↓

| $\boldsymbol{w_{-2}}$ | $\boldsymbol{w_{-1}}$ | $\boldsymbol{w_0}$ | $\boldsymbol{w_1}$ | $\boldsymbol{w_2}$ |
|---|---|---|---|---|
| Recurrent | Neural | | Language | Model |

- Conversely, CBOW tries to predict $w_0$ when given $w_{-2}$, $w_{-1}$, $w_1$, and $w_2$

# Skip-grams

- Using a given word, we predict the neighborhood of 2L words, L previous and L following ones
- for each word $w_j$ in a dictionary, we estimate the probability that the neighborhood contains word $w_k$, $p(w_k|w_j)$
- estimate the dot product $c_k \cdot v_j$, where $c_k$ is the context vector and $v_j$ the target vector of j-th word



# Skip-gram algorithm

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

# Skip-Gram Training Data

- Training sentence:
- ... lemon, a tablespoon of **apricot** jam a pinch ...
- c1 c2 target c3 c4

Assume context words are those in +/- 2 word window

# Skip-Gram Goal

- Given a tuple (tic) = target, context
  - (*apricot*, *jam*)
  - (*apricot*, *aardvark*)
- Return probability that c is a real context word:
  - P(+|t,c)
  - $P(-|t,c) = 1 - P(+|t,c)$

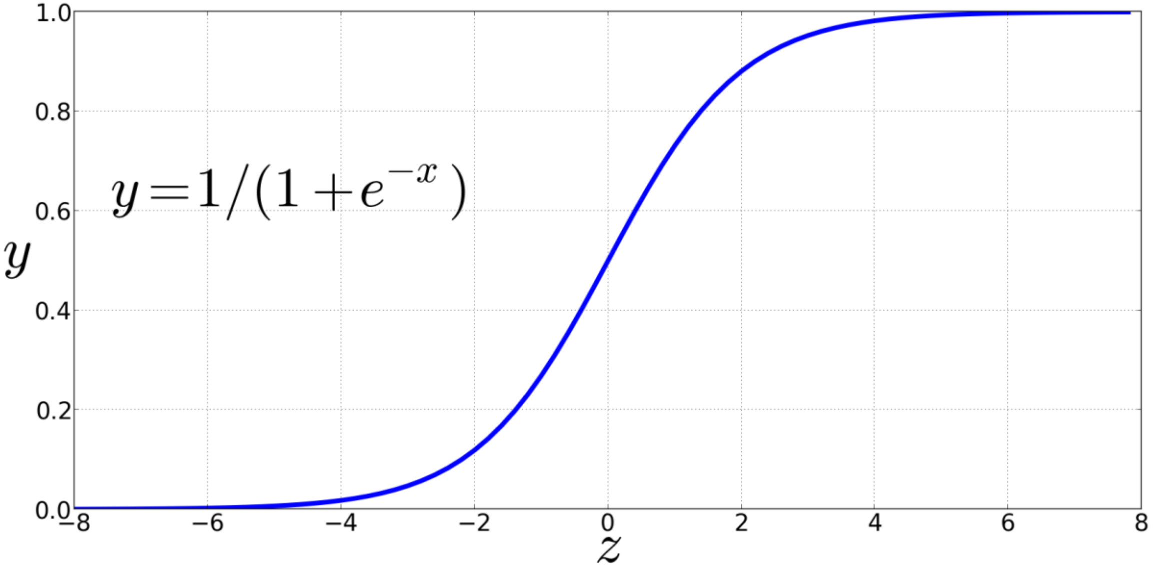
# How to compute p(+|t,c)?

- Intuition:
  - Words are likely to appear near similar words
  - We can model similarity with the dot-product!
  - Similarity(t,c) $\propto t \cdot c$
- *Problem:*
  - *Dot product is not a probability!*
    - *(Neither is cosine)*

# Turning dot product into a probability

- The sigmoid lies between 0 and 1:

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



# Turning dot product into a probability

$$P(+|t,c) = \frac{1}{1+e^{-t\cdot c}}$$

$$\begin{aligned} P(-|t,c) &= 1-P(+|t,c) \\ &= \frac{e^{-t\cdot c}}{1+e^{-t\cdot c}} \end{aligned}$$

# For all the context words:

- Assume all context words are independent

$$P(+|t,c_{1:k}) \;=\; \prod_{i=1}^{\kappa} \frac{1}{1+e^{-t\cdot c_i}}$$

$$\log P(+|t,c_{1:k}) \;=\; \sum_{i=1}^{k} \log \frac{1}{1+e^{-t\cdot c_i}}$$

# Skip-Gram Training Data

- Training sentence:
- ... lemon, a tablespoon of **apricot** jam a pinch ...
- c1 c2 t c3 c4

- Training data: input/output pairs centering on *apricot*
- Assume a +/- 2 word window

# Skip-Gram Training

- Training sentence:
- ... lemon, a tablespoon of apricot jam a pinch ...
- c1 c2 t c3 c4

**positive examples +**

| t | c |
|---|---|
| apricot | tablespoon |
| apricot | of |
| apricot | preserves |
| apricot | or |

- For each positive example, we'll create *k* negative examples.
- Using *noise* words
- Any random word that isn't *t*

# Skip-Gram Training

- Training sentence:
- ... lemon, a tablespoon of apricot jam a pinch ...
- c1 c2 t c3 c4

**positive examples +**

| t | c |
|---|---|
| apricot | tablespoon |
| apricot | of |
| apricot | preserves |
| apricot | or |

**negative examples -** k=2

| t | c | t | c |
|---|---|---|---|
| apricot | aardvark | apricot | twelve |
| apricot | puddle | apricot | hello |
| apricot | where | apricot | dear |
| apricot | coaxial | apricot | forever |

# Choosing noise words

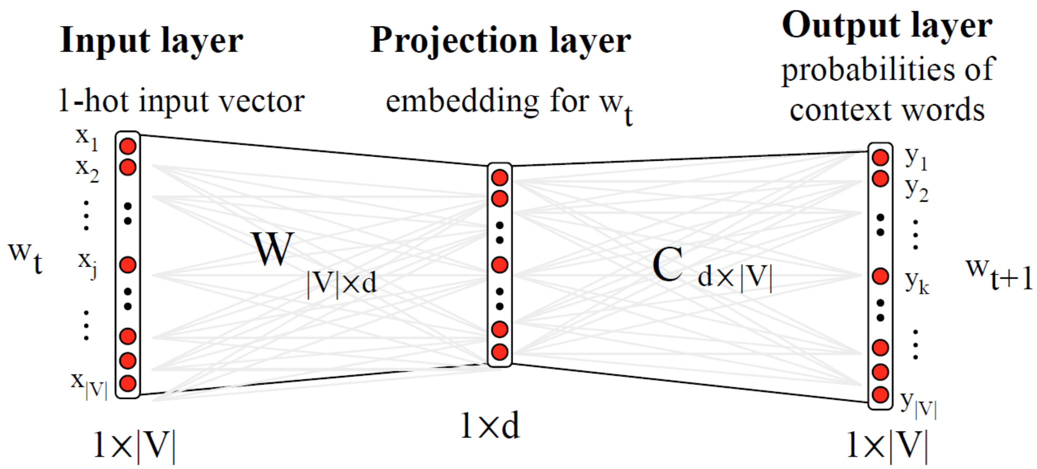
- Could pick w according to their unigram frequency P(w)
- More common to chose according to $p_\alpha(w)$

$$P_\alpha(w) = \frac{count(w)^\alpha}{\sum_w count(w)^\alpha}$$

- α= ¾ works well because it gives rare noise words slightly higher probability
- To show this, imagine two events p(a)=.99 and p(b) = .01:

$$P_\alpha(a) = \frac{.99^{.75}}{.99^{.75} + .01^{.75}} = .97$$
$$P_\alpha(b) = \frac{.01^{.75}}{.99^{.75} + .01^{.75}} = .03$$

# Setup



- Let's represent words as vectors of some length (say 300), randomly initialized.
- So we start with 300 * V random parameters
- Over the entire training set, we'd like to adjust those word vectors such that we
  - Maximize the similarity of the target word, context word pairs (t,c) drawn from the positive data
  - Minimize the similarity of the (t,c) pairs drawn from the negative data.

# Learning the classifier

- Iterative process.
- We’ll start with 0 or random weights
- Then adjust the word weights to
  - make the positive pairs more likely
  - and the negative pairs less likely
- Repeat over the entire training set.

# Objective Criteria

- We want to maximize

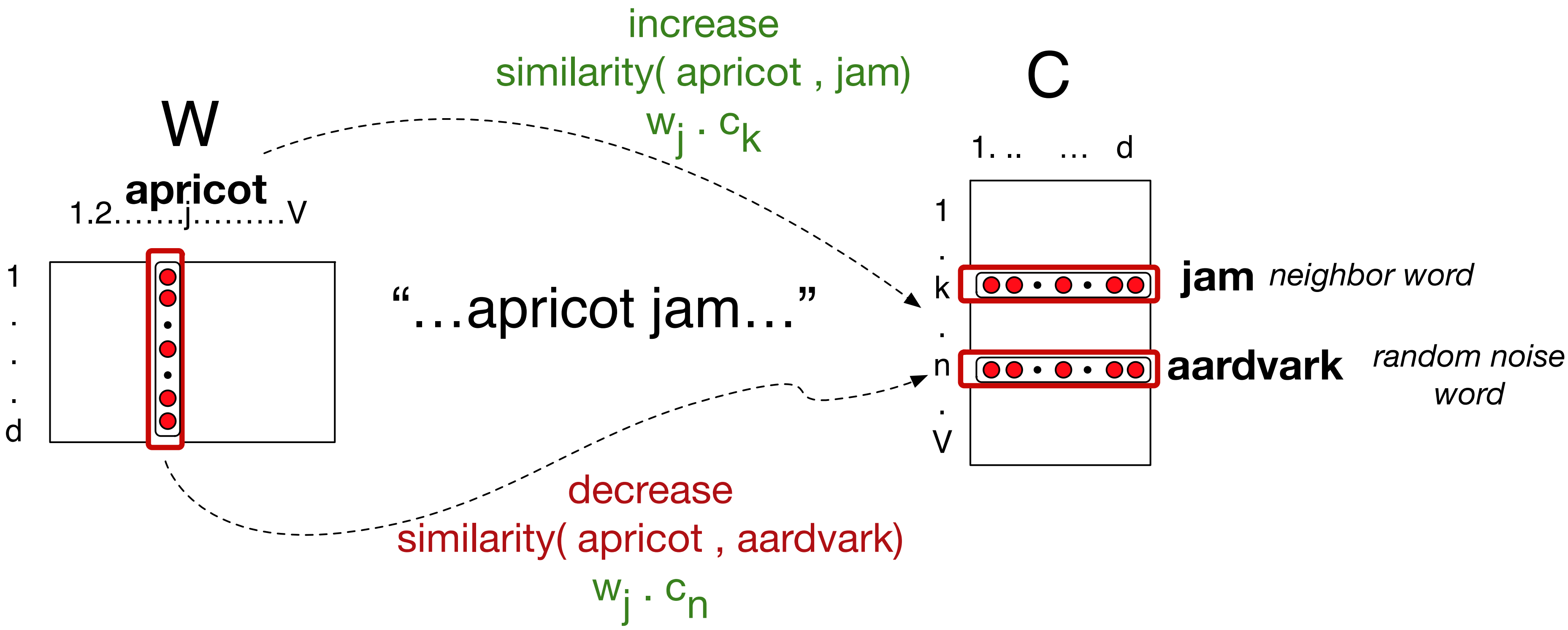
$$\sum_{(t,c)\in +} logP(+|t,c) + \sum_{(t,c)\in -} logP(-|t,c)$$

- Maximize the + label for the pairs from the positive training data, and the – label for the pairs sample from the negative data.

