FAKULTÄT
FÜR !NFORMATIK

Faculty of Informatics

Statistical Semantics with Dense Vectors

Word Representation Methods from Counting to Predicting

Navid Rekabsaz
rekabsaz@ifs.tuwien.ac.at

keystone

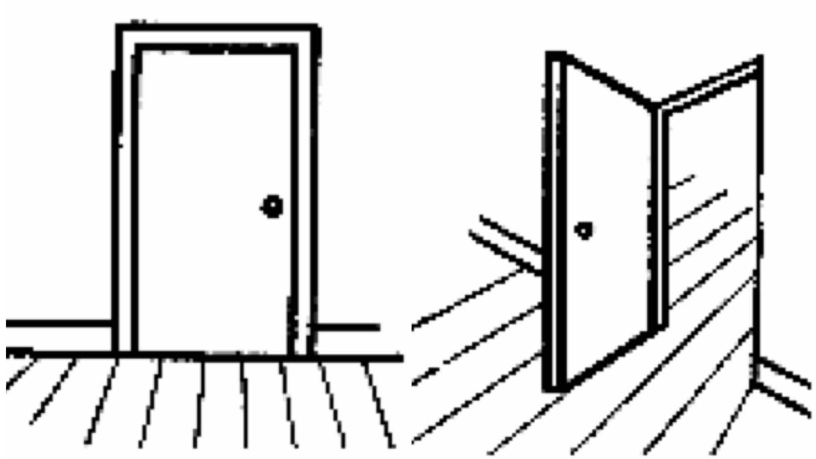
3rd KEYSTONE Training School
Keyword search in Big Linked Data

24/Aug/2017 Vienna, Austria

ifs

Semantics

- Understanding the semantics in language is a fundamental topic in text/language processing and has roots in linguistics, psychology, and philosophy
  - What is the meaning of a word? What does it convey?
  - What is the conceptual/semantical relation of two words?
  - Which words are similar to each other?



(a) The door is closed. (b) The door is open.

ifs

Semantics

- Two computational approaches to semantics:

Knowledge base

Statistical (Data-oriented) methods

word2vec

Auto-encoder decoder

LSA

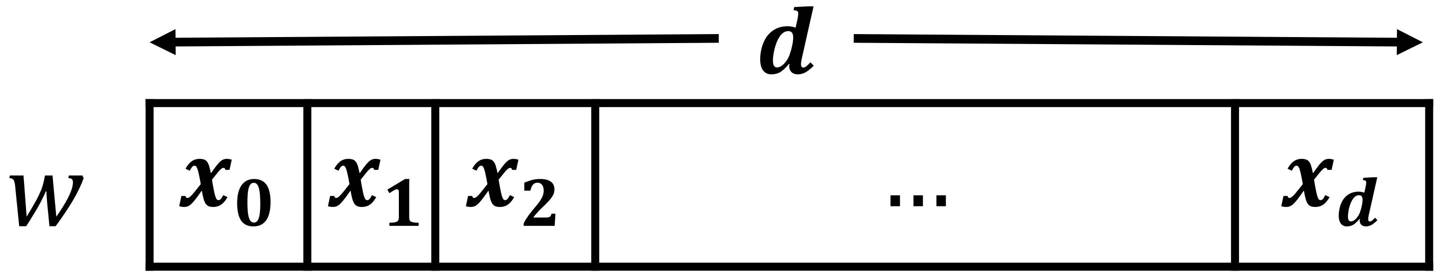
GloVe

RNN

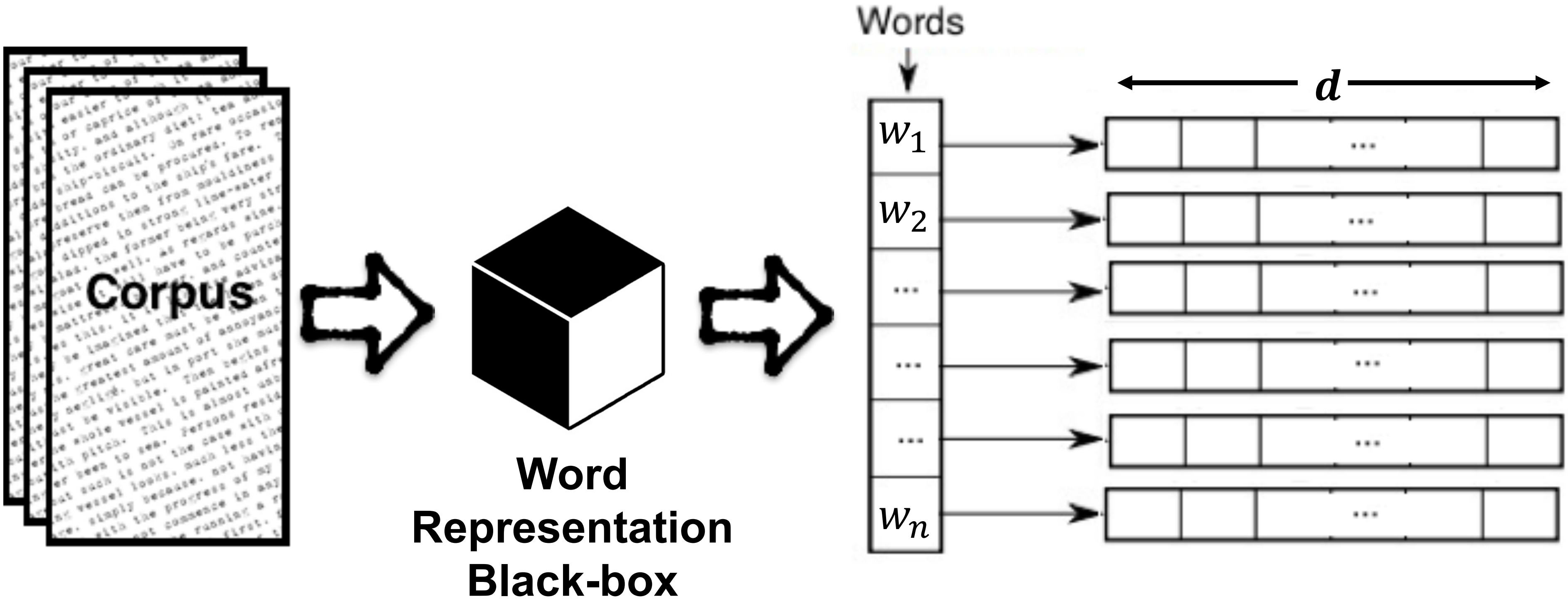
LSTM

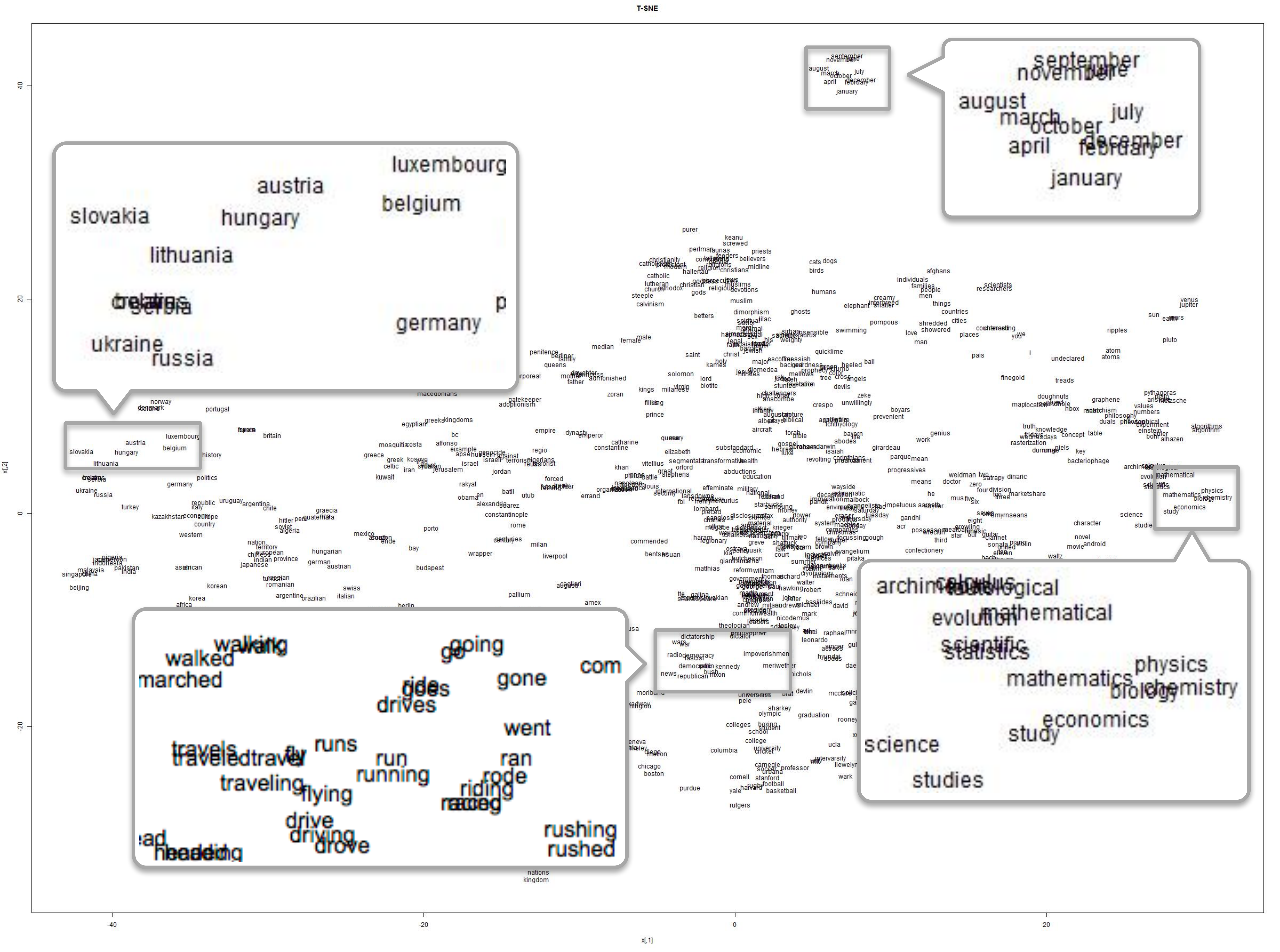
Statistical Semantics with Vectors

- A word is represented with a vector of *d* dimensions
- The vector aim to capture the semantics of the word
- Every dimension usually reflects a concept, but may or may not be interpretable

