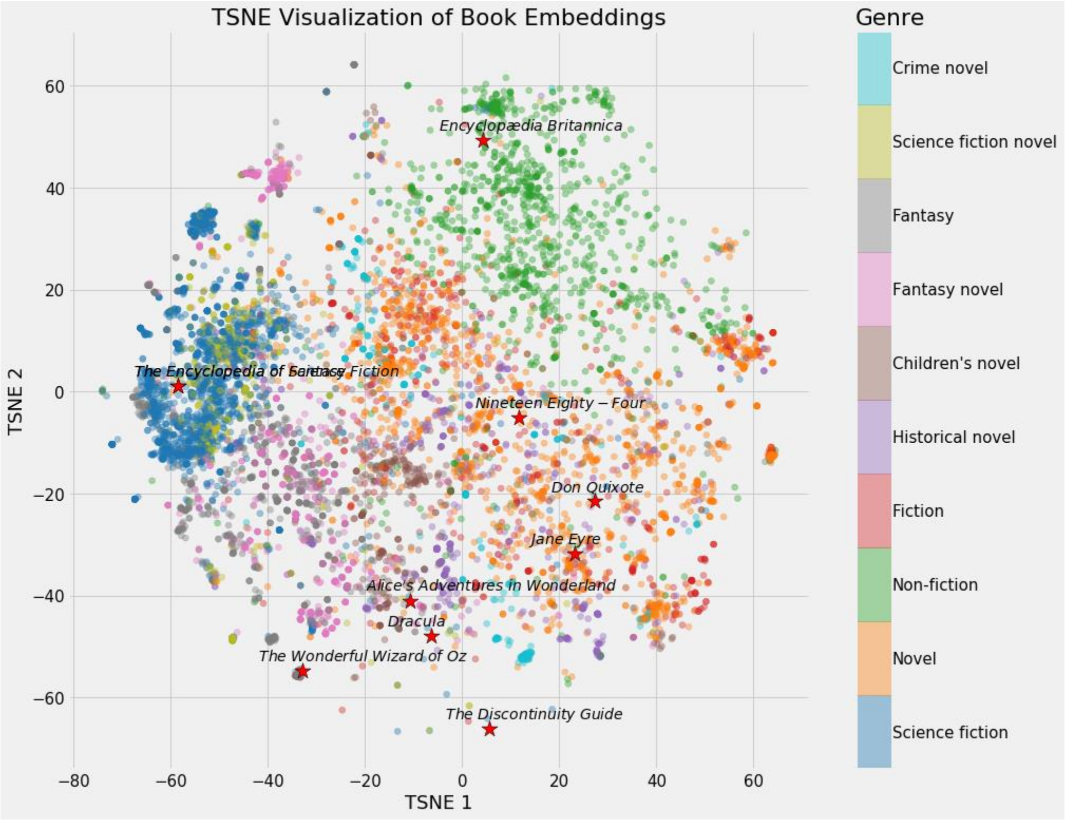
University of Ljubljana, Faculty of Computer and Information Science

Static neural embeddings



Prof Dr Marko Robnik-Šikonja

Natural Language Processing, Edition 2023

Contents

- Static neural language models
- word2vec word representation
- cross-lingual embeddings
- properties of dense embeddings
- doc2vec document representation

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

Vector Representation of Words

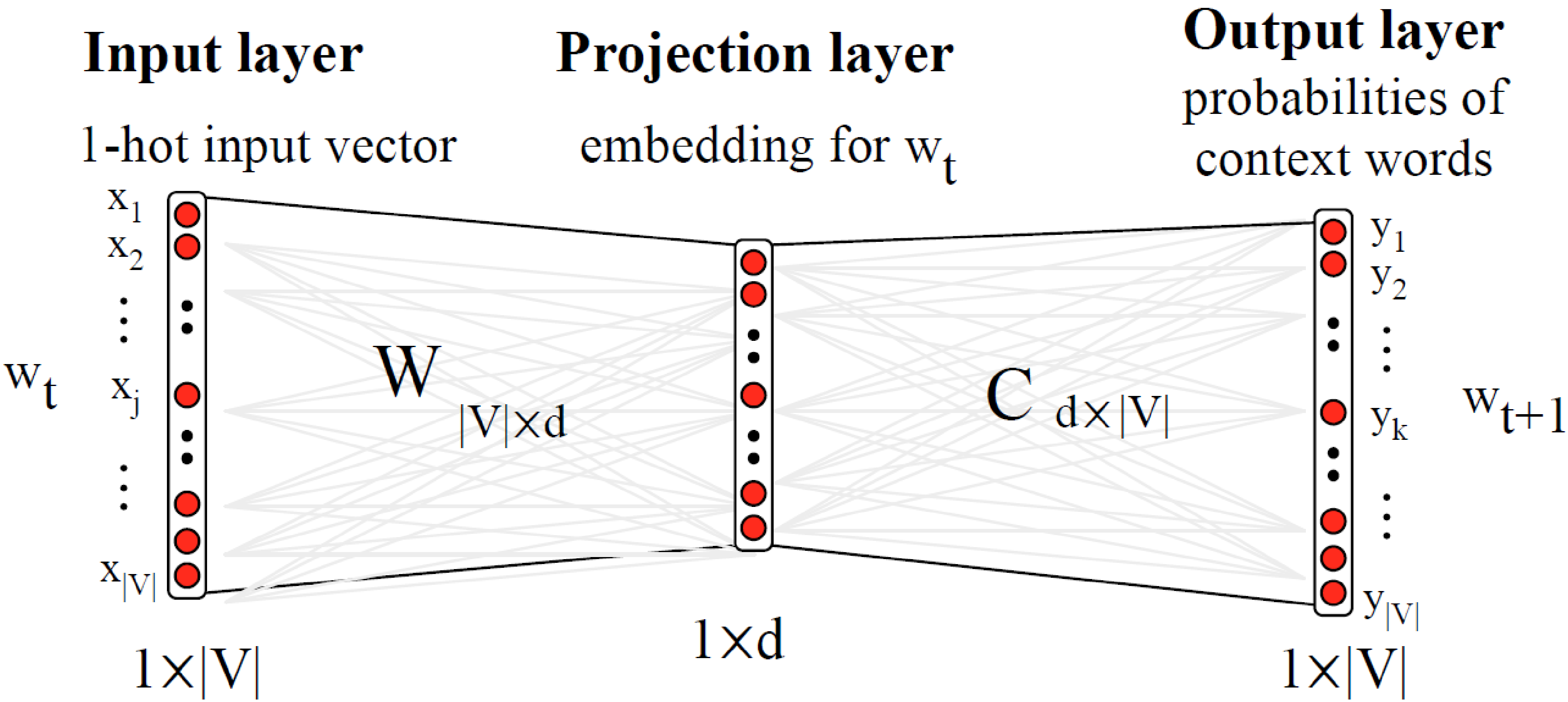
- Vector space models represent (embed) words in a continuous vector space
  - Theoretical foundation in Linguistics: Distributional Hypothesis
    - Words with similar meanings will occur with similar neighbors if enough text material is available (Rubenstein et al. 1967).
- Approaches that leverage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

Static 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
- Why these embeddings are called static?
  - a single fixed representation of a word
  - does not take into account polysemy of words (mixing all meanings)
  - does not take into account that the meaning of words may slightly change based on the context
- We will deal with contextual embeddings, such as ELMo and BERT, later

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

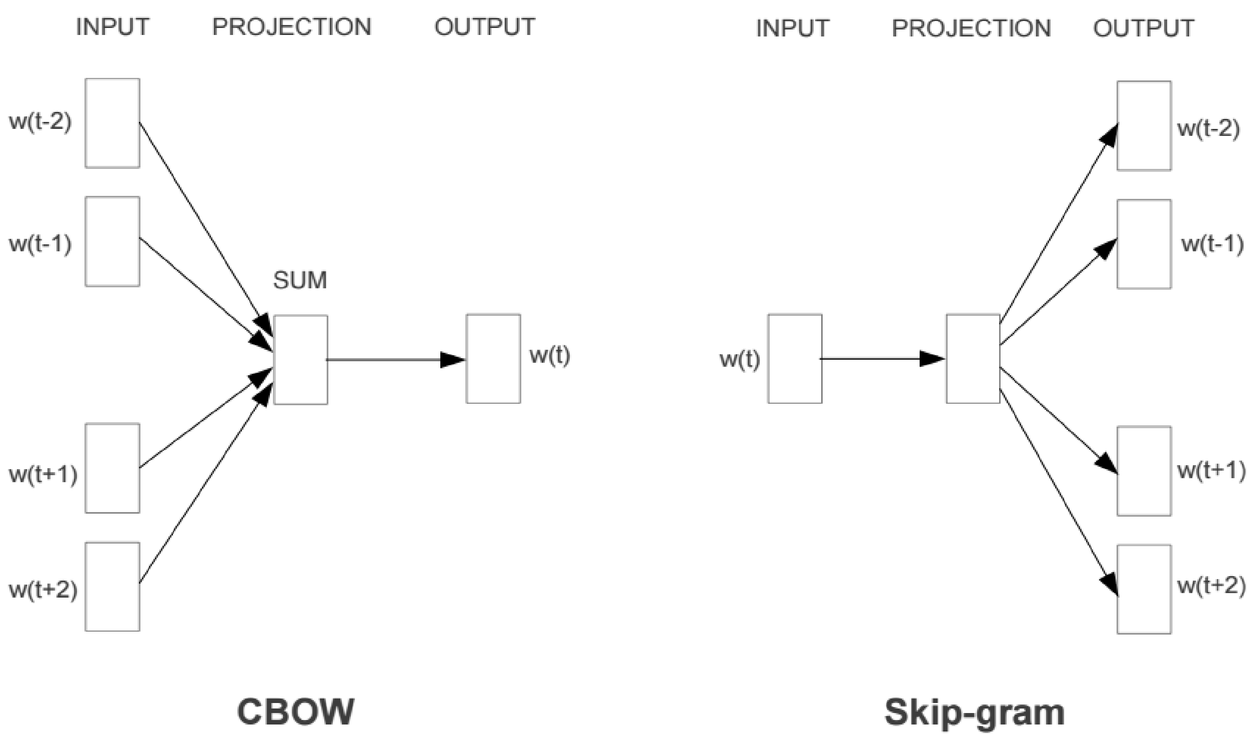
- Two variants:
  - CBOW: Predict target from the bag of words context
  - Skipgram: Predict context words from target (position-independent)