# Focusing on one target word t:

- going back to the dot product

$$
\begin{aligned}
L(\theta) &= \log P(+|t,c) + \sum_{i=1} \log P(-|t,n_i) \\
&= \log \sigma(c \cdot t) + \sum_{i=1}^{k} \log \sigma(-n_i \cdot t) \\
&= \log \frac{1}{1+e^{-c \cdot t}} + \sum_{i=1}^{k} \log \frac{1}{1+e^{n_i \cdot t}}
\end{aligned}
$$



increase
similarity( apricot , jam)
$w_j \cdot c_k$

“…apricot jam…”

**jam** *neighbor word*

**aardvark** *random noise word*

decrease
similarity( apricot , aardvark)
$w_j \cdot c_n$

# Train using gradient descent

- Actually learns two separate embedding matrices W and C
- Can use W and throw away C, or merge them somehow

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

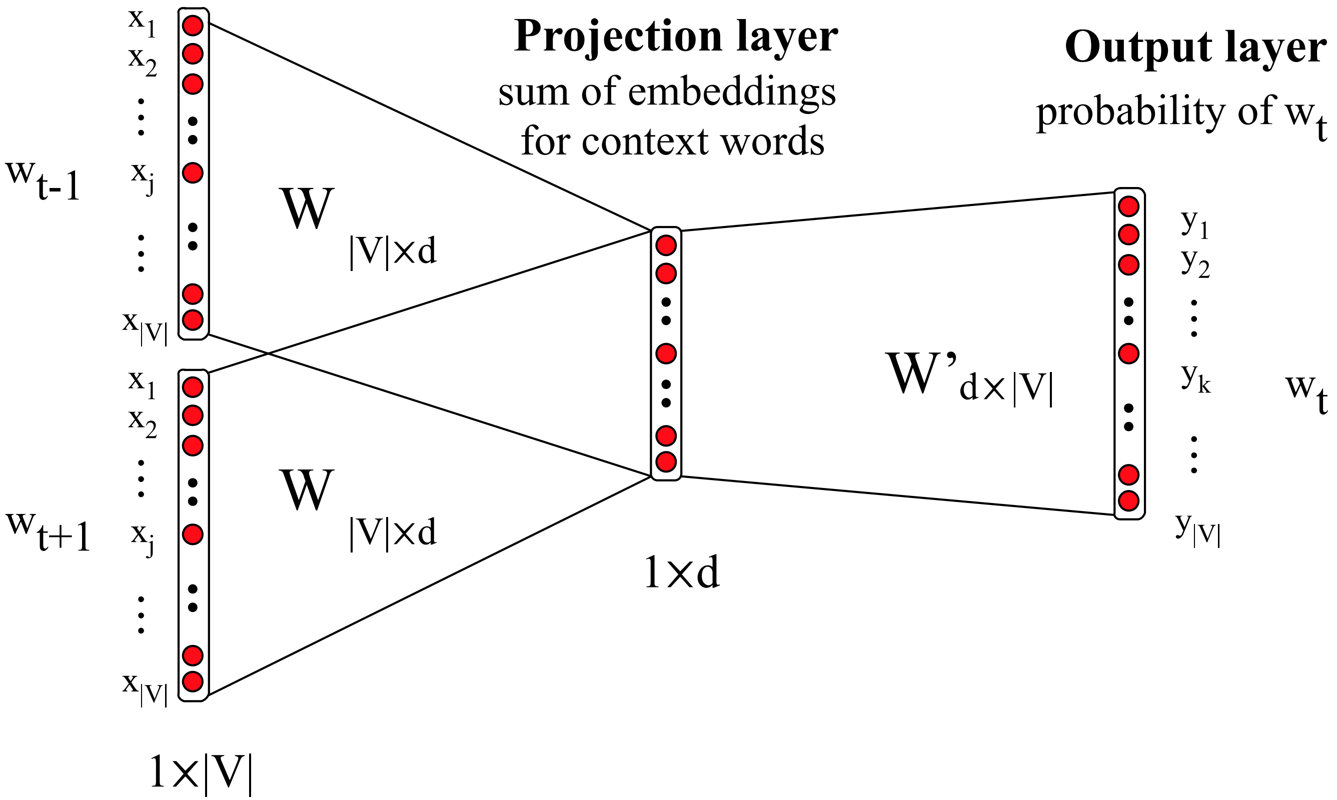
- Start with V random 300-dimensional vectors as initial embeddings
- Use logistic regression,
  - Take a corpus and take pairs of words that co-occur as positive examples
  - Take pairs of words that don't co-occur as negative examples
  - Train the classifier to distinguish these by slowly adjusting all the embeddings to improve the classifier performance
- Throw away the classifier code and keep the embeddings.

# CBOW (Continuous Bag of Words)

CBOW learns a word embedding by maximizing the log conditional probability of a word given the bag of context words occurring within a fixed-sized window around that word.

**Input layer**

1-hot input vectors
for each context word



**Projection layer**

sum of embeddings
for context words

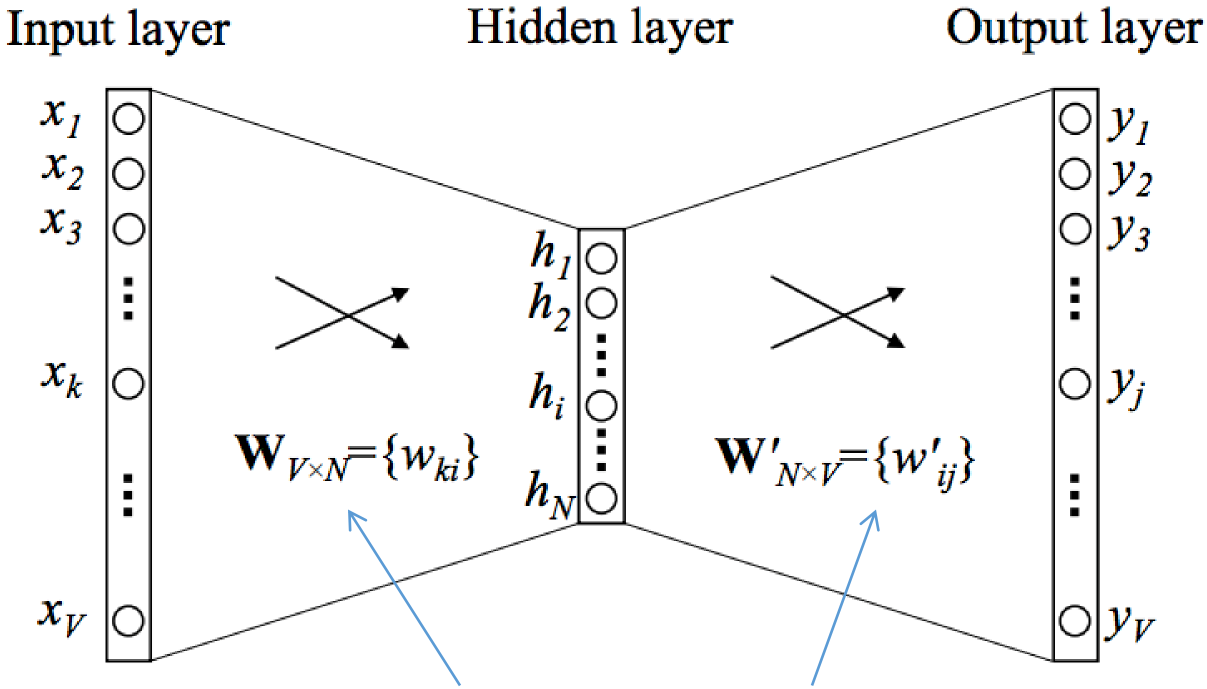
**Output layer**

probability of $w_t$

# Details of 1 word context CBOW

- Objective function: Maximize the log probability of a target word given a context word

One-hot vector



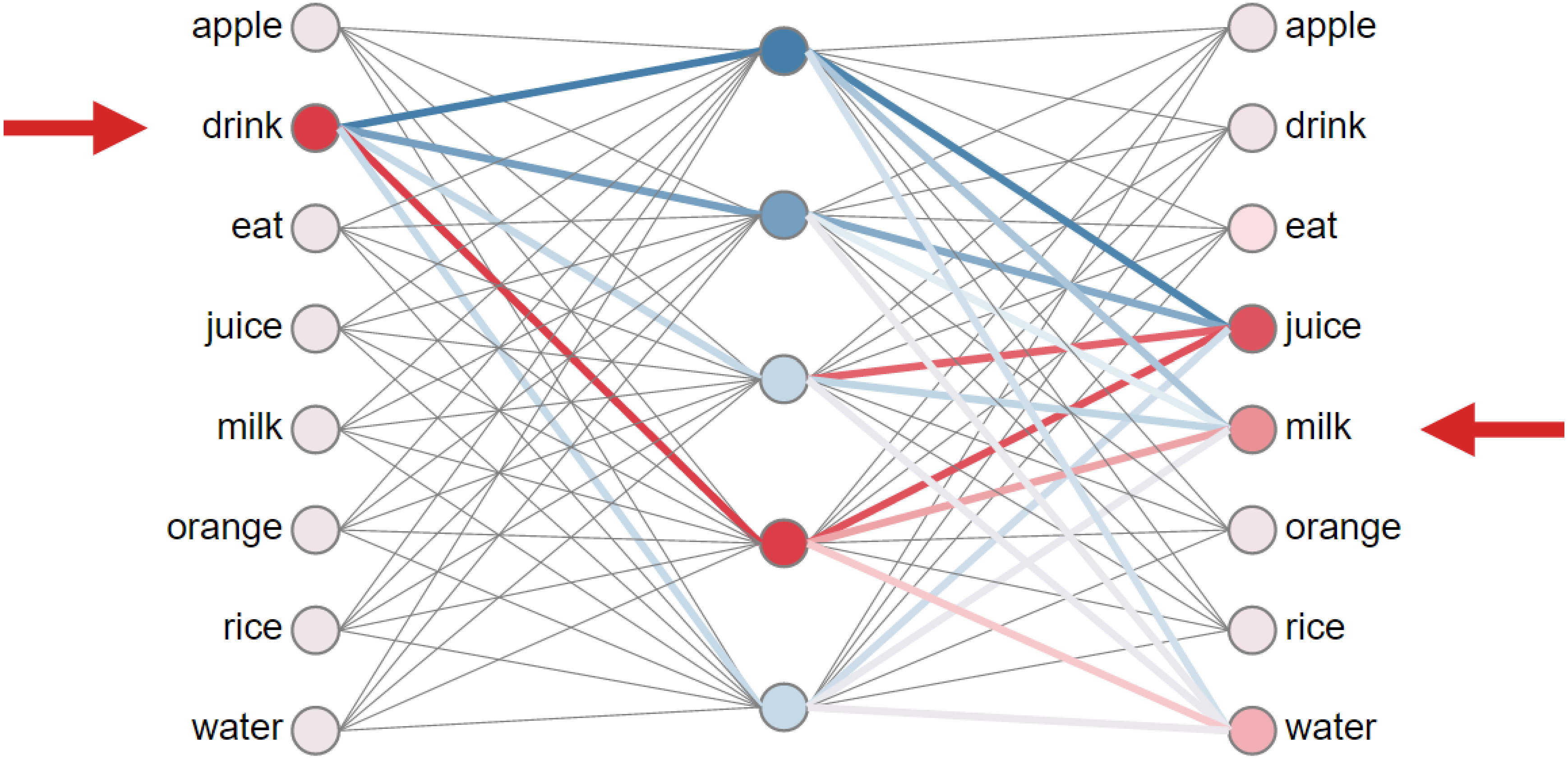
Attempts to predict one-hot vector

These matrices have word vectors!