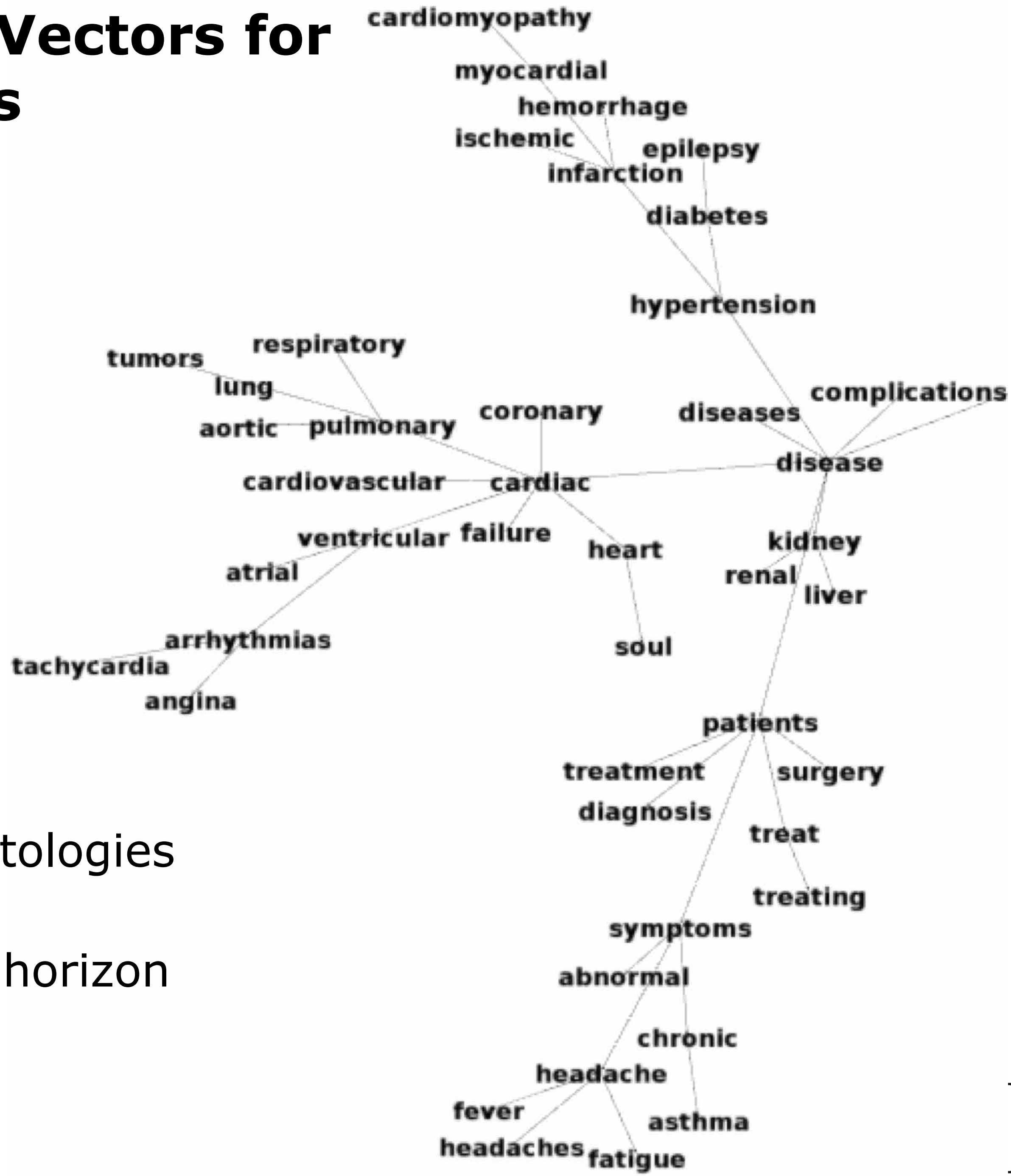


Statistical Semantics – From Corpus to Semantic Vectors




T-SNE
x[,1]
x[,2]
-40
-20
0
20
40
september
november
august
march
july
october
december
april
february
january
luxembourg
austria
belgium
slovakia
hungary
lithuania
germany
ukraine
russia
walked
marched
going
gone
drives
went
runs
run
running
ran
traveling
flying
riding
drive
rushing
rushed
mathematical
evolution
physics
mathematics
economics
study
science
studies

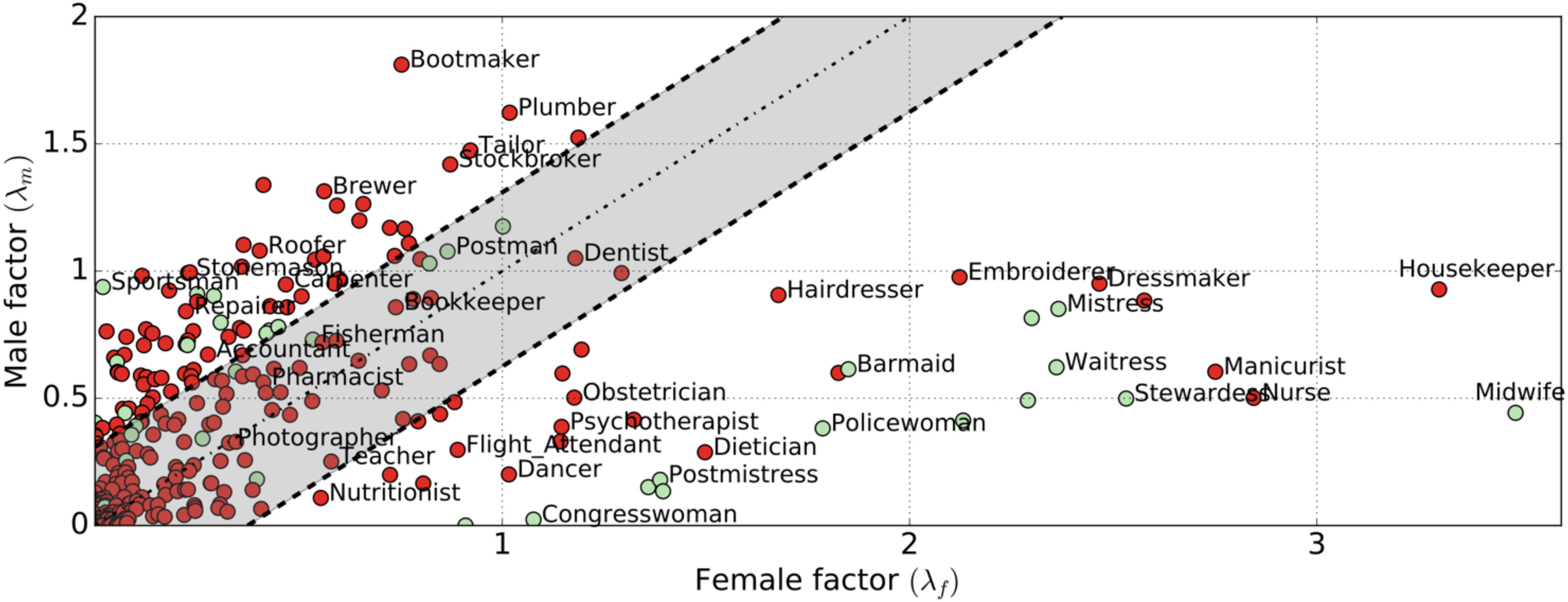
Semantic Vectors for Ontologies



- Enriching existing ontologies with similar words
- Navigating semantic horizon
  Gyllensten and Sahlgren [2015]