- 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

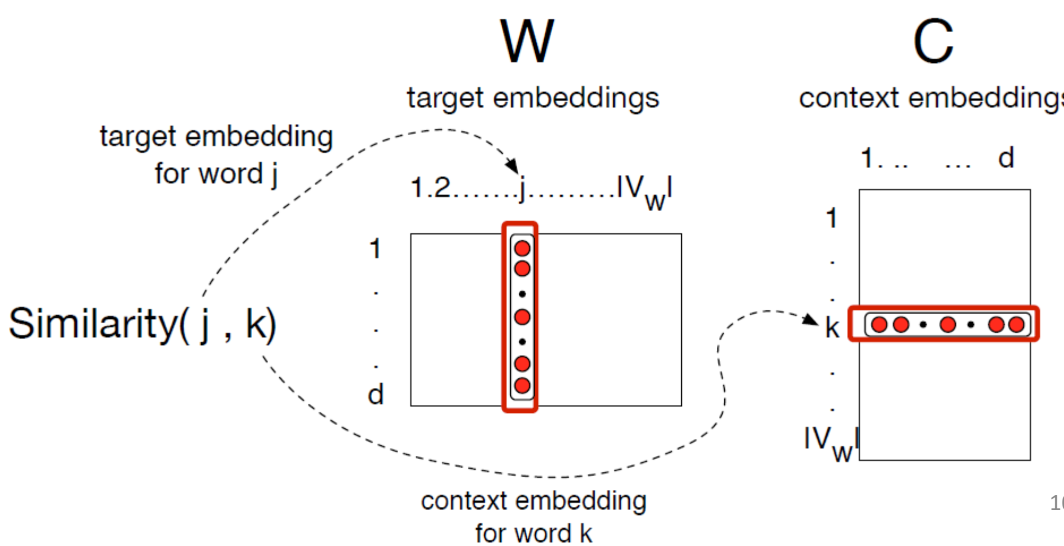
w_{-2}	w_{-1}	w_0	w_1	w_2
?	?	Network	?	?

w_{-2}	w_{-1}	w_0	w_1	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 the logistic regression to train a classifier that distinguishes 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:
  - $P(+|t,c)$
  - $P(-|t,c) = 1 - P(+|t,c)$

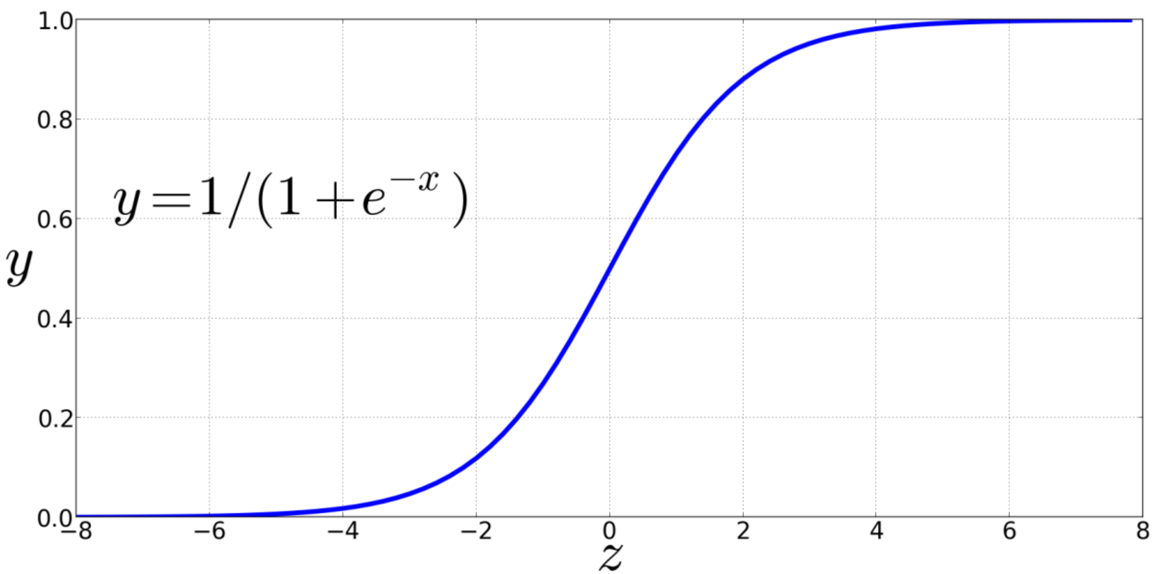
How to compute p(+|t,c)?

- 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

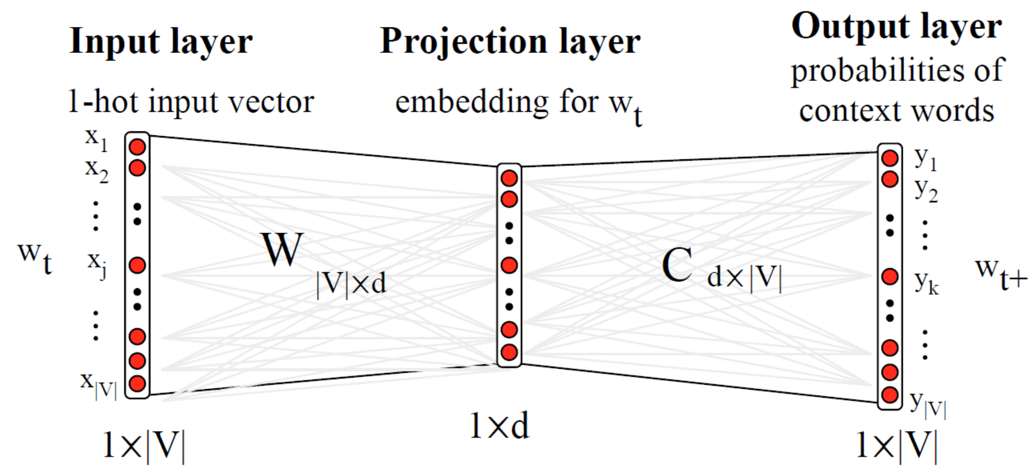
- Could pick w according to their unigram frequency P(w)
- More common to chose according to $p_\alpha(w)$

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

- 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:

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

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

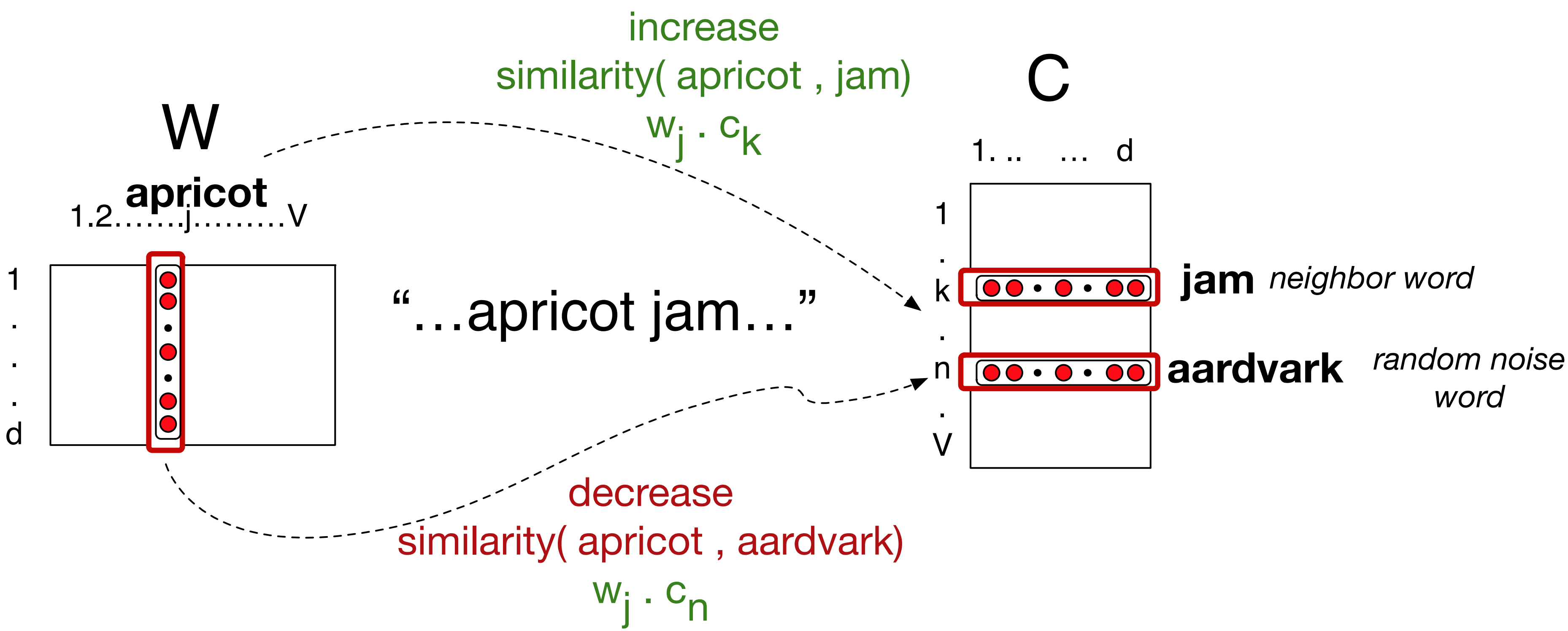
$$\sum_{(t,c)\in+} logP(+|t,c) + \sum_{(t,c)\in-} logP(-|t,c)$$

- 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)
$w_j . c_k$
C
W
apricot
1.2…….j………V
1. ..   … d
1
k
n
V
1
d
jam neighbor word
"…apricot jam…"
aardvark random noise word
decrease
similarity(apricot , aardvark)
$w_j . c_n$