# Training regime

- Start with small, random vectors for words
- Iteratively go through millions of words in contexts
    - Work out prediction, work out error
    - Backpropagate error to update word vectors
    - Repeat
- Result is dense vectors for all words

$$\textit{linguistics} = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{pmatrix}$$


apple
drink
eat
juice
milk
orange
rice
water
apple
drink
eat
juice
milk
orange
rice
water

# Training word2Vec embeddings

- Download, e.g., https://code.google.com/archive/p/word2vec/
- Word2Vec comes bundled with many files. Two important ones:
  - *Word2vec.c* - the actual Word2Vec program written in C; is executed in command line
  - *Demo-word.sh* - shell script containing example of how to run *Word2Vec.c* on test data
- To use Word2Vec, you need a corpus (e.g., collection of tweets, news articles, product reviews)
  - Word2Vec expects a sequence of sentences as input
  - One input file containing many sentences, with one sentence per line
- Precomputed embeddings exist for many languages
- Word Embedding Visualization http://ronxin.github.io/wevi/

- fastText variant or word2vec uses subword input and is more suitable for morphologically rich languages https://fasttext.cc

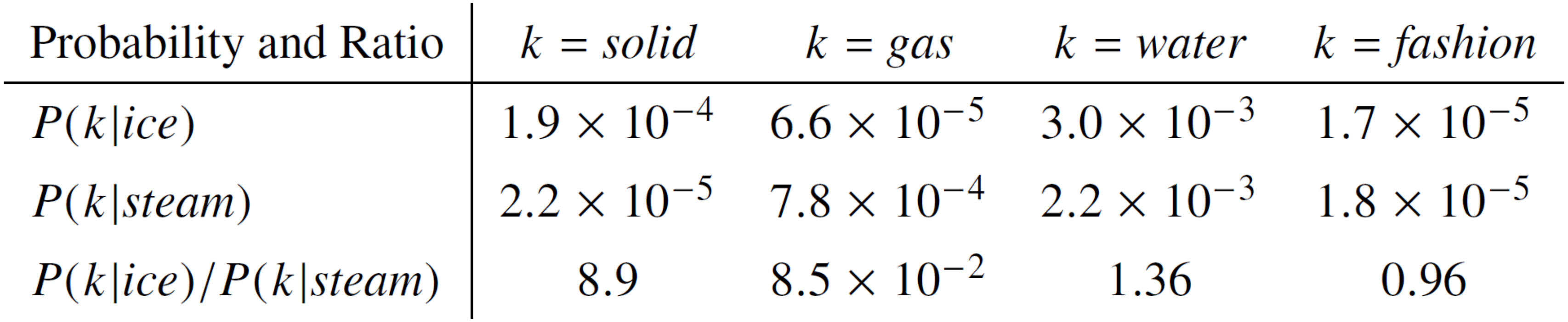
# GloVe: Global Vectors for Word Representations

- First appeared in 2014
- It leverages the statistics of the word occurrences in the corpus and uses neural network to represent the meaning of such statistics
- Idea similar to Latent Semantic Analysis (LSA)
- Captures the global and local context of the word
- Training only on the nonzero elements in a word-word cooccurrence matrix, rather than on the entire sparse matrix or on individual context windows in a large corpus.

Pre-trained models
https://nlp.stanford.edu/projects/glove/

# GloVe: the main idea

- Uses ratios of co-occurrence probabilities, rather than the co-occurrence probabilities themselves
- Target words: steam and ice, probe words k



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

- Only in the ratio does noise from non-discriminative words like water and fashion cancel out, so that large values (much greater than 1) correlate well with properties specific to ice, and small values (much less than 1) correlate well with properties specific of steam.

# GloVe loss function: Least Squares Problem

context vector

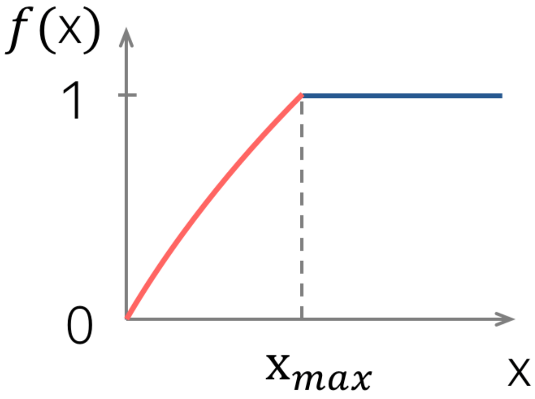
word vector

bias terms (also learned)

$$J(\theta) = \sum_{w,c \in V} \underbrace{f(\mathrm{N}(w, c))} \cdot (u_c^T v_w + b_c + \overline{b_w} - \log \mathrm{N}(w, c))^2$$

Weighting function to:

- penalize rare events
- not to over-weight frequent events



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

$\alpha = 0.75, x_{max} = 100$

# FastText representation

- First appeared in 2016
- Based on the word2vec **skipgram** model, only that is uses the subword information (revised later in the slides)
- A word is represented as a sum of character n-gram embeddings that appeared in the word

$$\sum_{t=1}^{T}\left[\sum_{c\in\mathcal{C}_t}\mathcal{L}(s(w_t,w_c)) + \sum_{n\in\mathcal{N}_{t,c}}\mathcal{L}(-s(w_t,n))\right],\qquad s(w,c)=\sum_{g\in\mathcal{G}_w}\mathbf{z}_g^T\mathbf{v}_c$$

$$\mathcal{L}: x \to \log(1+e^{-x})$$

# FastText compared with word2vec skipgram model

- FastText outperforms skipgram in most scenarios and datasets when dealing with syntactic tasks
- For sematic tasks, the fastText is (2-5 per cent) less accurate than the skipgram model
- Is able to generate out-of-vocabulary word embeddings

Pre-trained models (157 languages, aligned vectors)
https://fasttext.cc/docs/en/english-vectors.html

Several variants for Slovene, see Clarin.si

# Phrase representation

- Word embedding models in their most basic form is based on unigrams
- Enriching the models with word n-grams to capture richer information
- The chosen bigrams are merged in a selected n-gram into a single token
- Usually done in 2-6 passes with decreasing threshold value

$$\text{score}(w_i, w_j) = \frac{\text{count}(w_i, w_j) - \delta}{\text{count}(w_i) \cdot \text{count}(w_j)}$$

*Word2phrase* from word2vec (C programming language)
https://github.com/tmikolov/word2vec

# Subword Information

- Standard word embedding models ignore the internal structure and information of the words
- An effective approach is to enrich the word vectors with a bag of character n-grams (as in *fastText*)
  - Can be also derived from the singular value decomposition (SVD) of the co-occurrence matrix

$$v_w + \frac{1}{|N|} \sum_{n \in N} x_n$$

- In practice, the set of n-grams is restricted with 3-6 characters

# Position-dependent weighting

## Common Practice when Training Models

- The context vector is simply the average of the word vectors contained in it – oblivious to the position of each word
- A simple solution is to learn position representations and use them to reweight the word vectors
- Adds minimal computational cost

$$v_C = \sum_{p \in P} d_p \odot u_{t+p}$$

# Model Comparison

| Model | Advantages | Disadvantages |
|---|---|---|
| Continuous BOW | • Mediocre semantic accuracy<br>• Absence in papers; unpopular in practice | • Ignores global vocabulary information<br>• Does not handle out-of-vocabulary words |
| Skipgram | • Good semantic accuracy<br>• Pre-trained models available | |
| GloVe | • Uses global information of vocabulary<br>• Captures local and global context of words<br>• Good syntactic and semantic accuracy<br>• Pre-trained models available | • Does not handle out-of-vocabulary words |
| FastText | • Handles out-of-vocabulary words<br>• Good at syntactic tasks<br>• Pre-trained models available<br>• Available aligned word vectors | • Takes 1.5x longer to train than *skipgram* |

# Evaluation of embeddings

- Related to general evaluation in NLP: intrinsic vs. extrinsic
- Intrinsic:
  - Evaluation on a specific/intermediate subtask
  - Fast to compute
  - Helps to understand that system
  - Not clear if really helpful unless correlation to real task is established
- Extrinsic:
  - Evaluation on a real task
  - Can take a long time to compute accuracy
  - Unclear if the subsystem is the problem or its interaction or other subsystems
  - If replacing exactly one subsystem with another improves accuracy then we are doing well

# Intrinsic human-based evaluation

- Compare to human scores on word similarity-type tasks:
- WordSim-353 (Finkelstein et al., 2002)
- SimLex-999 (Hill et al., 2015)
- CoSimLex (SemEval 2020, words in context)
- Stanford Contextual Word Similarity (SCWS) dataset (Huang et al., 2012)
- TOEFL dataset: *Levied is closest in meaning to: imposed, believed, requested, correlated*

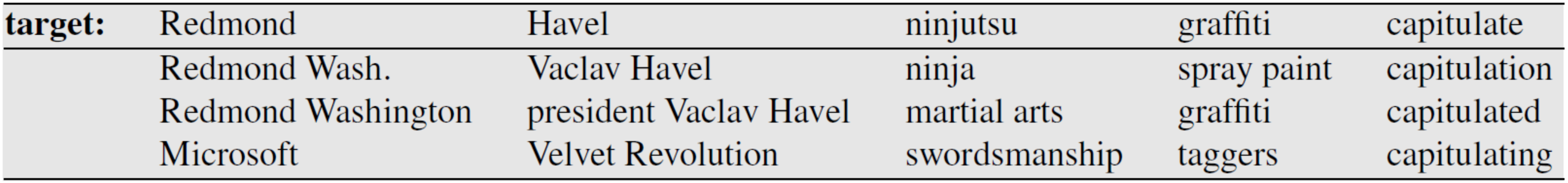
# Properties of embeddings

Similarity depends on window size C

- C = ±2 The nearest words to *Hogwarts:*
  - *Sunnydale*
  - *Evernight*
- C = ±5 The nearest words to *Hogwarts:*
  - *Dumbledore*
  - *Malfoy*
  - *halfblood*

# Examples of embeddings

- groups of similar words (extension to multi word expressions)

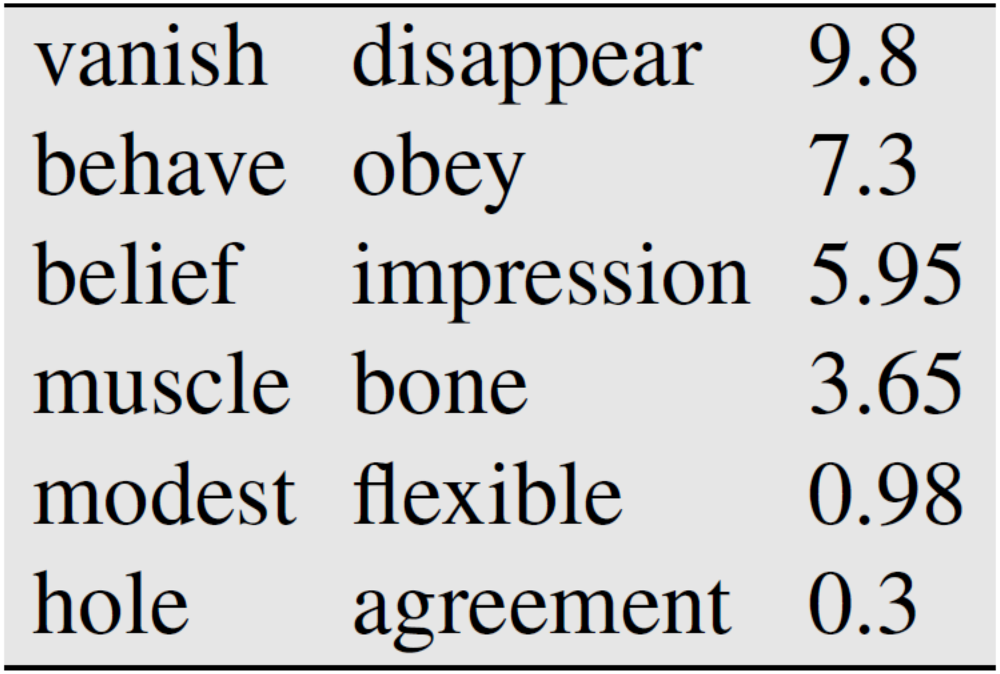


| **target:** | Redmond | Havel | ninjutsu | graffiti | capitulate |
|---|---|---|---|---|---|
| | Redmond Wash. | Vaclav Havel | ninja | spray paint | capitulation |
| | Redmond Washington | president Vaclav Havel | martial arts | graffiti | capitulated |
| | Microsoft | Velvet Revolution | swordsmanship | taggers | capitulating |