Semantic Vectors for Gender Bias Study

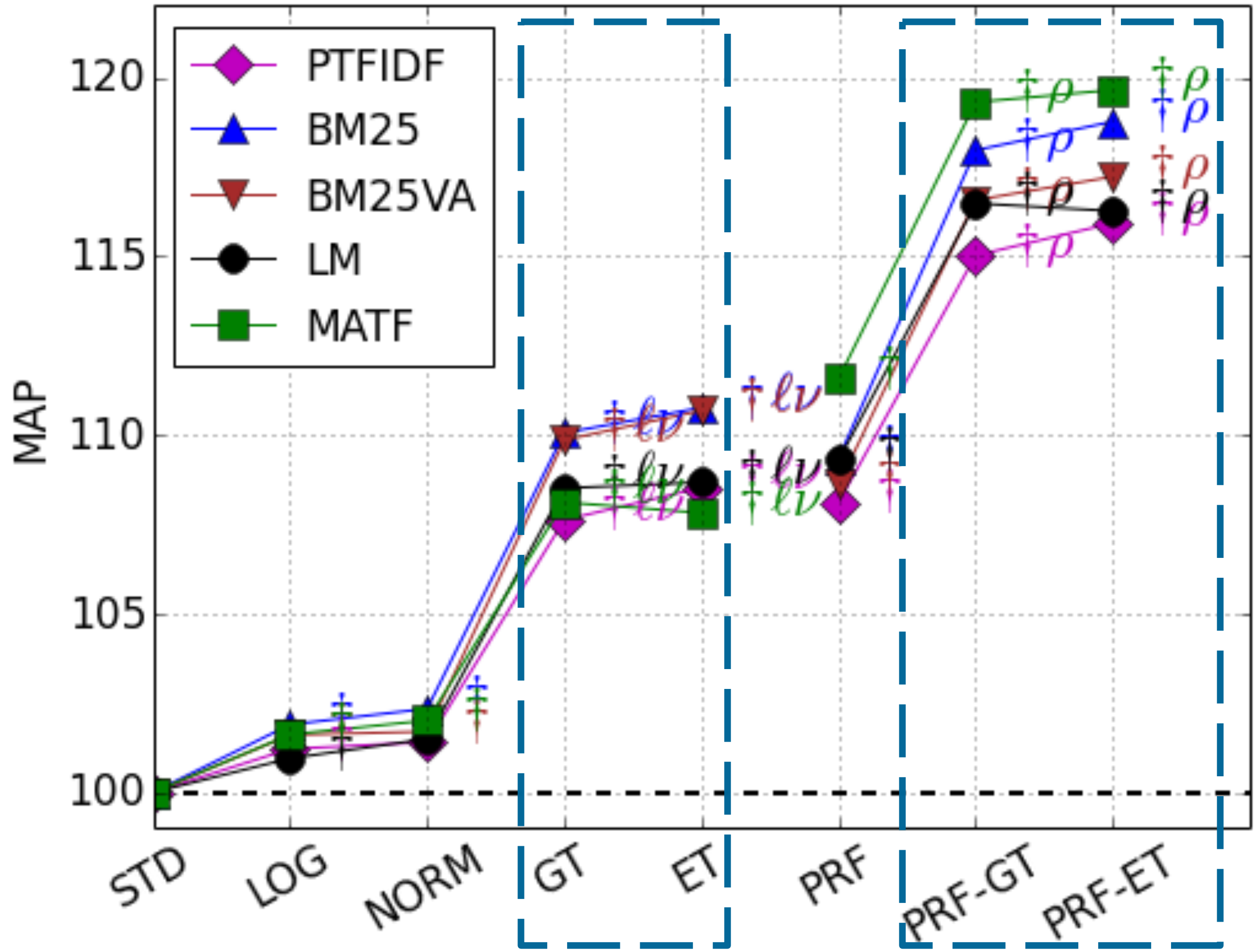
- The inclinations of 350 occupations to female/male factors as represented in Wikipedia