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

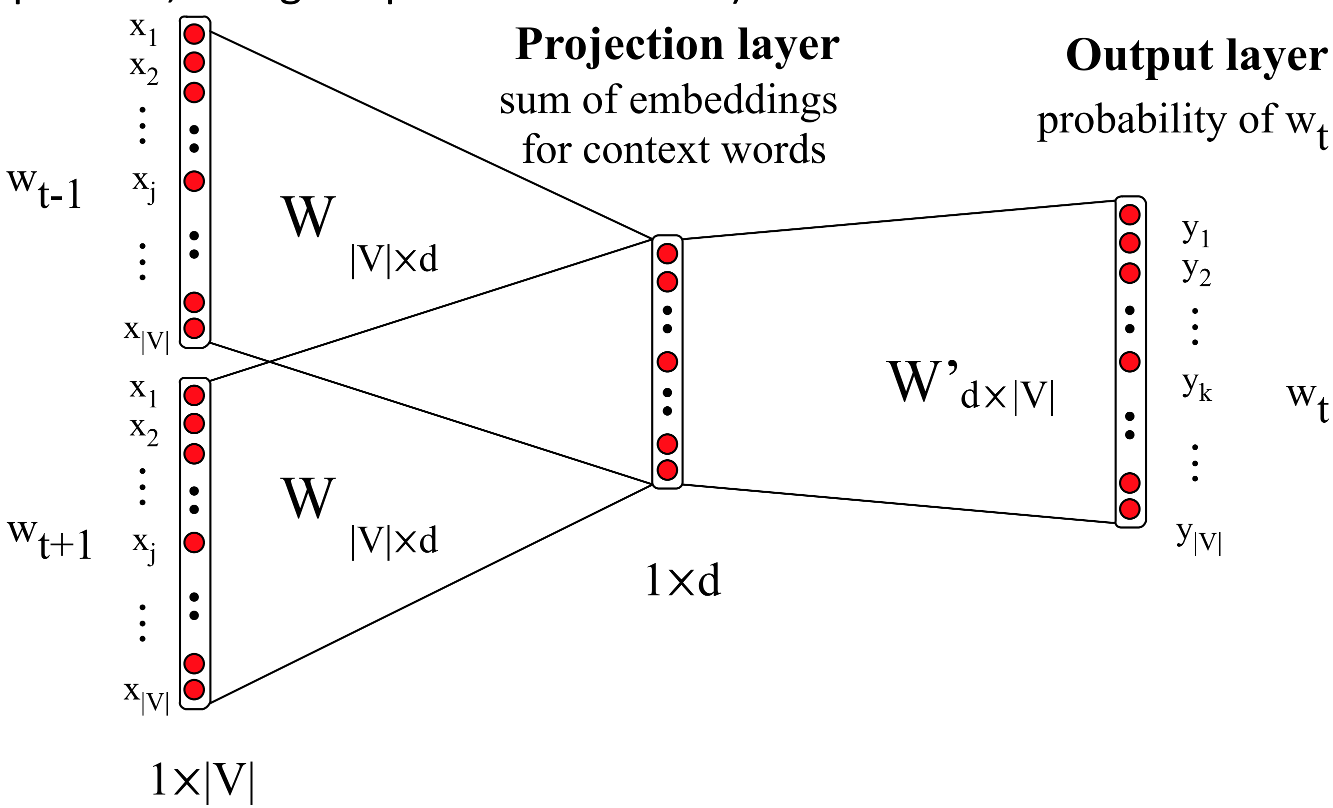
- Start with V random 300-dimensional vectors as initial embeddings
- Use logistic regression,
  - Take a corpus and take pairs of words that co-occur as positive examples
  - Take pairs of words that don't co-occur as negative examples
  - Train the classifier to distinguish these by slowly adjusting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

CBOW (Continuous Bag of Words) schema

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

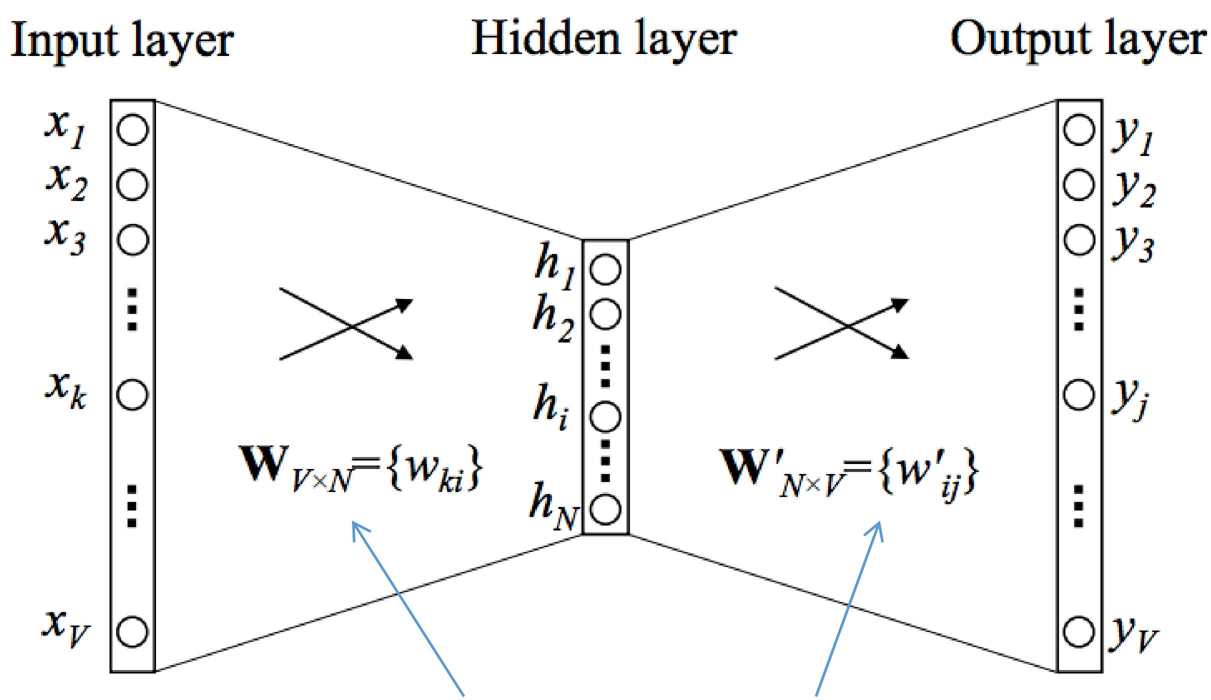
1-hot input vectors
for each context word
(in practice, a single input vector is used)



Details of 1 word context CBOW

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

One-hot vector



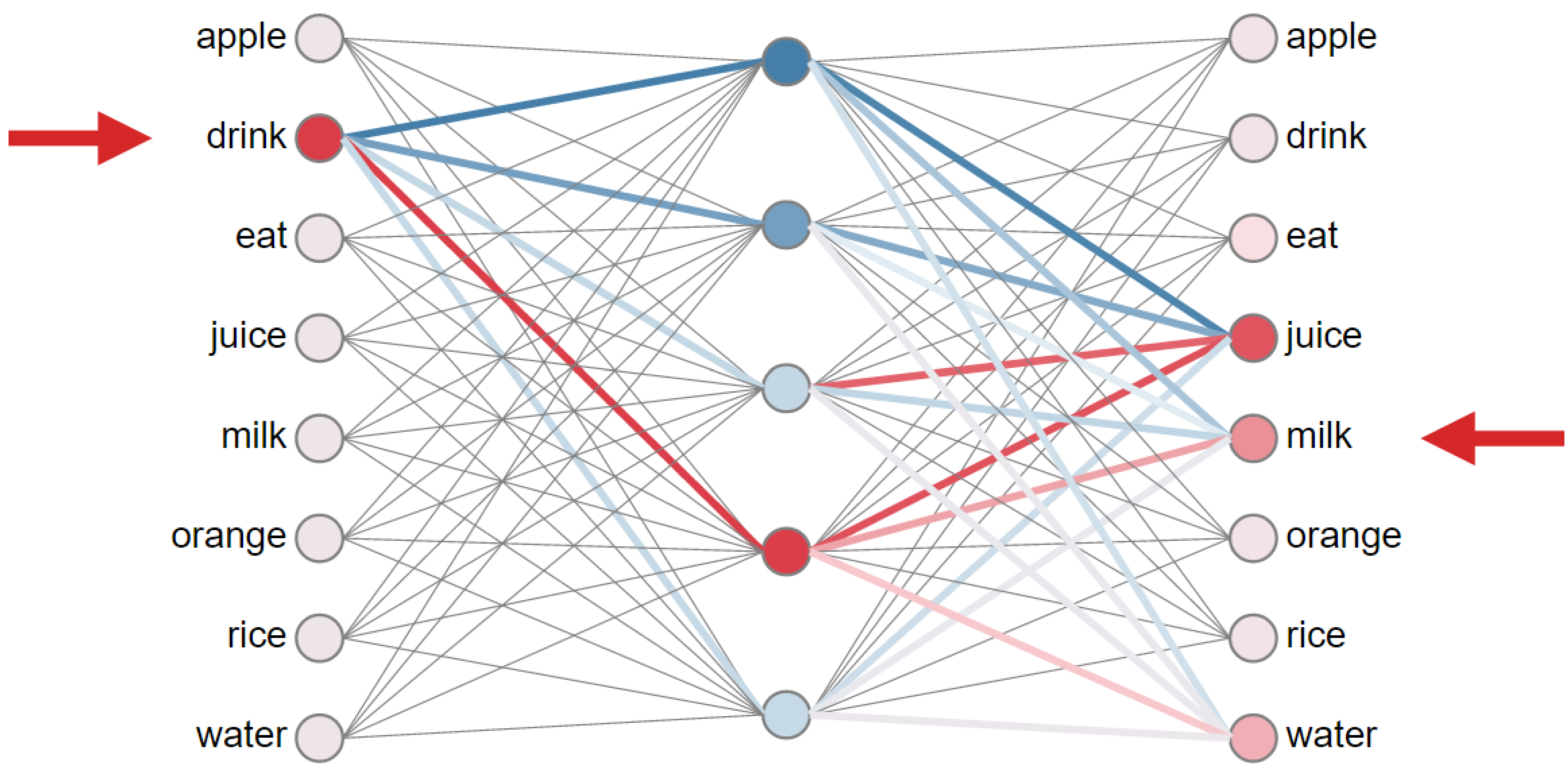
Attempts to predict one-hot vector

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/
- To learn 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