- relational similarity

# Simlex-999

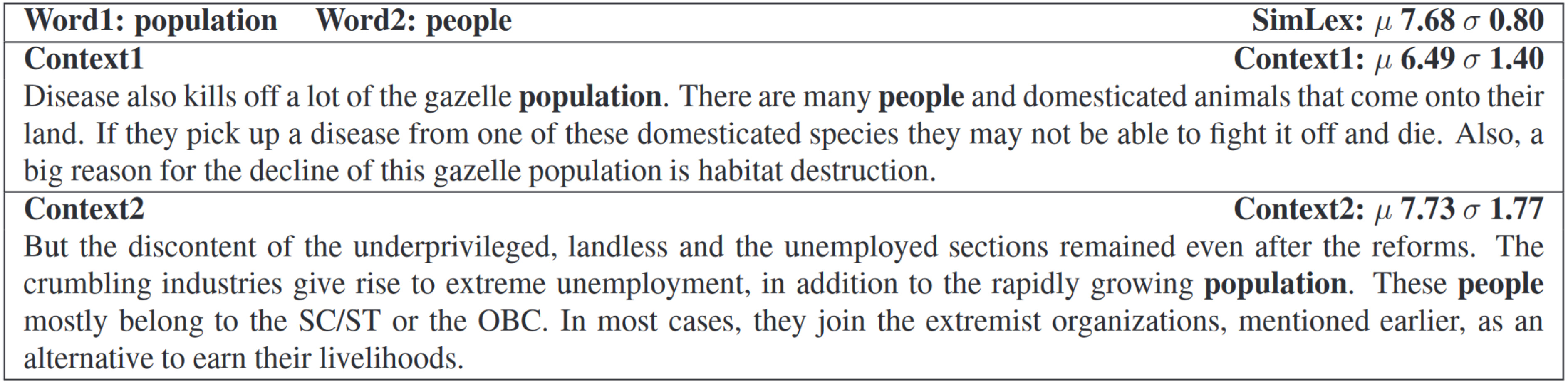
- ask humans to judge how similar one word is to another
- SimLex-999 dataset (Hill et al., 2015) gives values on a scale from 0 to 10
- weakness: no context



| | | |
|---|---|---|
| vanish | disappear | 9.8 |
| behave | obey | 7.3 |
| belief | impression | 5.95 |
| muscle | bone | 3.65 |
| modest | flexible | 0.98 |
| hole | agreement | 0.3 |

# CoSimLex

- human judgement of word similarity in context
- 4 languages (English, Slovene, Croatian, Finnish)



**Word1: population** **Word2: people** **SimLex: $\mu$ 7.68 $\sigma$ 0.80**

**Context1** **Context1: $\mu$ 6.49 $\sigma$ 1.40**

Disease also kills off a lot of the gazelle **population**. There are many **people** and domesticated animals that come onto their land. If they pick up a disease from one of these domesticated species they may not be able to fight it off and die. Also, a big reason for the decline of this gazelle population is habitat destruction.

**Context2** **Context2: $\mu$ 7.73 $\sigma$ 1.77**

But the discontent of the underprivileged, landless and the unemployed sections remained even after the reforms. The crumbling industries give rise to extreme unemployment, in addition to the rapidly growing **population**. These **people** mostly belong to the SC/ST or the OBC. In most cases, they join the extremist organizations, mentioned earlier, as an alternative to earn their livelihoods.

Armendariz, C.S., Purver, M., Ulčar, M., Pollak, S., Ljubešić, N. and Granroth-Wilding, M., 2020, May. CoSimLex: A Resource for Evaluating Graded Word Similarity in Context. In *Proceedings of The 12th Language Resources and Evaluation Conference,* pp. 5878-5886.

# Linear Relationships in word2vec

These representations are *very good* at encoding similarity and dimensions of similarity!

- Analogies testing dimensions of similarity can be solved quite well just by doing vector subtraction in the embedding space
- Syntactically
  - $x_{apple} - x_{apples} \approx x_{car} - x_{cars} \approx x_{family} - x_{families}$
  - Similarly for verb and adjective morphological form
- Semantically (Semeval 2012 task 2)
  - $x_{shirt} - x_{clothing} \approx x_{chair} - x_{furniture}$
  - $x_{king} - x_{man} \approx x_{queen} - x_{woman}$
- 15 relations in 7 languages

- Ulčar, M., Vaik, K., Lindström, J., Dailidėnaitė, M. and Robnik-Šikonja, M., 2020. Multilingual Culture-Independent Word Analogy Datasets. In *Proceedings of The 12th Language Resources and Evaluation Conference* (pp. 4074-4080).

# Word Analogies

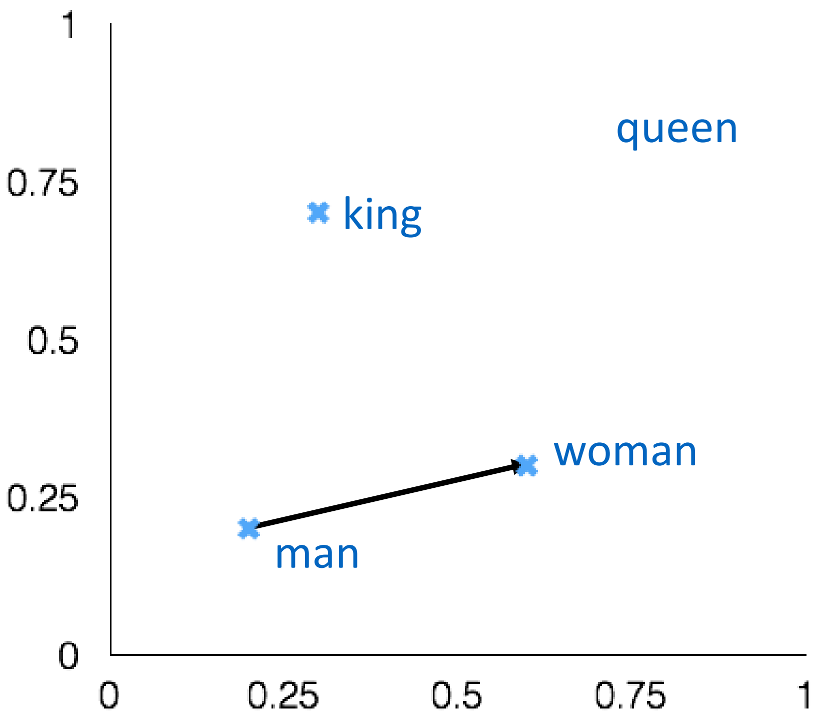
Test for linear relationships, examined by Mikolov et al.

a:b :: c:? →

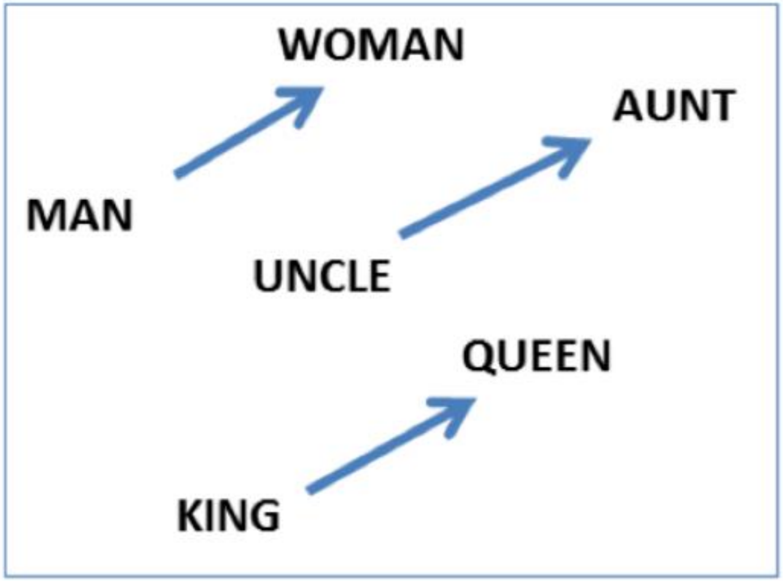
$$d = \arg\max_{x} \frac{(w_b - w_a + w_c)^T w_x}{||w_b - w_a + w_c||}$$

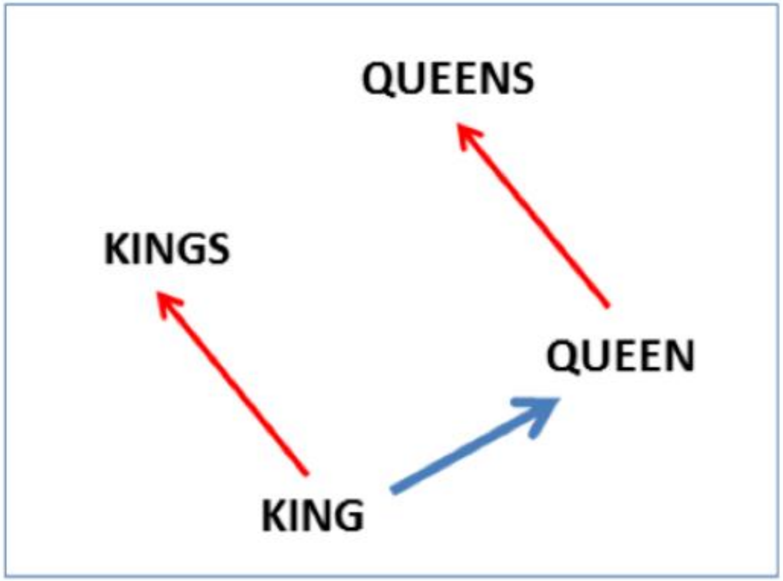
man:woman :: king:?

| | | |
|---|---|---|
| + | king | [ 0.30 0.70 ] |
| – | man | [ 0.20 0.20 ] |
| + | woman | [ 0.60 0.30 ] |
| | queen | [ 0.70 0.80 ] |