work in progress

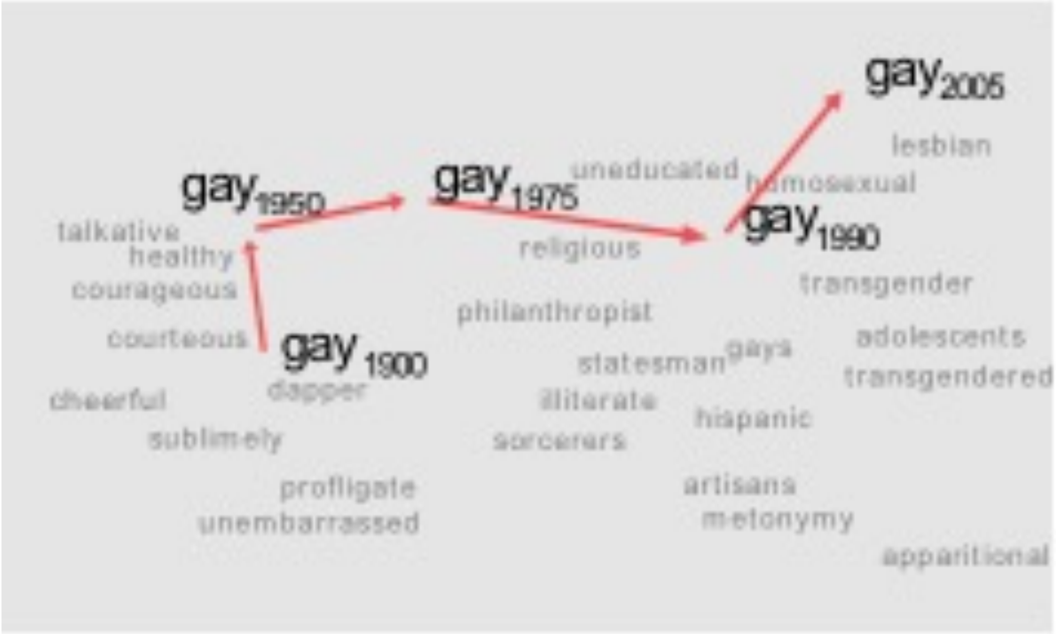
Semantic Vectors for Search

Gain of the evaluation results of document retrieval using semantic vectors expanding query terms



Rekabsaz et al.[2016]

Semantic Vectors in Text Analysis



Historical meaning shift *Kulkarni et al.[2015]*

Semantic vectors are the building blocks of many applications:

- Sentiment Analysis
- Question answering
- Plagiarism detection
- …

Terminology

Various names:

- Semantic vectors
- Vector representations of words
- Semantic word representation
- Distributional semantics
- Distributional representations of words
- Word embedding

Agenda

- Sparse vectors
  - Word-context co-occurrence matrix with term frequency or Point Mutual Information (PMI)
- Dense Vectors
  - Count-based: Singular Value Decomposition (SVD) in the case of Latent Semantic Analysis (LSA)
  - Prediction-based: word2vec Skip-Gram, inspired from neural network methods

Intuition

"You shall know a word by the company it keeps!"

J. R. Firth, A synopsis of linguistic theory 1930–1955 (1957)

Intuition

"In most cases, the meaning of a word is its use."

Ludwig Wittgenstein, Philosophical Investigations (1953) drink

drunk

alcohol

on the table

Tesgüino

make

out of corn

fermented

Mexico

bottle of

Nida[1975]

Dutch

drunk

pale

Heineken

brew

red star

bar

drink

green bottle

alcohol

Tesgüino ←→ Heineken

Algorithmic intuition:

Two words are related when they have similar context words

Sparse Vectors

Word-Document Matrix

- D is a set of documents (plays of Shakespeare)
- V is the set of words in the collection
- Words as rows and documents as columns
- Value is the count of word w in document d: $tc_{w,d}$
- Matrix size $|V| \times |D|$

	d_1 As You Like It	d_2 Twelfth Night	d_3 Julius Caesar	d_4 Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0
...	...	...	...	...

- Other word weighting models: tf, $tfidf$, $BM25$

[1]

Word-Document Matrix

	d_1 As You Like It	d_2 Twelfth Night	d_3 Julius Caesar	d_4 Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0

- Similarity between the vectors of two words:

$$sim(\text{soldier}, \text{clown}) = \cos\left(\overrightarrow{W}_{\text{soldier}}, \overrightarrow{W}_{\text{clown}}\right) = \frac{\overrightarrow{W}_{\text{soldier}} \cdot \overrightarrow{W}_{\text{clown}}}{|\overrightarrow{W}_{\text{soldier}}||\overrightarrow{W}_{\text{clown}}|}$$

Context

- Context can be defined in different ways
  - Document
  - Paragraph, tweet
  - **Window of some words (2-10) on each side of the word**

- Word-Context matrix
  - We consider every word as a dimension
  - Number of dimensions of the matrix: $|V|$
  - Matrix size: $|V| \times |V|$

Word-Context Matrix

- Window context of 7 words

sugar, a sliced lemon, a tablespoonful of	**apricot**	preserve or 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

	c_1 aardvark	c_2 computer	c_3 data	c_4 pinch	c_5 result	c_6 sugar
w_1 apricot	0	0	0	1	0	1
w_2 pineapple	0	0	0	1	0	1
w_3 digital	0	2	1	0	1	0
w_4 information	0	1	6	0	4	0

[1]

Co-occurrence Relations

	c_1 aardvark	c_2 computer	c_3 data	c_4 pinch	c_5 result	c_6 sugar
w_1 apricot	0	0	0	1	0	1
w_2 pineapple	0	0	0	1	0	1
w_3 digital	0	2	1	0	1	0
w_4 information	0	1	6	0	4	0

- First-order co-occurrence relation
  - Each cell of the word-context matrix
  - Words that appear near each other in the language
  - Like *drink* to *beer* or *wine*
- Second-order co-occurrence relation
  - Cosine similarity between the semantic vectors
  - Words that appear in similar contexts
  - Like *beer* to *wine*, or *knowledge* to *wisdom*