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

<table>
<tr><th>Model</th><th>Advantages</th><th>Disadvantages</th></tr>
<tr><td>Continuous BOW</td><td>• Mediocre semantic accuracy
• Absence in papers; unpopular in practice</td><td rowspan="2">• Ignores global vocabulary information
• Does not handle out-of-vocabulary words</td></tr>
<tr><td>Skipgram</td><td>• Good semantic accuracy
• Pre-trained models available</td></tr>
<tr><td>GloVe</td><td>• Uses global information of vocabulary
• Captures local and global context of words
• Good syntactic and semantic accuracy
• Pre-trained models available</td><td>• Does not handle out-of-vocabulary words</td></tr>
<tr><td>FastText</td><td>• Handles out-of-vocabulary words
• Good at syntactic tasks
• Pre-trained models available
• Available aligned word vectors</td><td>• Takes 1.5x longer to train than skipgram</td></tr>
</table>

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: μ 7.68 σ 0.80**
Context1	**Context1: μ 6.49 σ 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: μ 7.73 σ 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., Robnik-Šikonja, M., Granroth-Wilding, M. and Vaik, K., 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

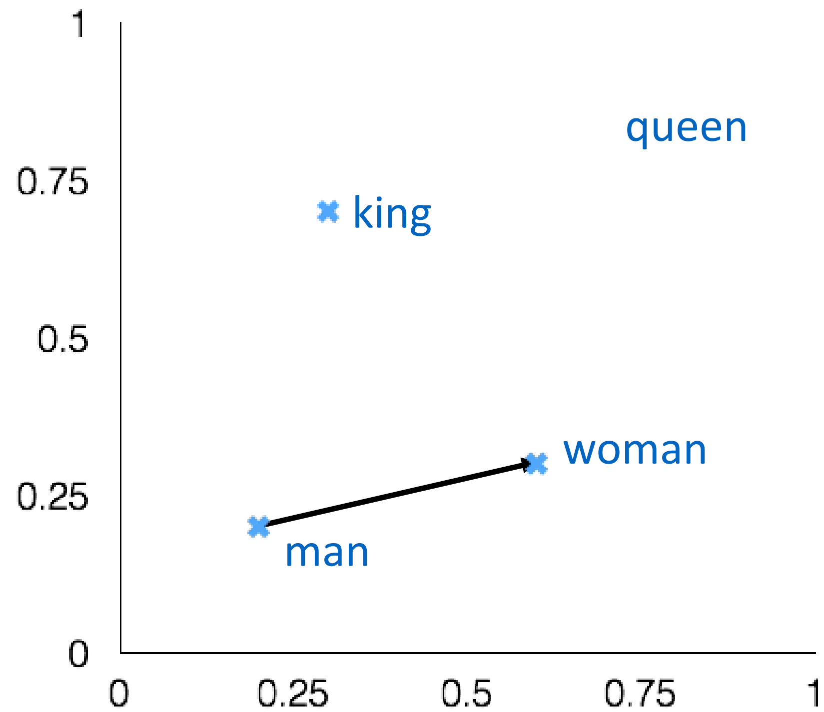
Test for linear relationships, examined by Mikolov et al.

a:b :: c:?

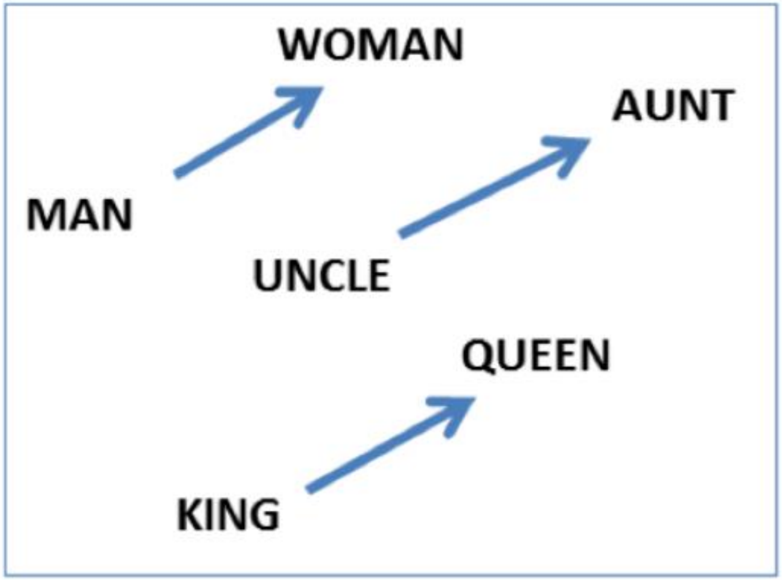
$$d = \arg\max_x \frac{(w_b - w_a + w_c)^T w_x}{||w_b - w_a + w_c||}$$

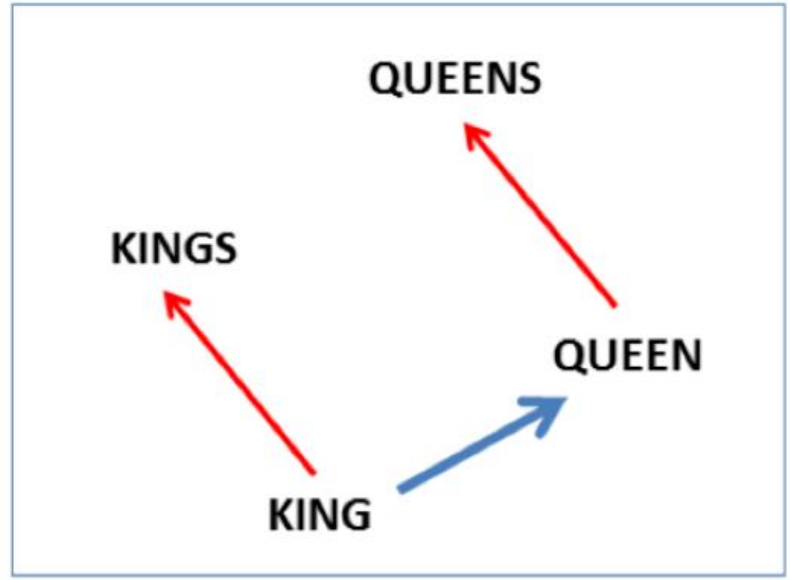
man:woman :: king:?

+	king	[0.30 0.70]
−	man	[0.20 0.20]
+	woman	[0.60 0.30]
	queen	[0.70 0.80]