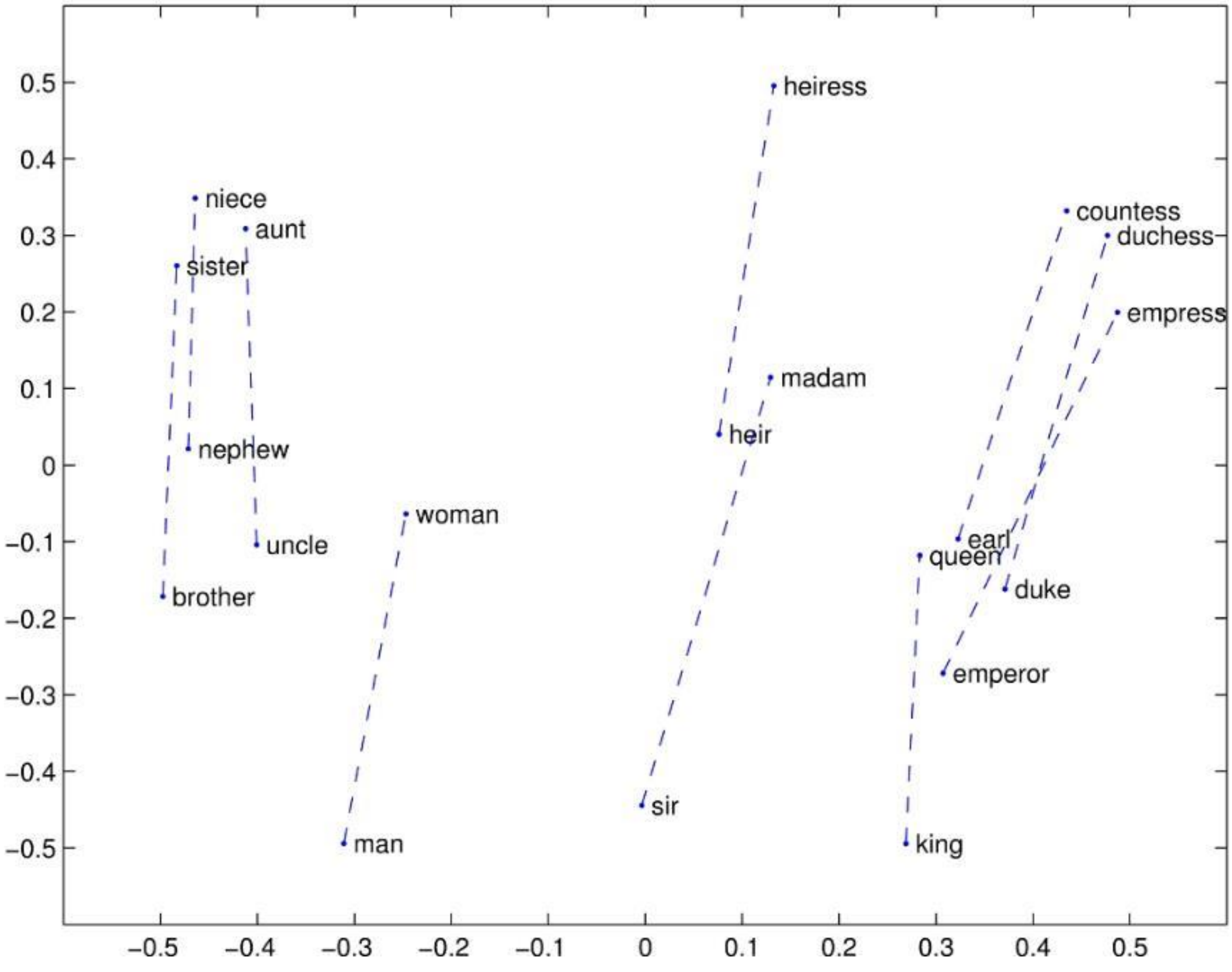
# Relational similarity

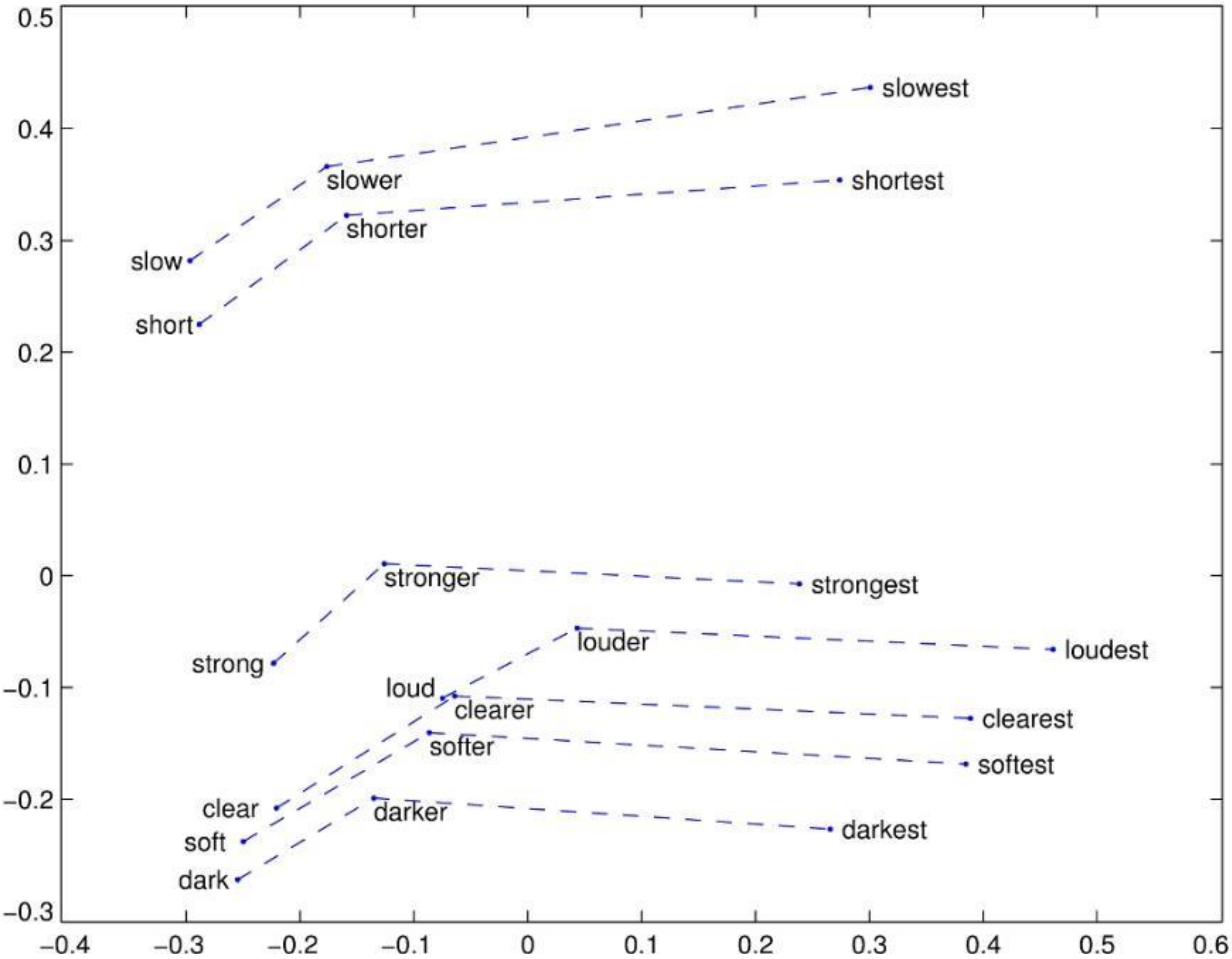




vector(*‘king’*) - vector(*‘man’*) + vector(*‘woman’*) ≈ vector(‘queen’)

vector(*‘Paris’*) - vector(*‘France’*) + vector(*‘Italy’*) ≈ vector(‘Rome’)


heiress
niece
aunt
sister
countess
duchess
empress
madam
heir
nephew
woman
uncle
earl
queen
duke
brother
emperor
sir
man
king
0.5
0.4
0.3
0.2
0.1
0
−0.1
−0.2
−0.3
−0.4
−0.5
−0.5
−0.4
−0.3
−0.2
−0.1
0
0.1
0.2
0.3
0.4
0.5


slowest
slower
shortest
shorter
slow
short
stronger
strongest
strong
louder
loudest
loud
clearer
clearest
softer
softest
clear
darker
darkest
soft
dark
0.5
0.4
0.3
0.2
0.1
0
−0.1
−0.2
−0.3
−0.4
−0.3
−0.2
−0.1
0
0.1
0.2
0.3
0.4
0.5
0.6

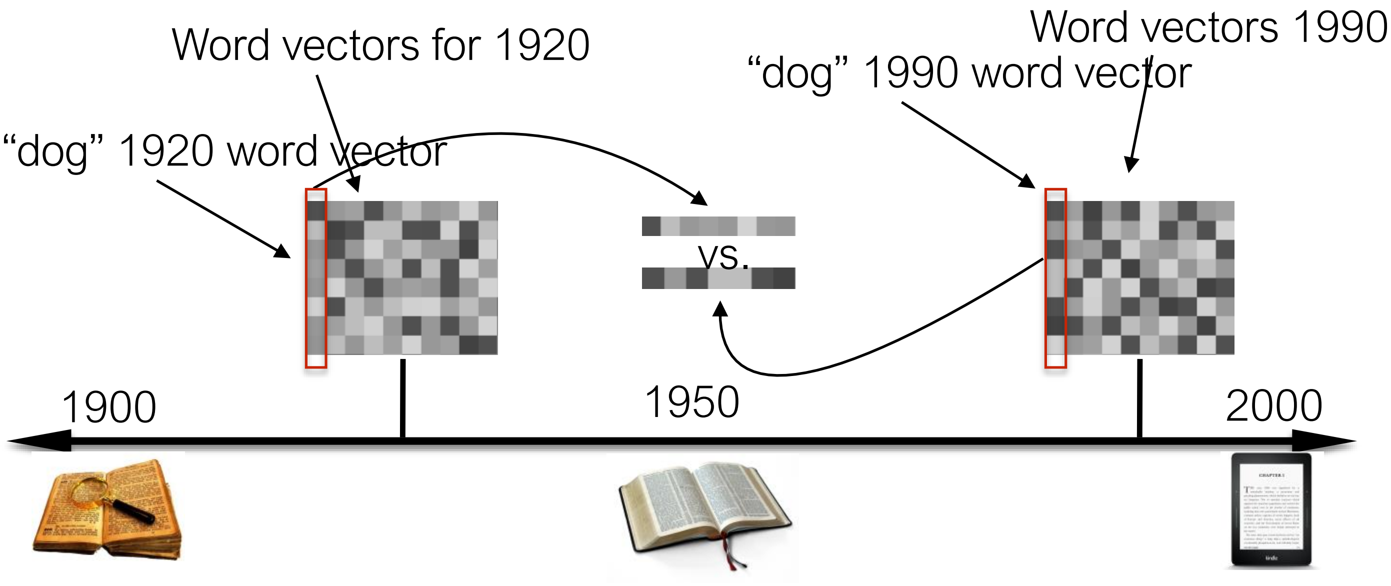
# Embeddings visualization

- https://projector.tensorflow.org/

# Embeddings can help study word history

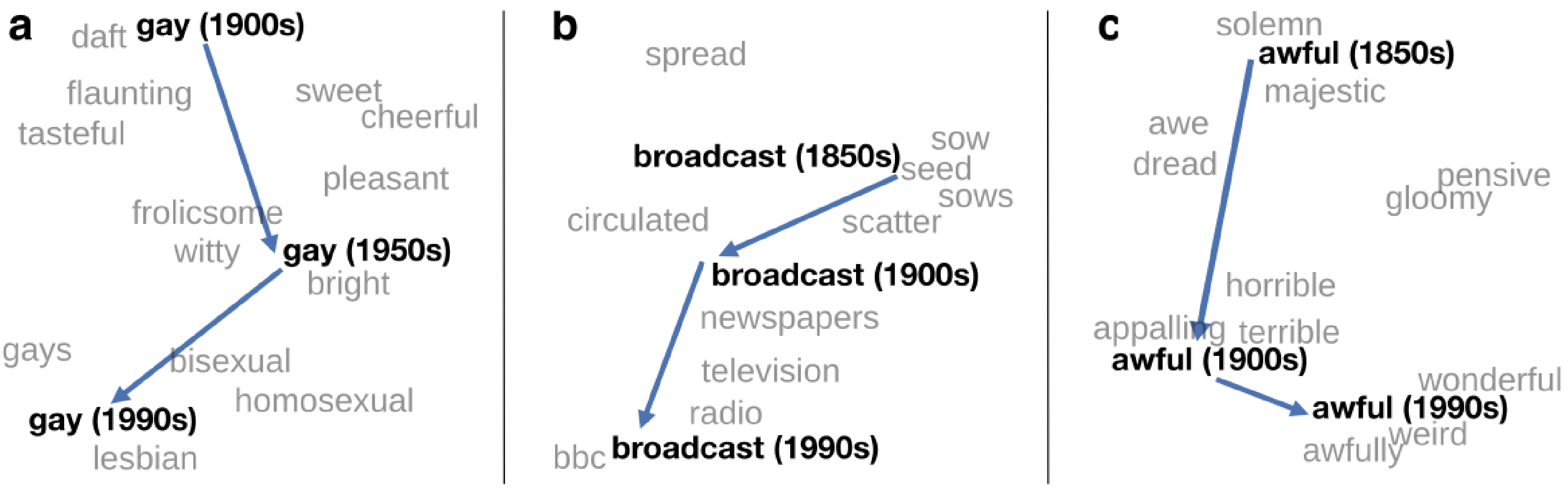
- Train embeddings on old books to study changes in word meaning

# Diachronic word embeddings for studying language change



# Visualizing changes

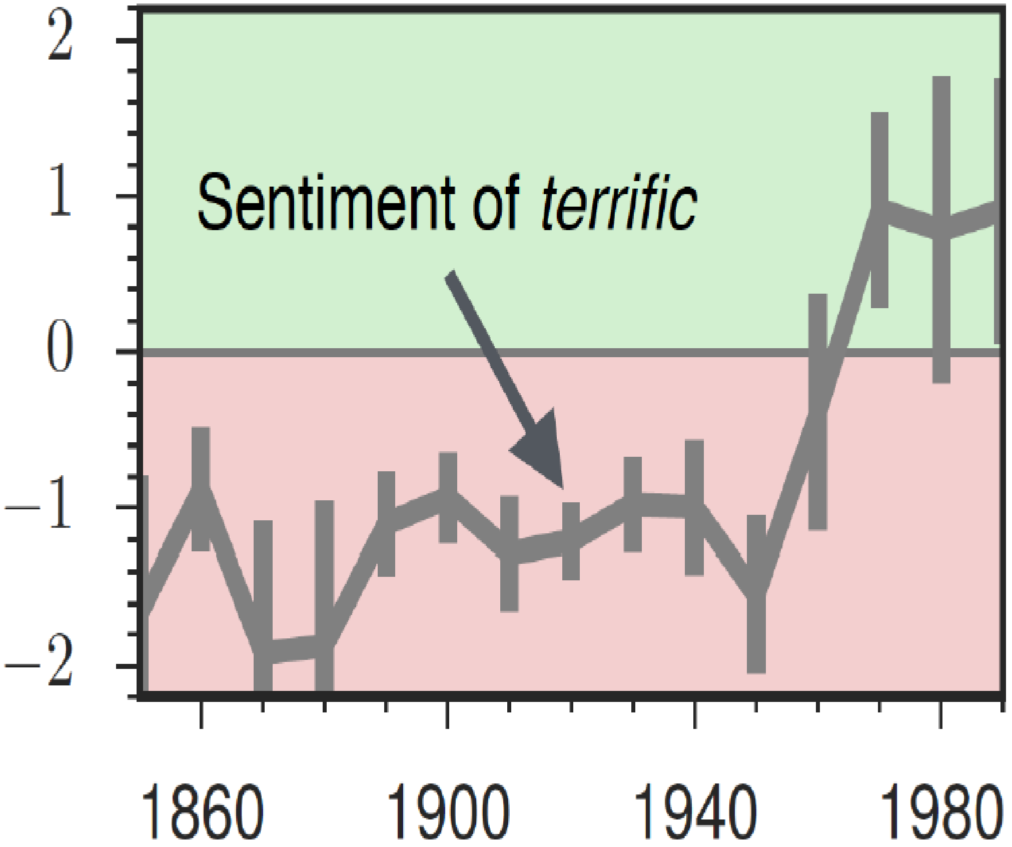
Project 300 dimensions down into 2



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

# The evolution of sentiment words

Negative words change faster than positive words



# Embeddings reflect cultural bias

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 *Advances in Neural Information Processing Systems*, pp. 4349-4357. 2016.