Point Mutual Information

- Problem with raw counting methods
  - Biased towards high frequent words (“and”, “the”) although they don’t contain much of information
- We need a measure for the first-order relation to assess how informative the co-occurrences are
- Use the ideas in information theory
- Point Mutual Information (PMI)
  - Probability of the co-occurrence of two events, divided by their independent occurrence probabilities

$$PMI(X,Y) = \log_2 \frac{P(X,Y)}{P(X)P(Y)}$$

Point Mutual Information

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

$$P(w,c) = \frac{\#(w,c)}{\sum_{i=1}^{|V|} \sum_{j=1}^{|V|} \#(w_i,c_j) = S}$$

$$P(w) = \frac{\sum_{j=1}^{|V|} \#(w,c_j)}{S} \qquad P(c) = \frac{\sum_{i=1}^{|V|} \#(w_i,c)}{S}$$

- Positive Point Mutual Information (PPMI)

$$PPMI(w,c) = \max(PMI, 0)$$

Point Mutual Information

	c_1 computer	c_2 data	c_3 pinch	c_4 result	c_5 sugar
w_1 apricot	0	0	1	0	1
w_2 pineapple	0	0	1	0	1
w_3 digital	2	1	0	1	0
w_4 information	1	6	0	4	0

$$P(w = \text{information}, c = \text{data}) = {}^{6}/_{19} = .32$$

$$P(w = \text{information}) = {}^{11}/_{19} = .58$$

$$P(c = \text{data}) = {}^{7}/_{19} = .37$$

$$PPMI(w = \text{information}, c = \text{data}) = \max(0, \frac{.32}{.58 * .37}) = .57$$

Point Mutual Information

Co-occurrence raw count matrix

	c_1 computer	c_2 data	c_3 pinch	c_4 result	c_5 sugar
w_1 apricot	0	0	1	0	1
w_2 pineapple	0	0	1	0	1
w_3 digital	2	1	0	1	0
w_4 information	1	6	0	4	0

PPMI matrix

	c_1 computer	c_2 data	c_3 pinch	c_4 result	c_5 sugar
w_1 apricot	-	-	2.25	-	2.25
w_2 pineapple	-	-	2.25	-	2.25
w_3 digital	1.66	0.00	-	0.00	-
w_4 information	0.00	0.57	-	0.47	-

Dense Vectors

Sparse vs. Dense Vectors

- Sparse vectors
  - Length between 20K to 500K
  - Many words don't co-occur; ~98% of the PPMI matrix is 0
- Dense vectors
  - Length 50 to 1000
  - Approximate the original data with lower dimensions -> lossy compression
- Why dense vectors?
  - Easier to store and load (efficiency)
  - Better for machine learning algorithms as features
  - Generalize better by removing noise for unseen data
  - Capture higher-order of relation and similarity: *car* and *automobile* might be merged into the same dimension and represent a topic

Dense Vectors

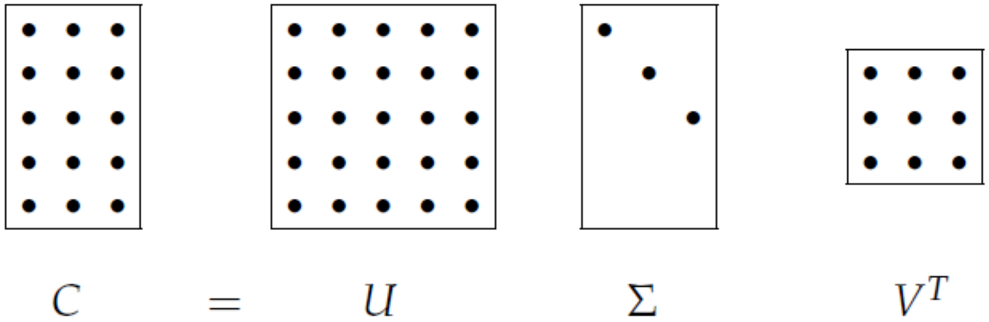
- Count based
  - Singular Value Decomposition in the case of Latent Semantic Analysis/Indexing (LSA/LSI)
  - Decompose the word-context matrix and truncate a part of it
- Prediction based
  - word2vec Skip-Gram model generates word and context vectors by optimizing the probability of co-occurrence of words in sliding windows

Singular Value Decomposition

- Theorem: An $m \times n$ matrix C of rank r has a Singular Value Decomposition (SVD) of the form