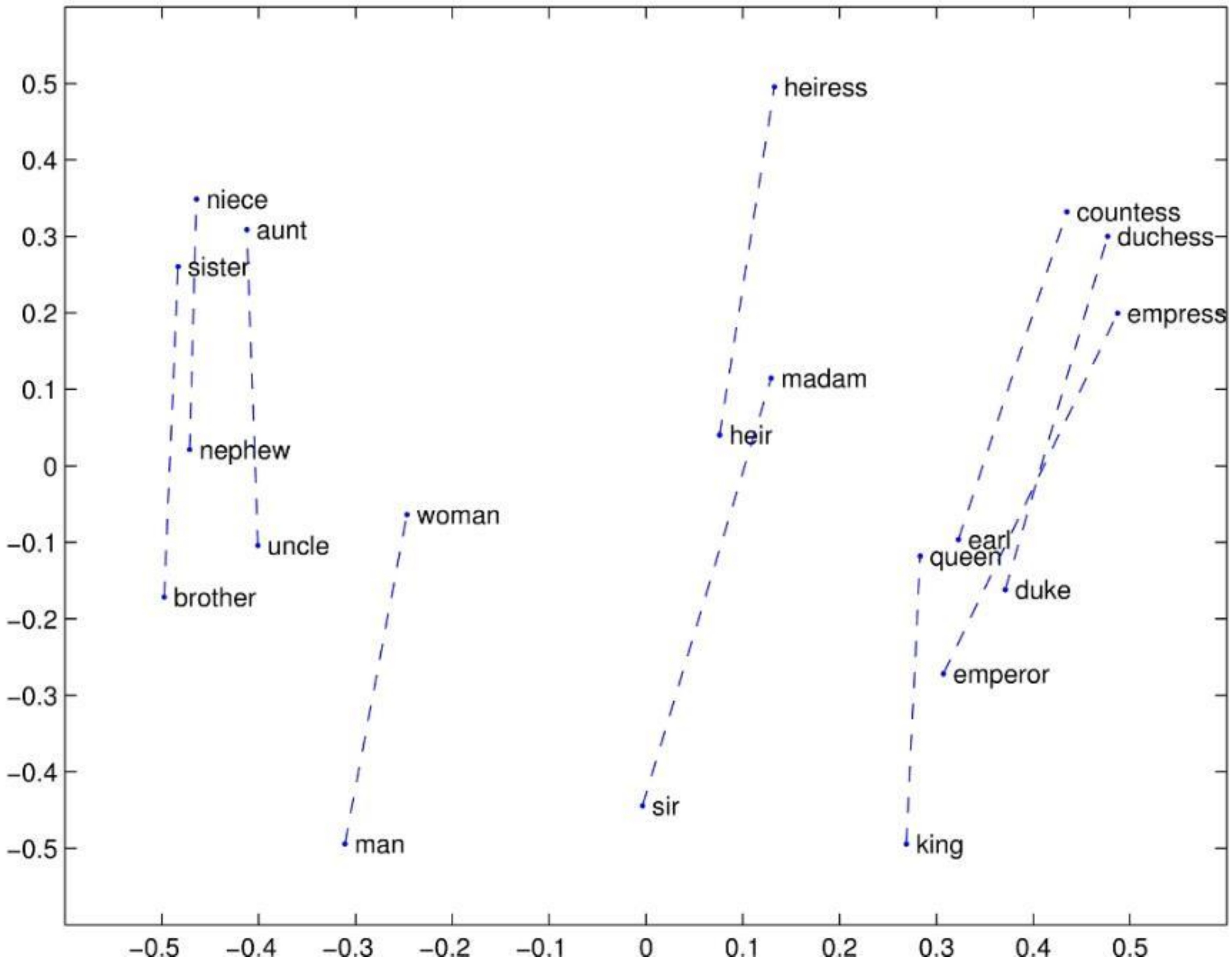
Relational similarity



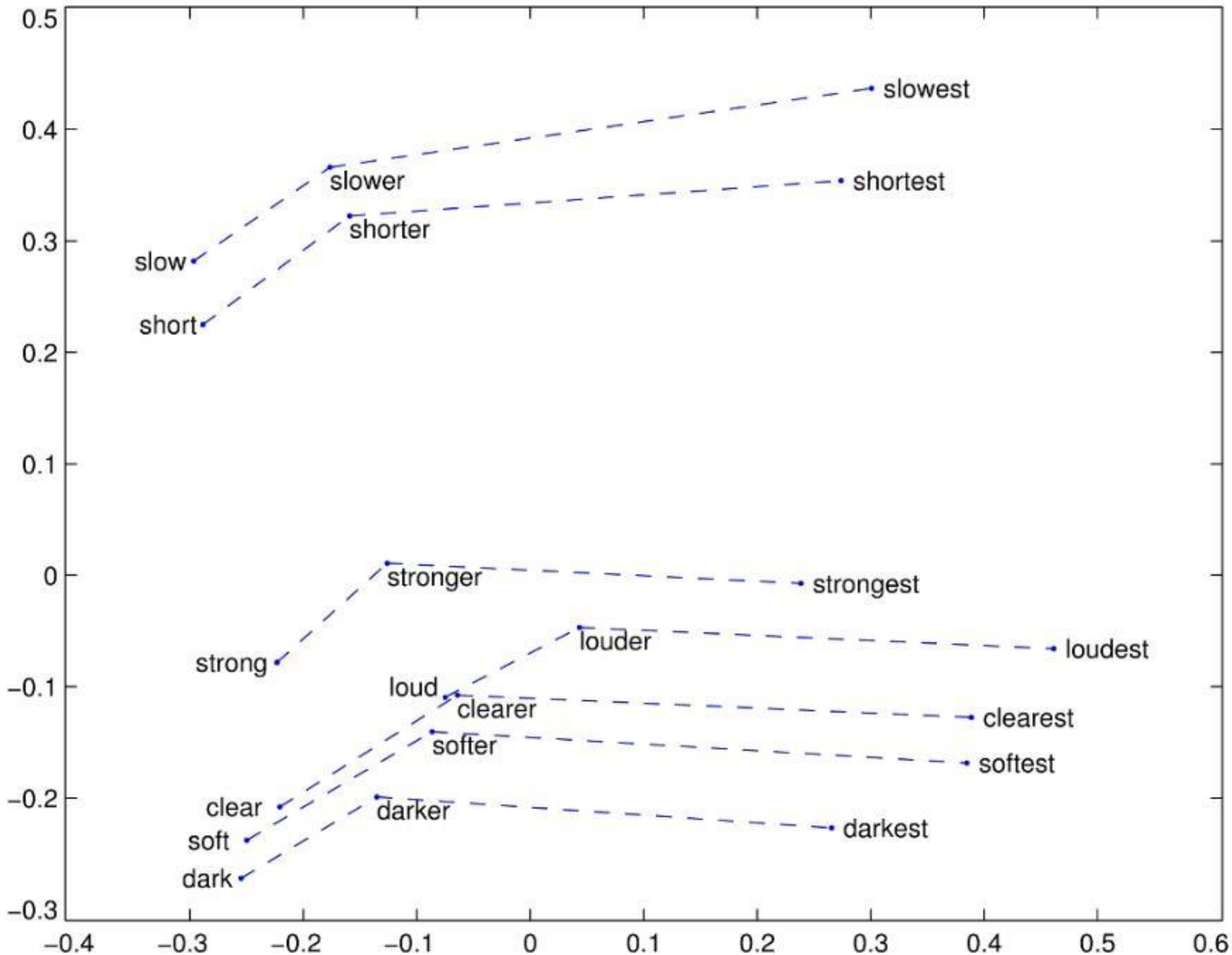


vector(*'king'*) - vector(*'man'*) + vector(*'woman'*) ≈ vector('queen')

vector(*'Paris'*) - vector(*'France'*) + vector(*'Italy'*) ≈ vector('Rome')



heiress
niece
aunt
sister
countess
duchess
empress
madam
heir
nephew
woman
uncle
earl
queen
brother
duke
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
soft
dark
darker
darkest
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

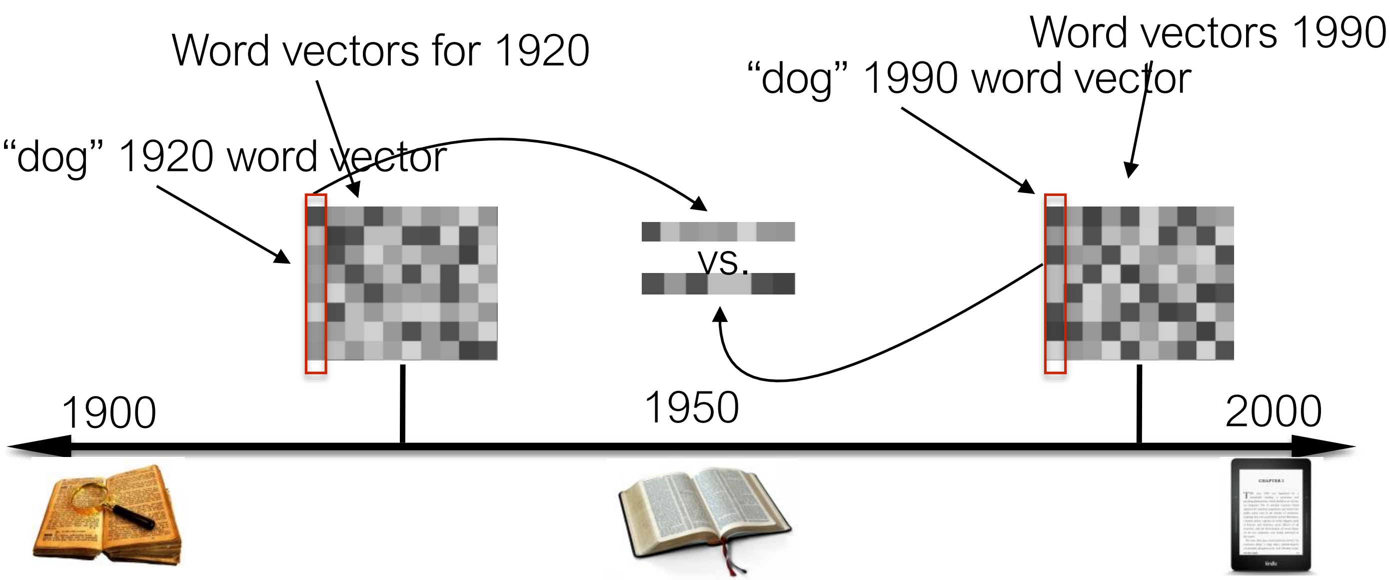
Embeddings visualization

- https://projector.tensorflow.org/

Embeddings can help study word history

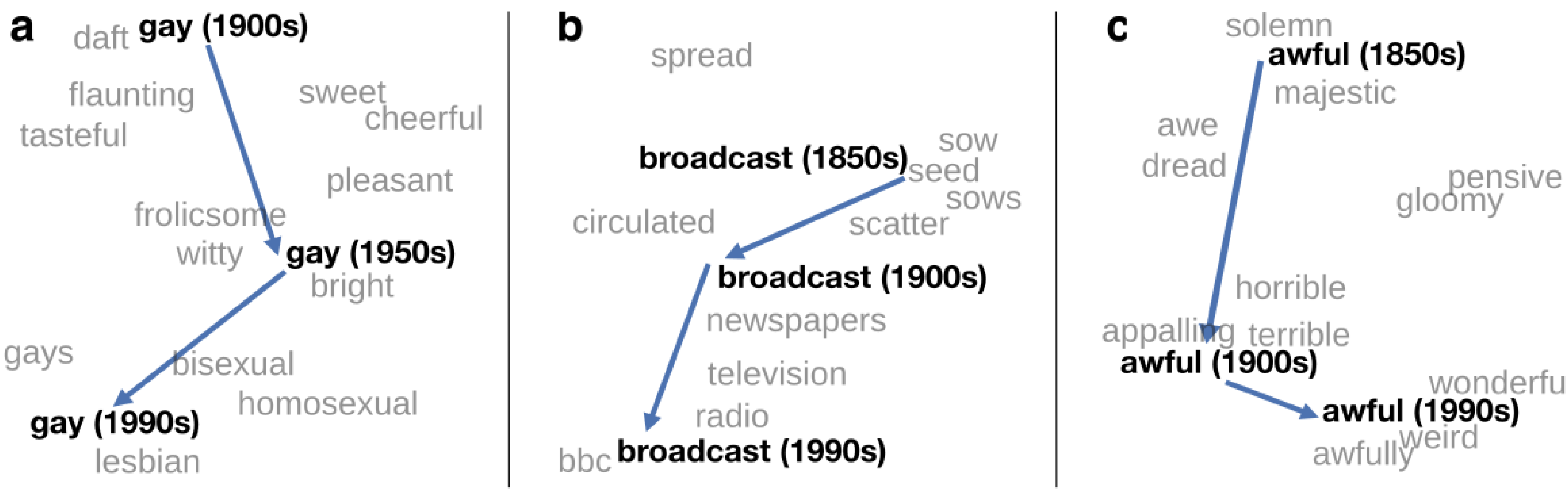
- Train embeddings on old books to study changes in word meaning