- Ask “Paris : France :: Tokyo : x”
  - x = Japan
- Ask “father : doctor :: mother : x”
  - x = nurse
- Ask “man : computer programmer :: woman : x”
  - x = homemaker

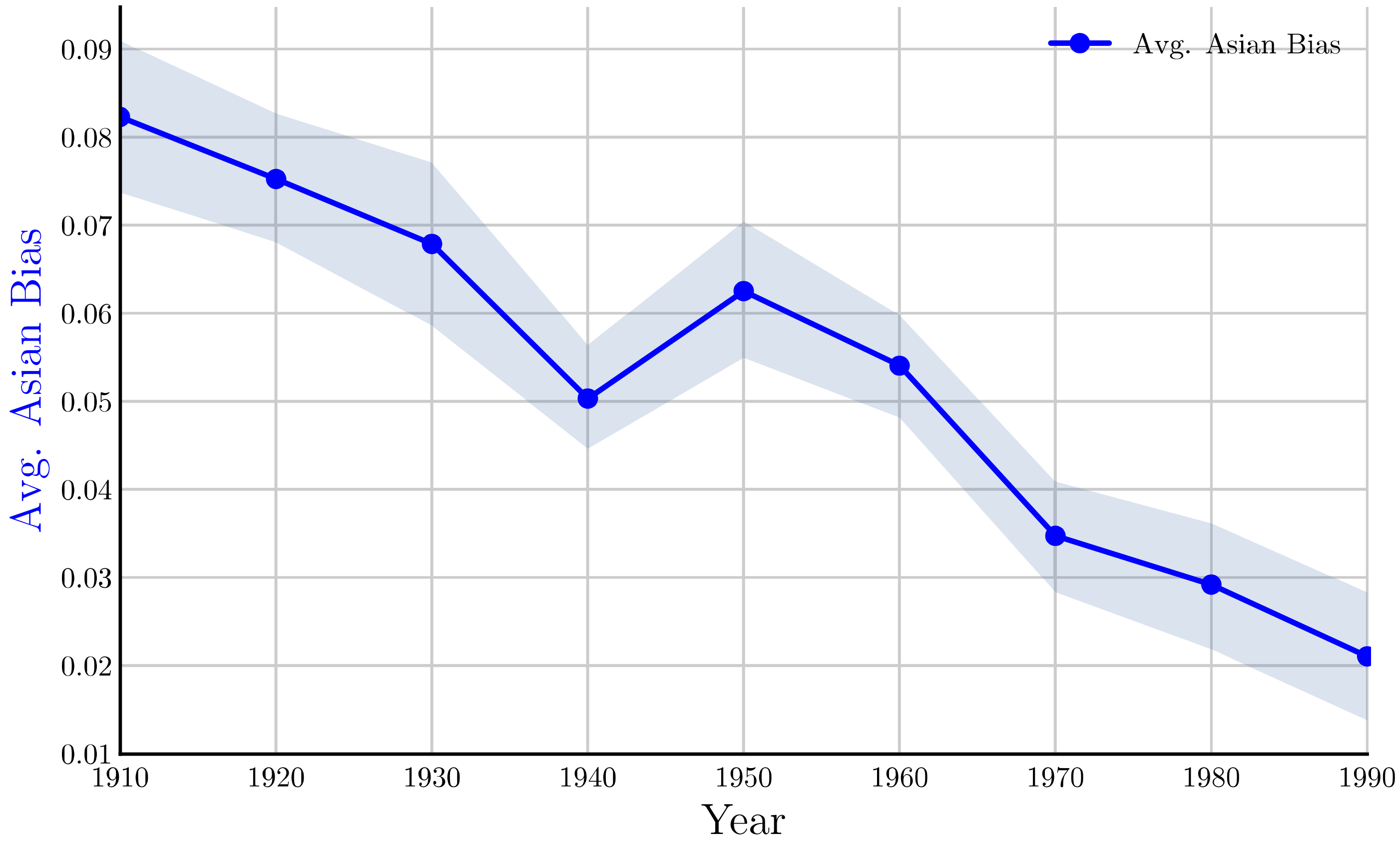
# Embeddings reflect cultural bias

Caliskan, Aylin, Joanna J. Bruson and Arvind Narayanan. 2017. Semantics derived automatically from language corpora contain human-like biases. Science 356:6334, 183-186.

- Implicit Association test (Greenwald et al 1998): How associated are
  - concepts (*flowers*, *insects*) & attributes (*pleasantness*, *unpleasantness*)?
  - Studied by measuring timing latencies for categorization.
- Psychological findings on US participants:
  - African-American names are associated with unpleasant words (more than European-American names)
  - Male names associated more with math, female names with arts
  - Old people's names with unpleasant words, young people with pleasant words.
- Caliskan et al. replication with embeddings:
  - African-American names (*Leroy, Shaniqua*) had a higher GloVe cosine with unpleasant words (*abuse, stink, ugly*)
  - European American names (*Brad, Greg, Courtney*) had a higher cosine with pleasant words (*love, peace, miracle*)
- Embeddings reflect and replicate all sorts of pernicious biases.

# Change in linguistic framing 1910-1990

Change in association of Chinese names with adjectives framed as "othering" (*barbaric*, *monstrous*, *bizarre*)



Garg, Nikhil, Schiebinger, Londa, Jurafsky, Dan, and Zou, James (2018). Word embeddings quantify 100 years of gender and ethnic stereotypes. *Proceedings of the National Academy of Sciences*, *115*(16), E3635–E3644

# Embeddings as a window onto history

Garg, Nikhil, Schiebinger, Londa, Jurafsky, Dan, and Zou, James (2018). Word embeddings quantify 100 years of gender and ethnic stereotypes. *Proceedings of the National Academy of Sciences*, *115*(16), E3635–E3644

- Use the Hamilton historical embeddings
- The cosine similarity of embeddings for decade X for occupations (like teacher) to male vs female names
    - Is correlated with the actual percentage of women teachers in decade X

# History of biased framings of women

Garg, Nikhil, Schiebinger, Londa, Jurafsky, Dan, and Zou, James (2018). Word embeddings quantify 100 years of gender and ethnic stereotypes. *Proceedings of the National Academy of Sciences*, *115*(16), E3635–E3644

- Embeddings for competence adjectives are biased toward men
  - *Smart, wise, brilliant, intelligent, resourceful, thoughtful, logical, etc.*
- This bias is slowly decreasing

# Embeddings reflect ethnic stereotypes over time

Garg, Nikhil, Schiebinger, Londa, Jurafsky, Dan, and Zou, James (2018). Word embeddings quantify 100 years of gender and ethnic stereotypes. *Proceedings of the National Academy of Sciences*, *115*(16), E3635–E3644

- Princeton trilogy experiments
- Attitudes toward ethnic groups (1933, 1951, 1969) scores for adjectives
  - *industrious, superstitious, nationalistic*, etc.
- Cosine of Chinese name embeddings with those adjective embeddings correlates with human ratings.

# Changes in framing: adjectives associated with Chinese

Garg, Nikhil, Schiebinger, Londa, Jurafsky, Dan, and Zou, James (2018). Word embeddings quantify 100 years of gender and ethnic stereotypes. *Proceedings of the National Academy of Sciences*, *115*(16), E3635–E3644

| 1910 | 1950 | 1990 |
|---|---|---|
| Irresponsible | Disorganized | Inhibited |
| Envious | Outrageous | Passive |
| Barbaric | Pompous | Dissolute |
| Aggressive | Unstable | Haughty |
| Transparent | Effeminate | Complacent |
| Monstrous | Unprincipled | Forceful |
| Hateful | Venomous | Fixed |
| Cruel | Disobedient | Active |
| Greedy | Predatory | Sensitive |
| Bizarre | Boisterous | Hearty |

# Directions

- Debiasing algorithms for embeddings
  - Bolukbasi, Tolga, Chang, Kai-Wei, Zou, James Y., Saligrama, Venkatesh, and Kalai, Adam T. (2016). Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *Advances in Neural Information Processing Systems*, pp. 4349–4357.
- Use embeddings as a historical tool to study bias

# Contextual embeddings

- word2vec produces the same vector for a word like bank irrespective of its meaning and context
- recent embeddings take the context into account
- already established as a standard
- ELMo and BERT

# ELMo

- ELMo looks at the entire sentence before assigning each word in it an embedding.
- ELMo predicts the next word in a sequence of words - a task called *Language Modeling*.
- It uses a bi-directional LSTM recurrent neural network
- includes subword units
- as an embedding ELMO uses several layers of the network
- first layers capture morphological and syntactic properties, deeper layers encode semantical properties
- uses several fine tuned parameters
- publicly available for many languages

Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K. and Zettlemoyer, L., 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*
Ulčar, M. and Robnik-Šikonja, M., 2019. High Quality ELMo Embeddings for Seven Less-Resourced Languages. *ArXiv preprint arXiv:1911.10049*.

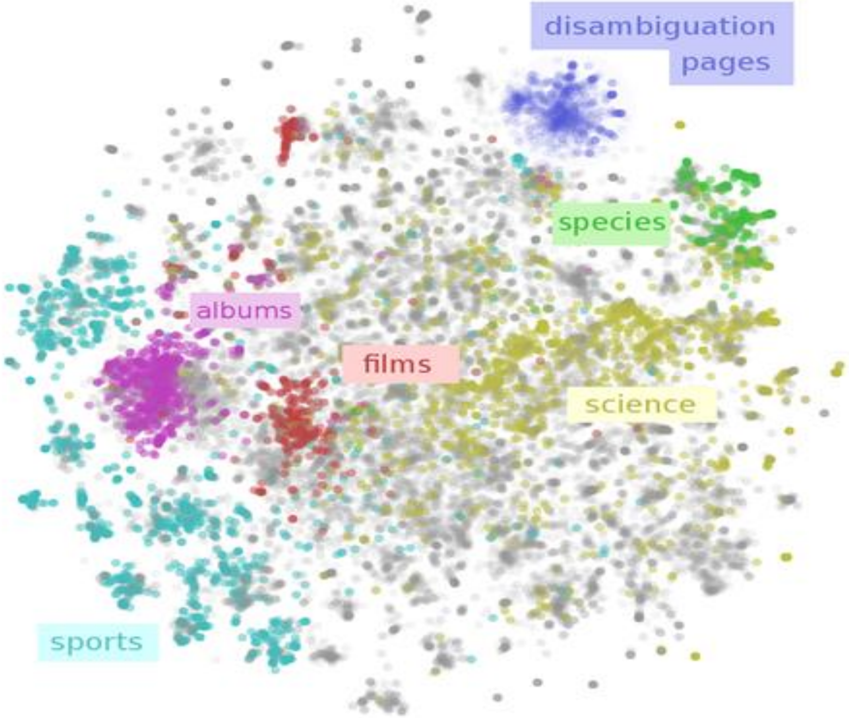
# BERT

- combines several tasks
- predicts masked words in a sentence
- also predicts order of sentences: is sentence A followed by sentence B or not
- combines several hidden layers of the network
- uses transformer neural architecture
- uses several fine tuned parameters
- multilingual variant supports 104 languages by training on Wikipedia
- publicly available

Devlin, J., Chang, M.W., Lee, K. and Tout nova, K., 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *ArXiv preprint arXiv:1810.04805*.