Diachronic word embeddings for studying language change



Visualizing changes

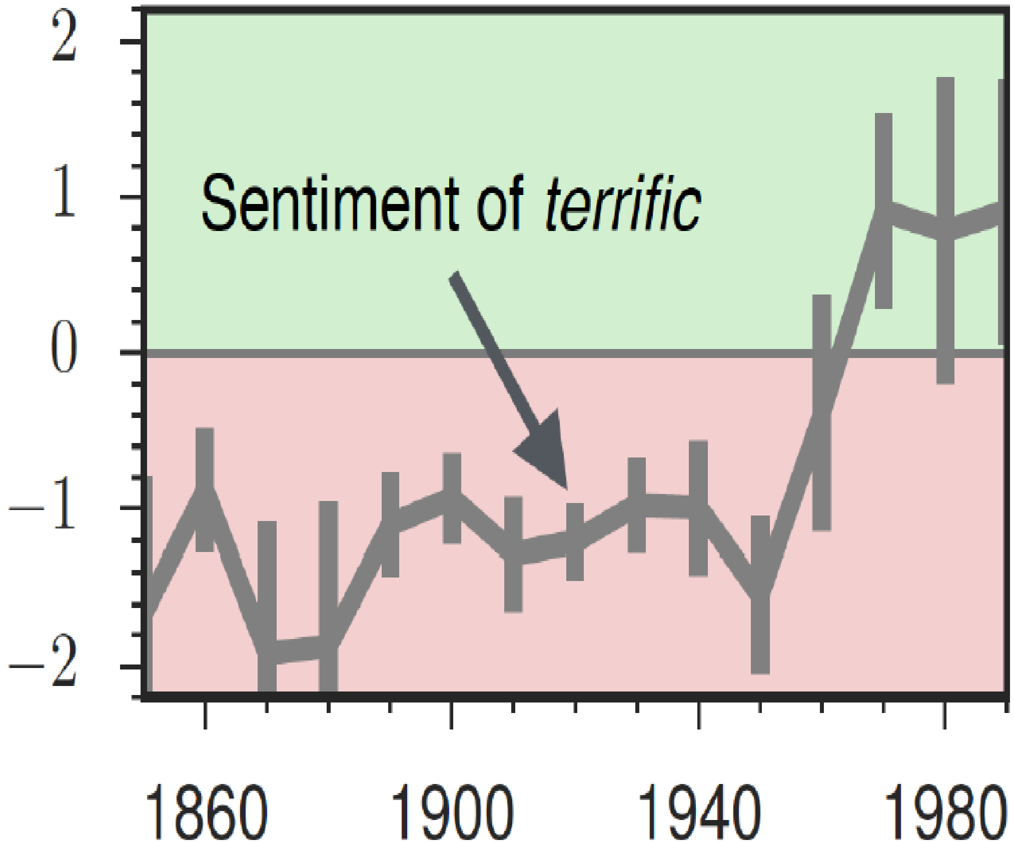
Project 300 dimensions down into 2



~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

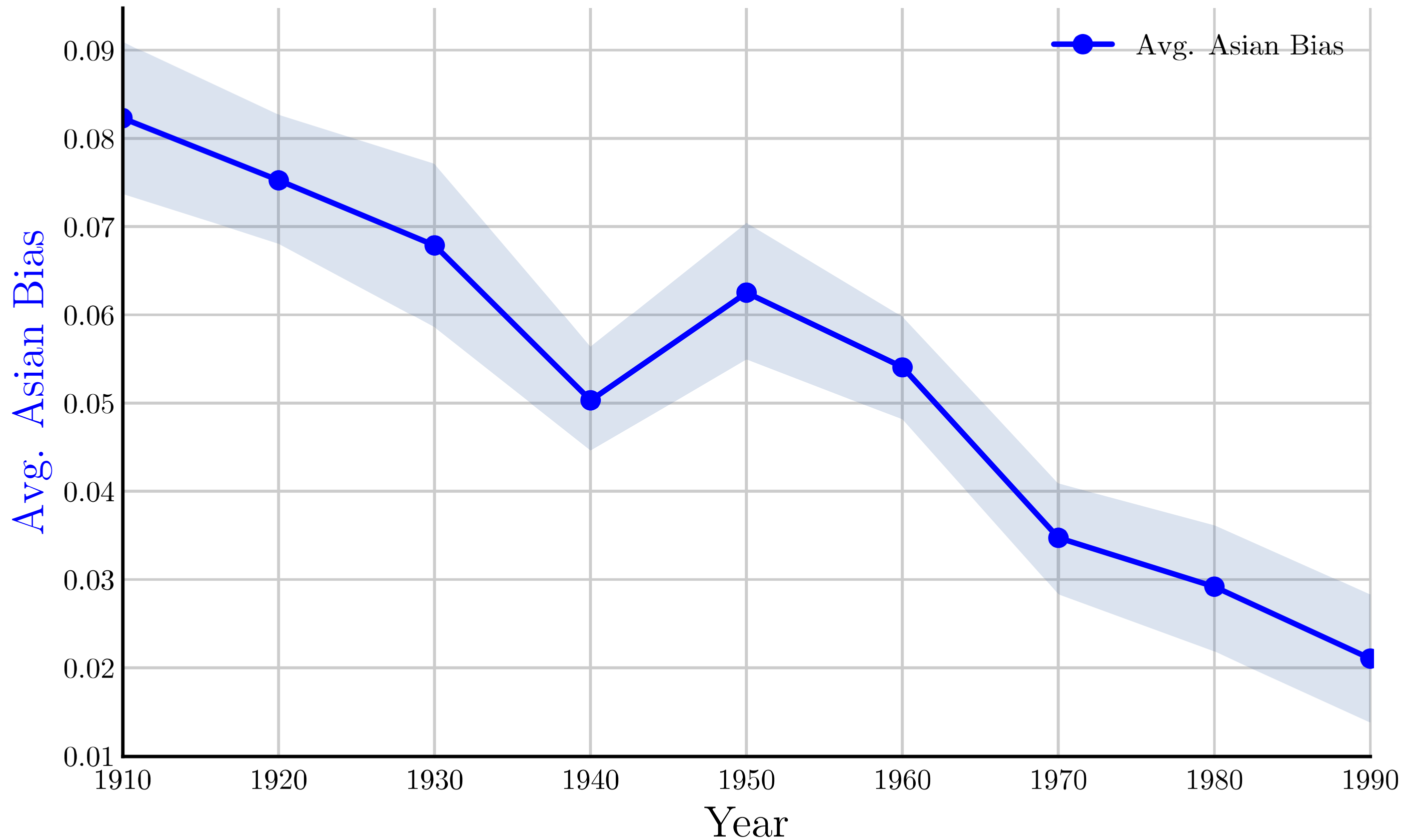
Embeddings reflect cultural bias

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

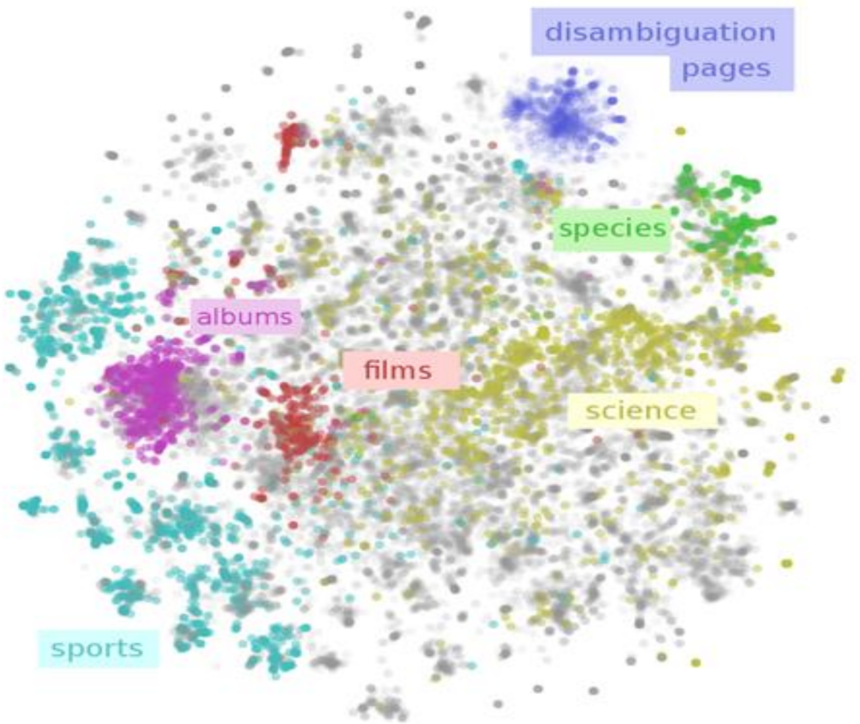
1910	1950	1990
Irresponsible	Disorganized	Inhibited
Envious	Outrageous	Passive
Barbaric	Pompous	Dissolute
Aggressive	Unstable	Haughty
Transparent	Effeminate	Complacent
Monstrous	Unprincipled	Forceful
Hateful	Venomous	Fixed
Cruel	Disobedient	Active
Greedy	Predatory	Sensitive
Bizarre	Boisterous	Hearty

Debiasing

- 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.
- hard to remove all biases

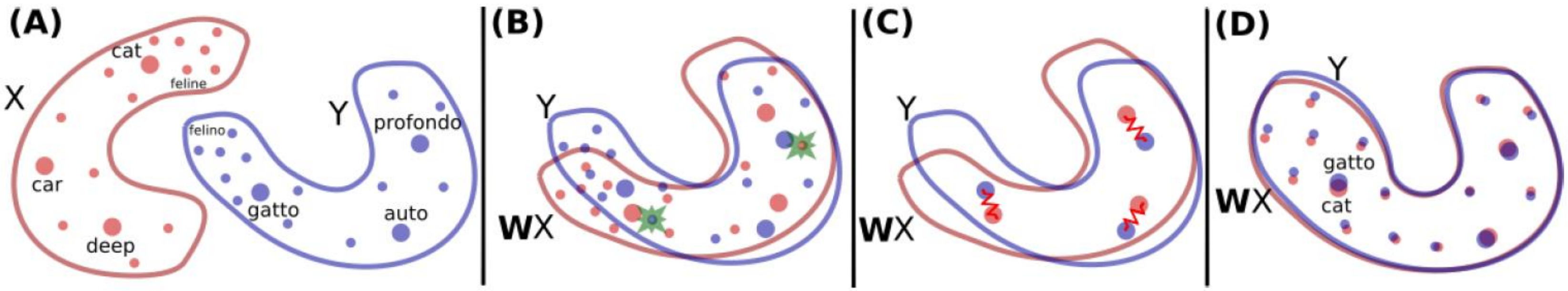
Cross-lingual embeddings

- embeddings are trained on monolingual resources
- words of one language form a cloud in high dimensional space
- clouds for different languages can be aligned
- $W_1 S \approx W_2 E$ or $W_1 S \approx E$



Cross-lingual embeddings

- alignment of different word clouds

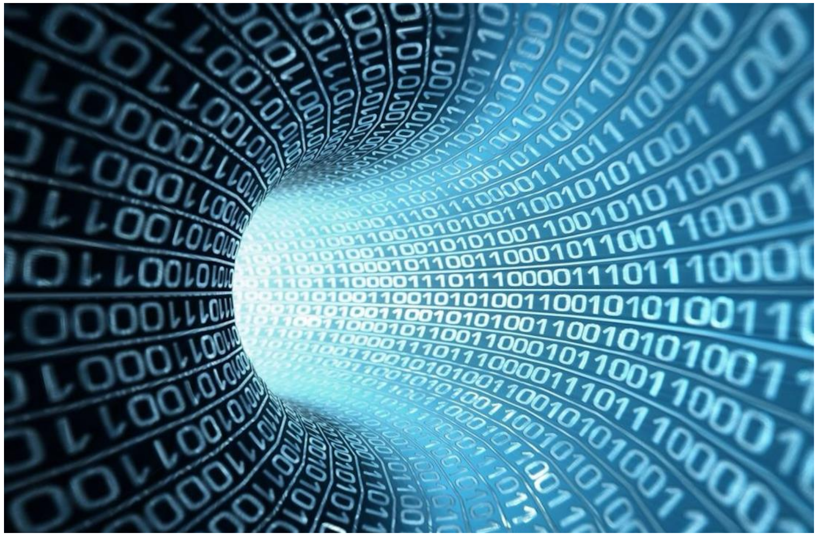


- in unsupervised or supervised way

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

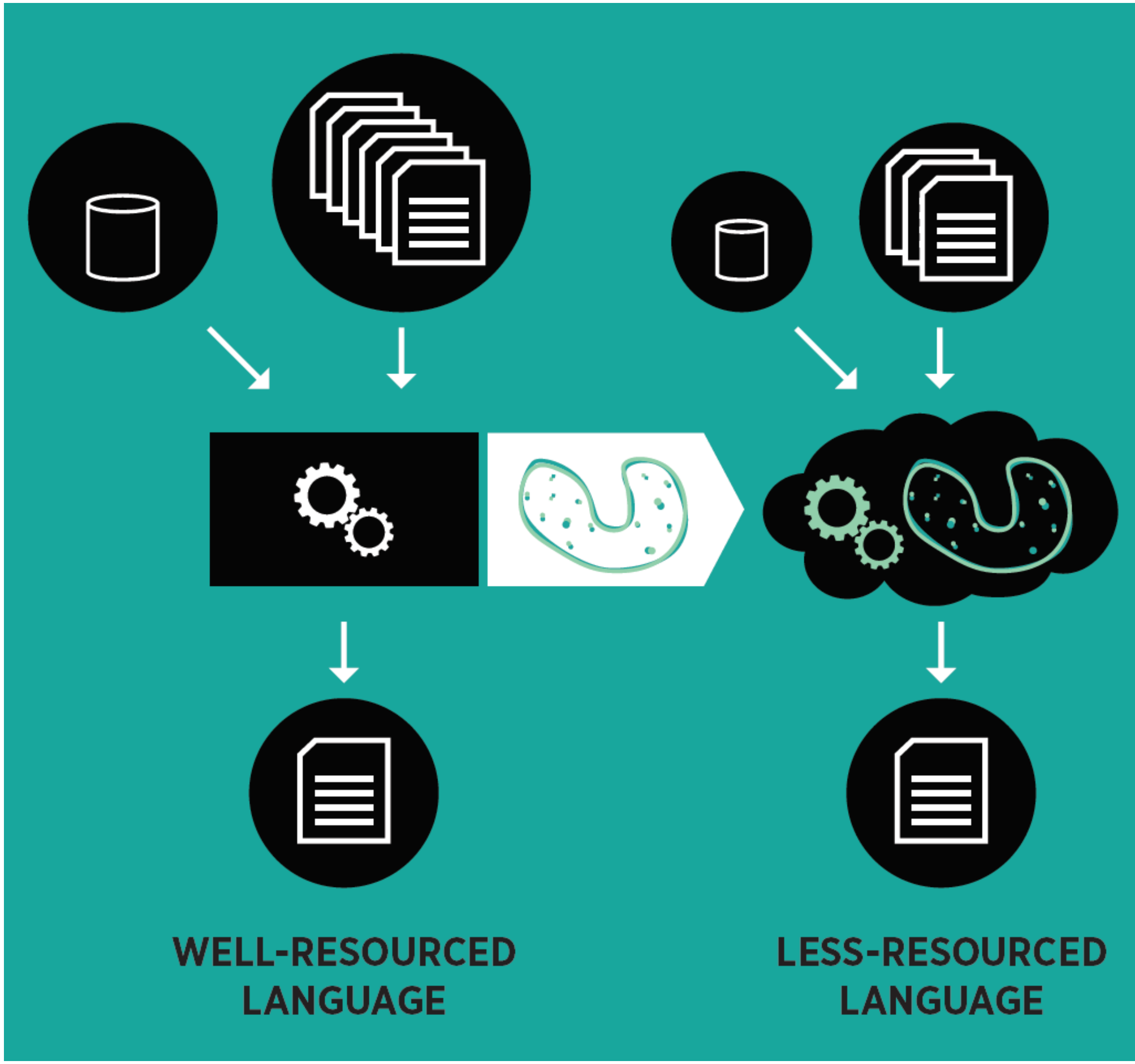
- bilingual and multilingual resources can provide anchoring points for alignment of different word clouds
- alignment of contextual embeddings

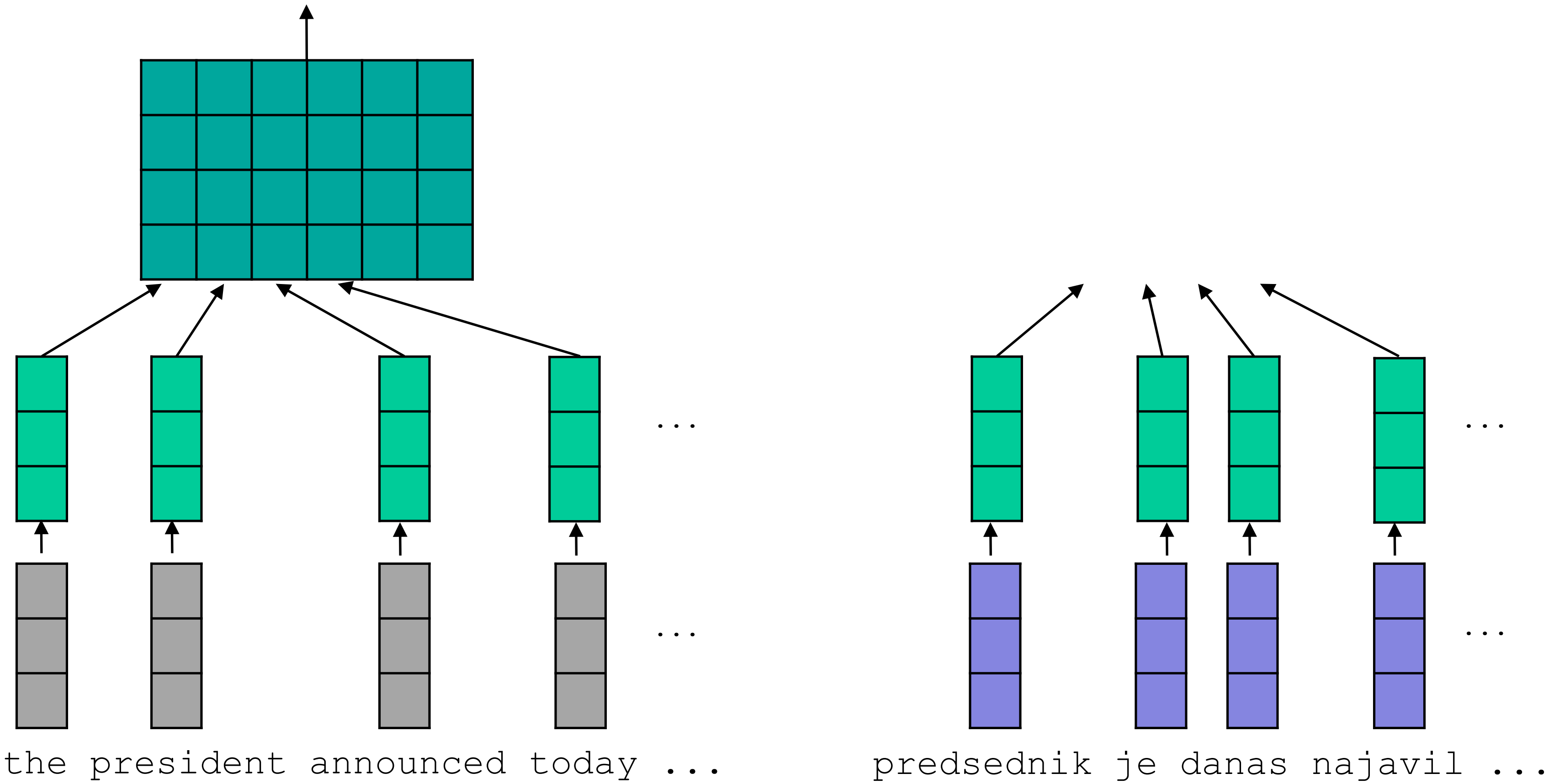


Artetxe, M. and Schwenk, H., 2018. Massively Multilingual Sentence Embeddings for Zero-Shot Cross-Lingual Transfer and Beyond. *ArXiv preprint arXiv:1812.10464*.