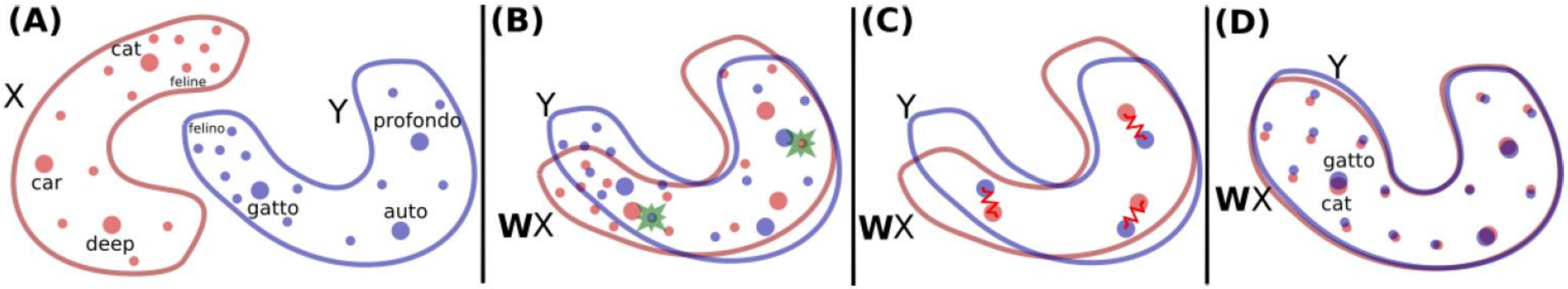
# Cross-lingual embeddings

- embeddings are trained on monolingual resources
- words of one language form a cloud in high dimensional space
- clouds for different languages can be aligned
- $W_1S \approx W_2E$ or $W_1S \approx E$



# Cross-lingual embeddings

- alignment of different word clouds

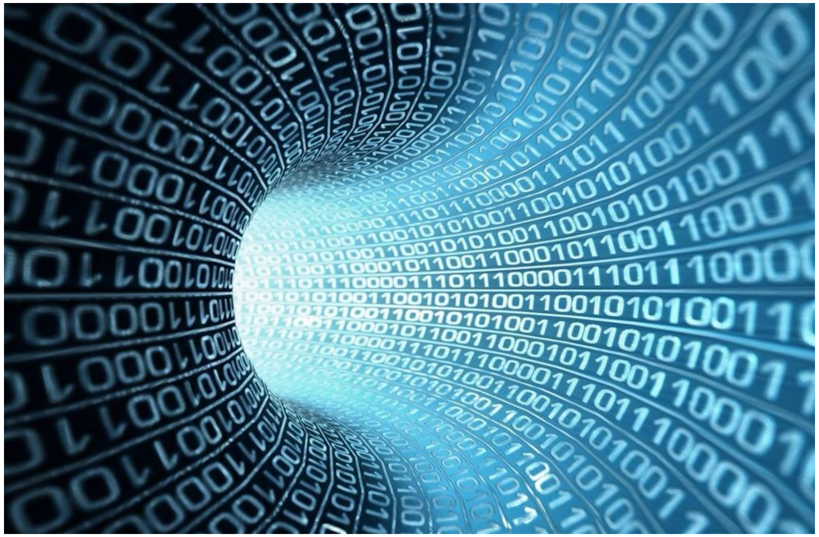


- in unsupervised or supervised way

Conneau, A., Lample, G., Ranzato, M.A., Denoyer, L. and Jégou, H., 2018. Word translation without parallel data. Proceedings of ICLR 2018,
also *ArXiv preprint arXiv:1710.04087*.

# Improving cross-lingual embeddings

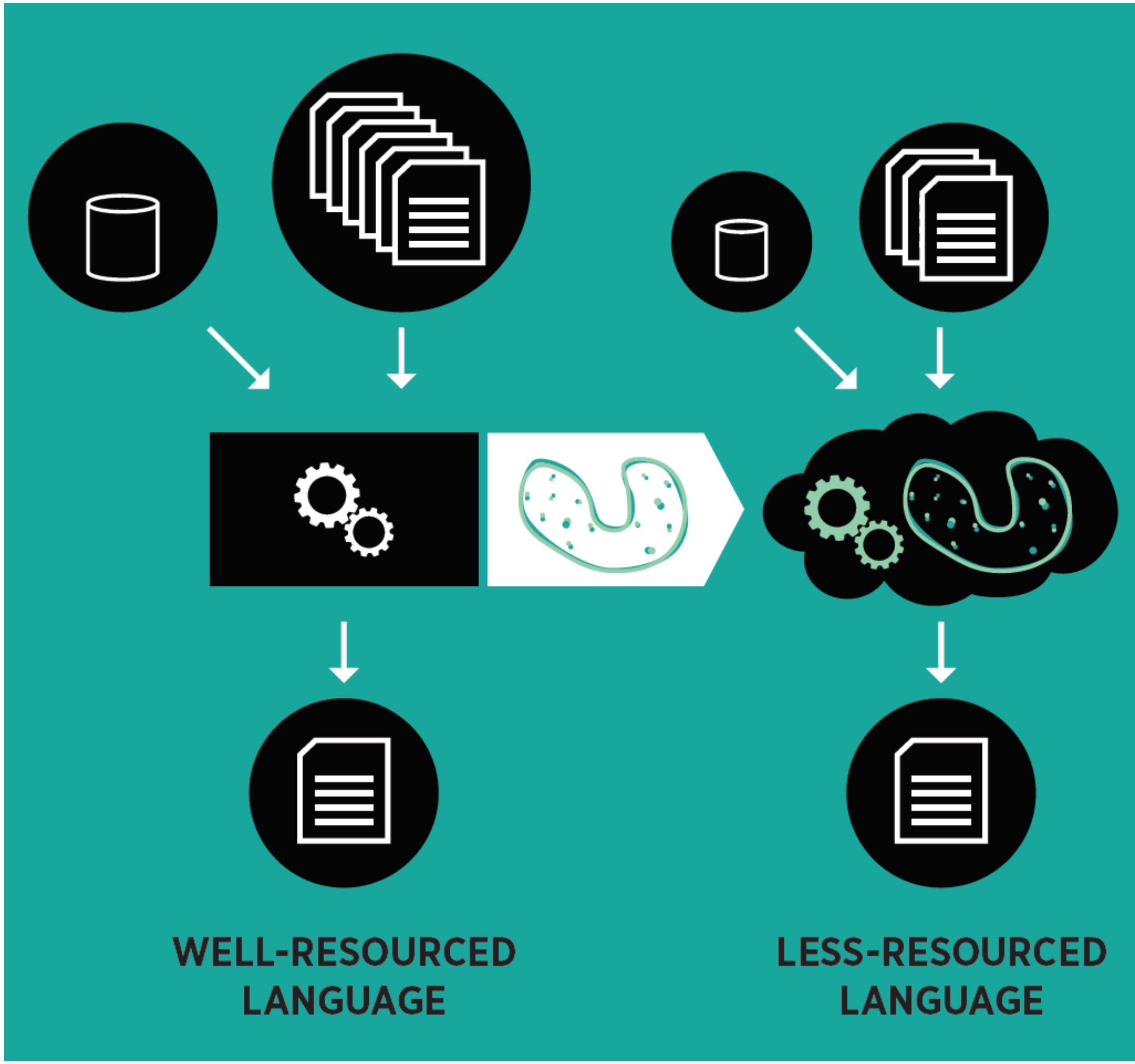
- bilingual and multilingual resources can provide anchoring points for alignment of different word clouds
- alignment of contextual embeddings

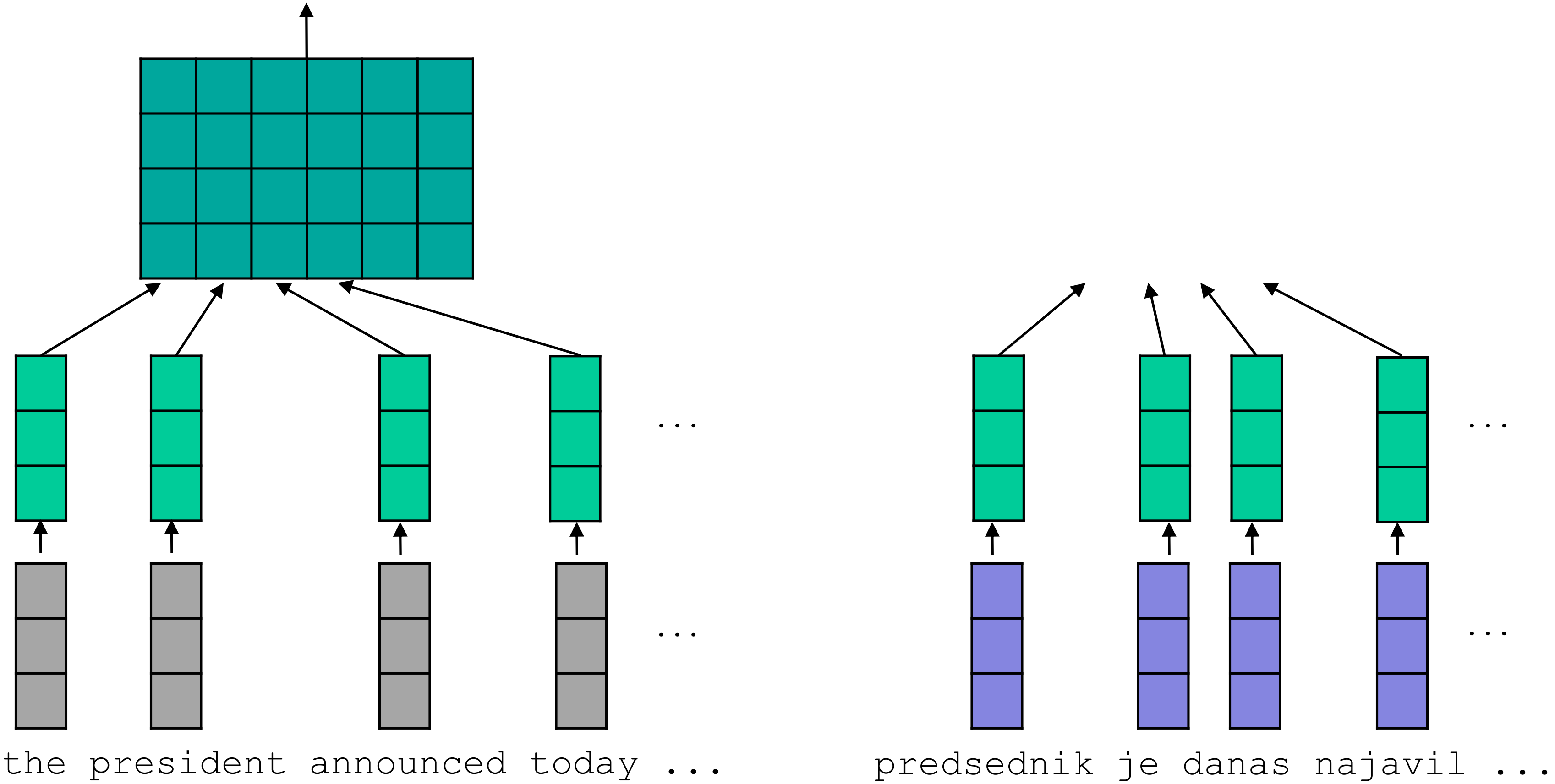


Artetxe, M. and Schwenk, H., 2018. Massively Multilingual Sentence Embeddings for Zero-Shot Cross-Lingual Transfer and Beyond. *ArXiv preprint arXiv:1812.10464*.