Cross-lingual transfer with embeddings

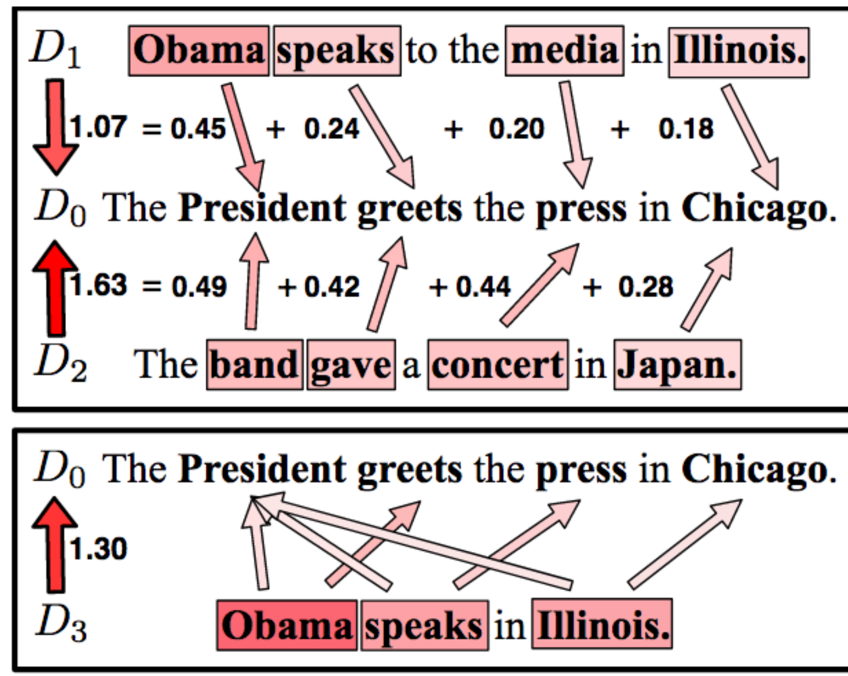
- Transfer of tools trained on mono-lingual resources




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

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

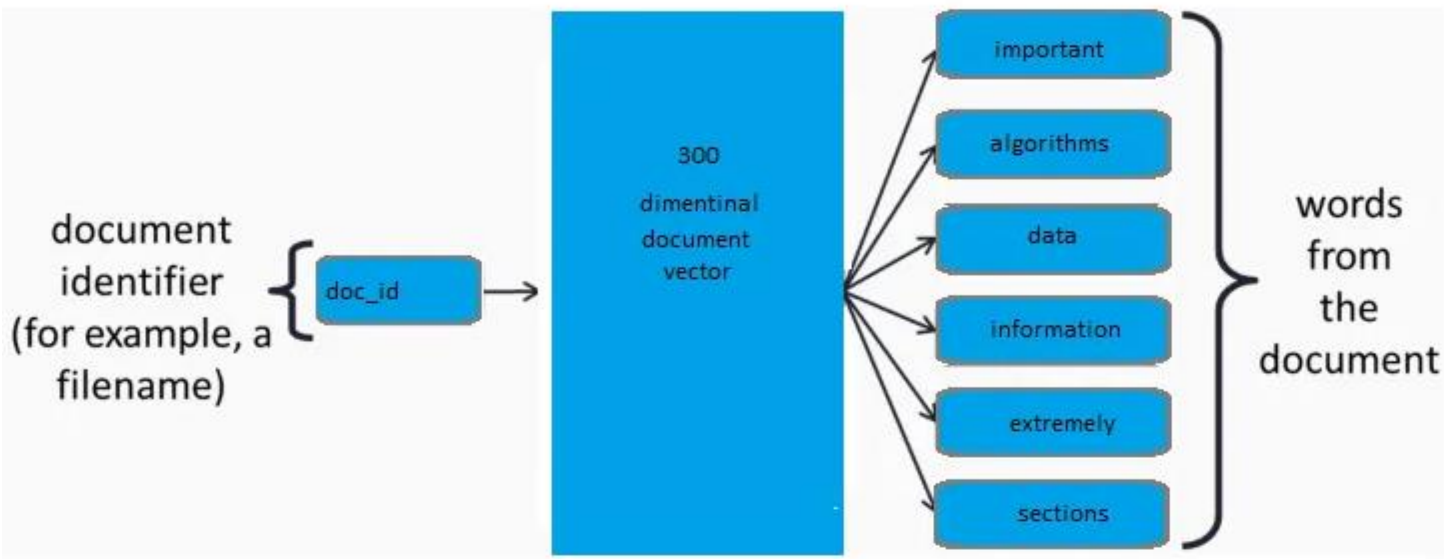


Conclusion of static dense word embeddings

- **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

Doc2vec representation

- **Doc2Vec** is an extension of **Word2vec** that encodes entire documents (or sentence, paragraph, article, etc.)
- The idea is to use Document ID vector as a context and use it to predict words from the document



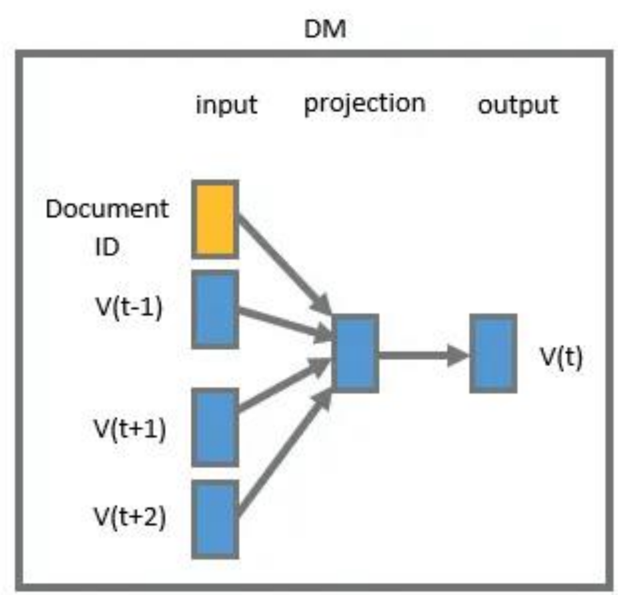
doc2vec images by Manish Nayak

Two doc2vec flavours

- similar to word2vec CBOW:
  Distributed Memory Model Of Paragraph Vectors (PV-DM)
- similar to word2vec Skipgram:
  Paragraph Vector With A Distributed Bag Of Words (PVDBOW)
- both extend the word context with the paragraph vector
- word vectors are the same (usually word2vec vectors), but paragraph vectors are unique for each text unit

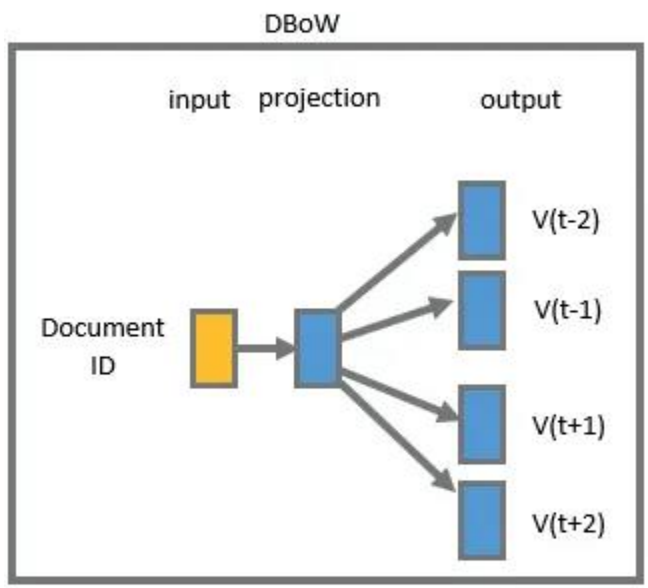
Doc2vec PV-DM

- Paragraph Vector - Distributed Memory doc2vec resembles the CBOW word2vec
- predicts a target word given the context words and additional paragraph ID
- The paragraph token can be thought of as another word. It acts as a memory that remembers what is missing from the current context – or the topic of the paragraph.
- the projection is either concatenation or averaging of input vectors
- for prediction layer, doc2vec usey softmax, i.e. logistic regression



Doc2vec PV-DBOW

- Paragraph Vector - Distributed Bag of Words doc2vec resembles the word2vec Skipgram model
- instead of using the target word as the input, it uses the document ID as the input and tries to predict randomly sampled words from the document.



Doc2vec for new documents

- For a new document, a doc2vec model needs a bit of additional training to construct a paragraph vector that will predict the words in the new document
- at this learning step, word vector and prediction weights are fixed

Doc2vec properties

- doc2vec learns which words go together in the document, i.e. which words are specific for a document
- Le & Mikolov recommend merging PV-DM and Pv-DBOW vectors
- doc2vec vectors work well in finding similar documents and other document level tasks
- Advantage over bag-of-words document representation:
  - doc2vwc includes semantics of the words from word2vec
  - paragraph vectors in PV-DM take into consideration the word order

Le, Quoc, and Tomaš Mikolov. "Distributed representations of sentences and documents." In *International conference on machine learning*, pp. 1188-1196. PMLR, 2014.

Trend: 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.