# Cross-lingual transfer with embeddings

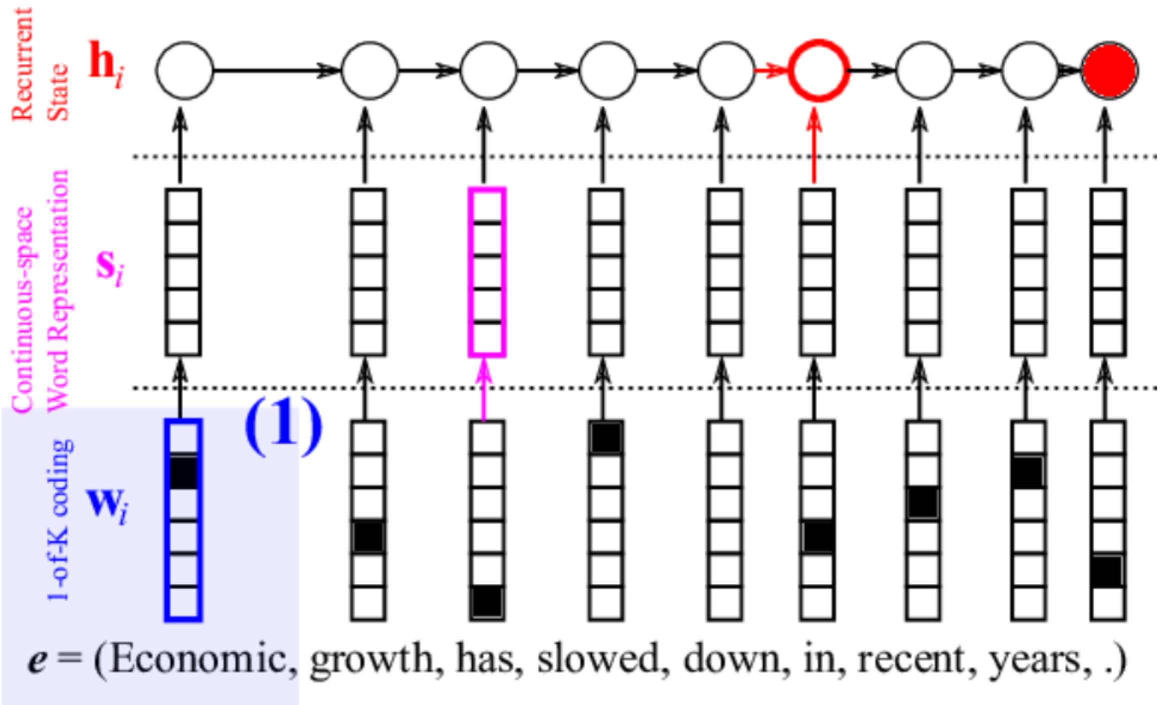
- Transfer of tools trained on monolingual resources




...
...
the president announced today ...
...
...
predsednik je danas najavil ...

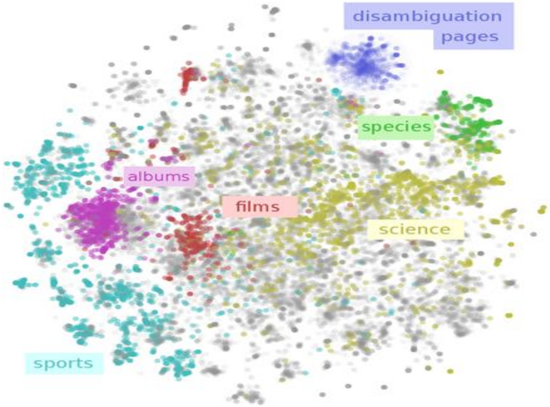
# Using cross-lingual embeddings

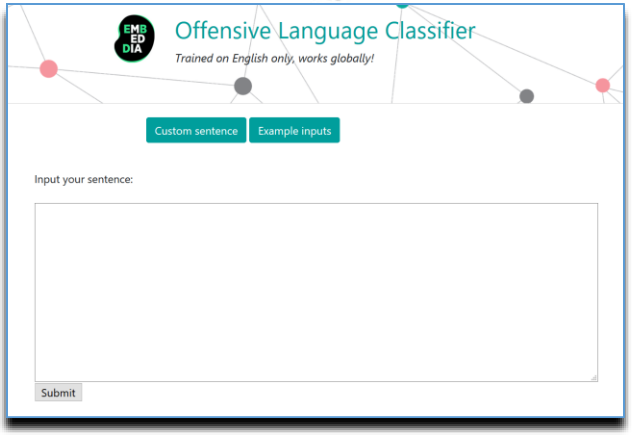
- transfer between languages: models, resources
- embedded words enter neural networks
- replace them with cross-lingual embeddings and easily switch languages


Recurrent State
$\mathbf{h}_i$
Continuous-space Word Representation
$\mathbf{s}_i$
1-of-K coding
$\mathbf{w}_i$
(1)
$e$ = (Economic, growth, has, slowed, down, in, recent, years, .)

# EMBEDDIA project: transfer of machine learning models across many languages

- Word clouds are similar across languages
- EU H2020 project EMBEDDIA develops cross-lingual mappings between word clouds
- Applications in news media industry
- Transfer of tools across languages
- Example: hate/offensive language detector trained on English, works for over 90 languages
- [http://www.embeddia.eu](http://www.embeddia.eu)





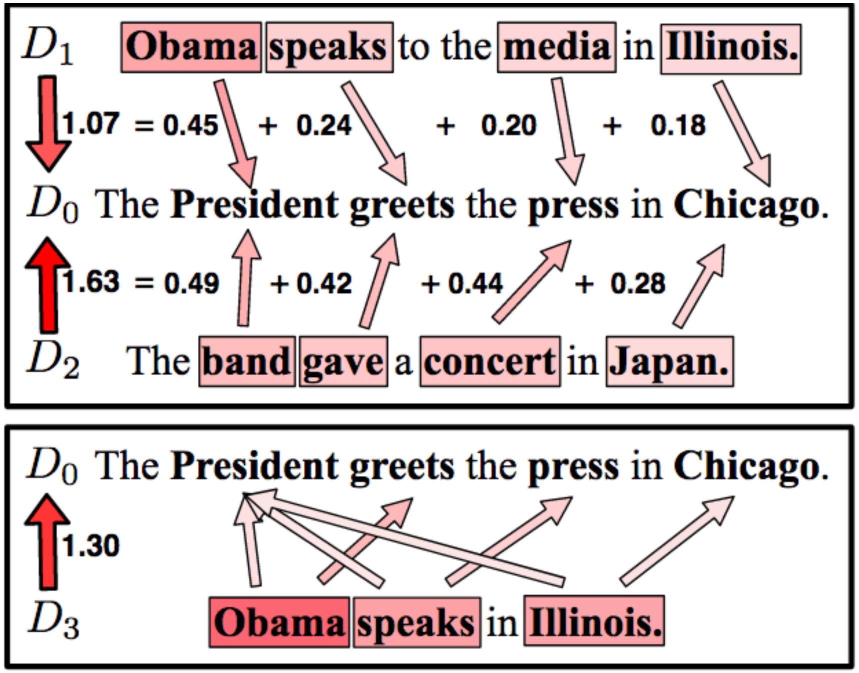
Offensive Language Classifier

*Trained on English only, works globally!*

Input your sentence:

# WMD: Word Mover’s Distance

- Presented in 2015, adaptation of the Earth Mover’s Distance
- Utilizes the distance between embedded word vectors
- The distance between two text documents A and B is viewed as the minimum cumulative distance that words from document A need to travel to match exactly the point cloud of document B



$D_1$ Obama speaks to the media in Illinois.
1.07 = 0.45 + 0.24 + 0.20 + 0.18
$D_0$ The President greets the press in Chicago.
1.63 = 0.49 + 0.42 + 0.44 + 0.28
$D_2$ The band gave a concert in Japan.
$D_0$ The President greets the press in Chicago.
1.30
$D_3$ Obama speaks in Illinois.

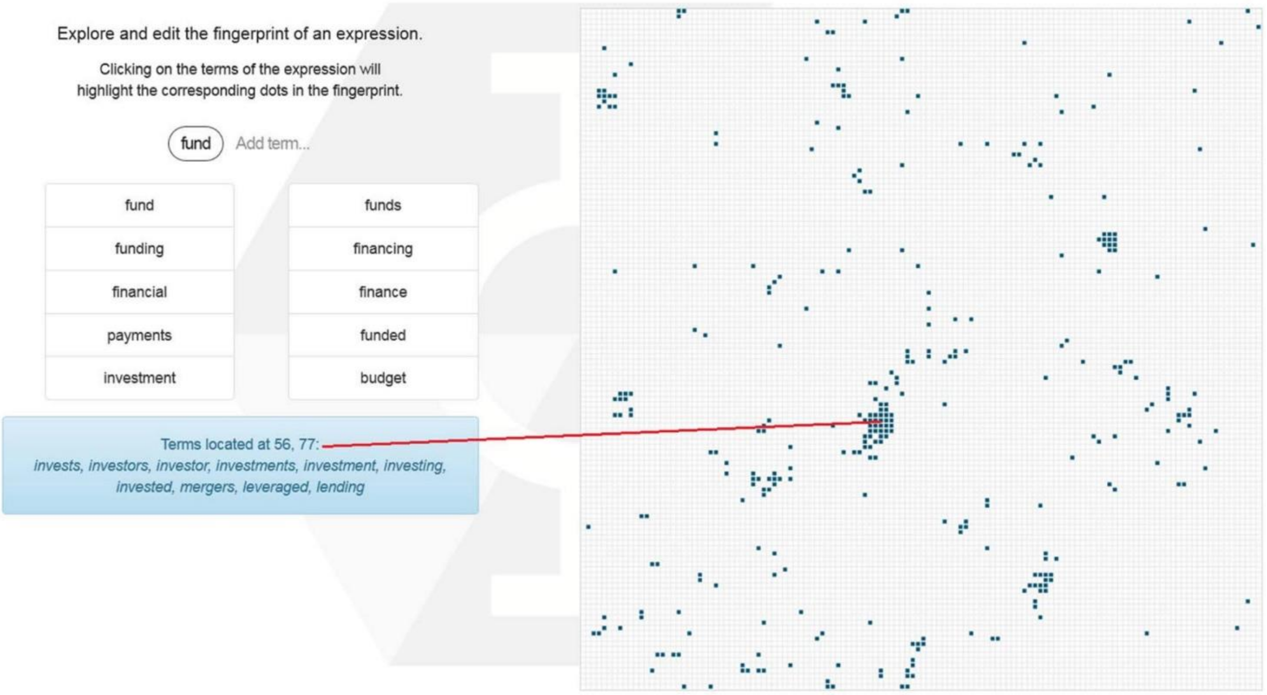
# Semantic fingerprinting

- Cluster a large collection of  documents (4.4 million preselected Wikipedia articles) into $2^{14}$ (or 16,384) clusters of co-occurring words;
- the membership of a word in a cluster represents its fingerprint, i.e.16,384 bits long sparse vectors, where 1 indicates that the word is linked to the particular semantic context, and 0 that it is not
- optionally: arrange bits into a 128 x 128 matrix, where related contexts (those that tend to be linked to the same words) are adjacent to one another
- the semantic fingerprint of a word can be visualized as a 128 × 128 matrix of 0s and 1s (or equivalently, a 16,384-dimensional vector of 0s and 1s).
- the similarity between the two documents can be calculated using the cosine similarity between the two vectors

# Semantic fingerprinting visualization

- Pros:
  - high degree of language independence
  - fast (only bit manipulation)
- Cons:
  - no public implementation

**Appendix A. The semantic fingerprint of the word 'fund'**
(from http://www.cortical.io/static/demos/html/fingerprint-editor.html)



Explore and edit the fingerprint of an expression.

Clicking on the terms of the expression will highlight the corresponding dots in the fingerprint.

| | |
|---|---|
| fund | funds |
| funding | financing |
| financial | finance |
| payments | funded |
| investment | budget |

Terms located at 56, 77:
*invests, investors, investor, investments, investment, investing, invested, mergers, leveraged, lending*

# Conclusion of dense embeddings

- **Concepts** or word senses
  - Have a complex many-to-many association with **words** (homonymy, multiple senses)
  - Have relations with each other
    - Synonymy, Antonymy, Superordinate
  - But are hard to define formally (necessary & sufficient conditions)
- **Embeddings** = vector models of meaning
  - More fine-grained than just a string or index
  - Especially good at modeling similarity/analogy
    - Just download them and use cosines
  - Useful in practice but know they encode cultural stereotypes

# Summary: Embed all the things!

Lots of applications whenever knowing word context or similarity helps prediction:

- Synonym handling in search
- Document topics and similarity
- Ad serving
- Language models: from spelling correction to email response
- Machine translation
- Sentiment analysis
- …
- Similar ideas applied to graphs, electronic health records, relational data, etc.