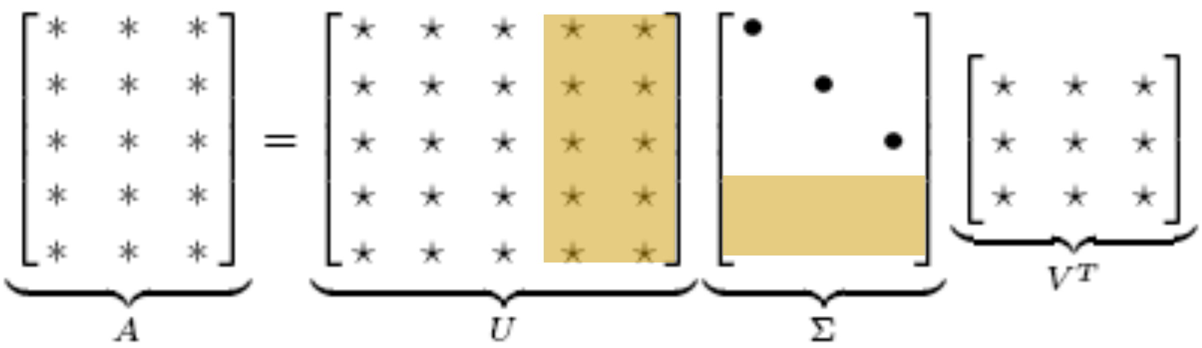
$$C = U\Sigma V^T$$

  - U is an $m \times m$ unitary matrix ($U^T U = UU^T = I$)
  - Σ is an $m \times n$ diagonal matrix, where the values (eigenvalues) are sorted, showing the importance of each dimension
  - V^T is an $n \times n$ unitary matrix



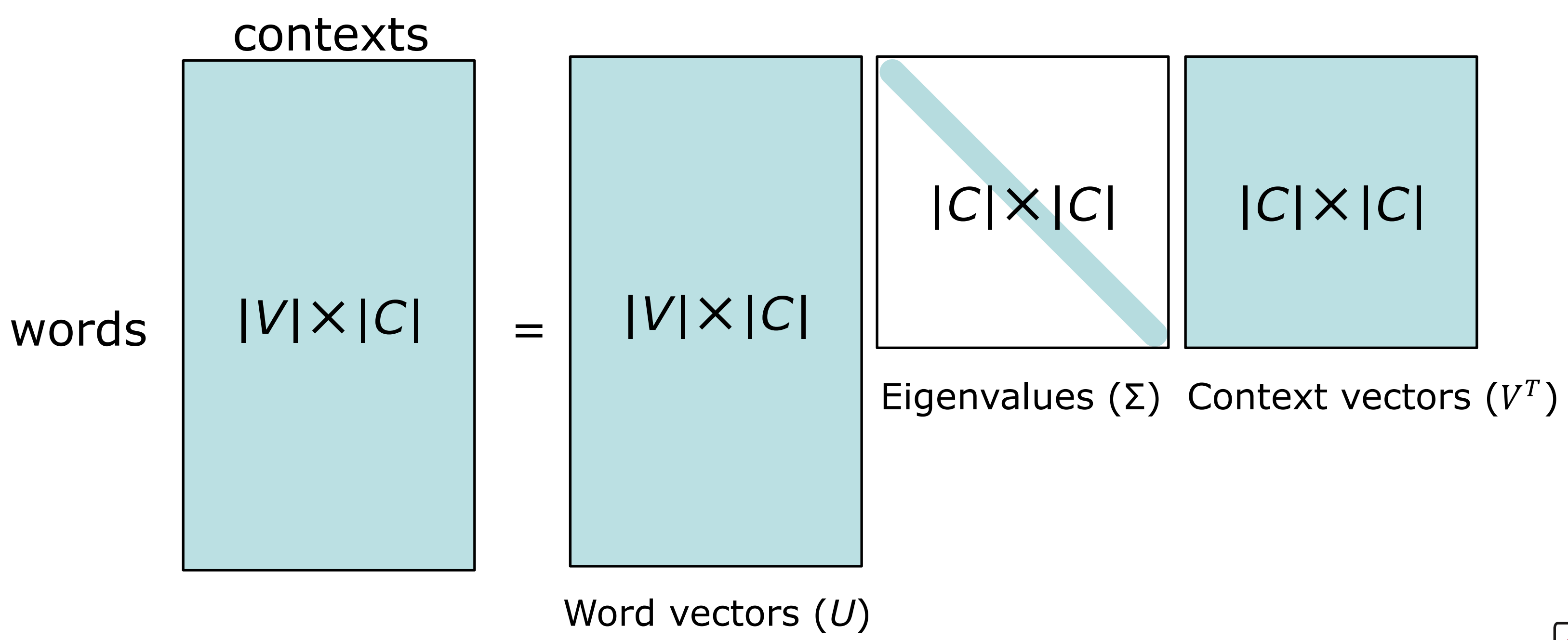
Singular Value Decomposition

- It is conventional to represent Σ as an $r \times r$ matrix
- Then the rightmost $m - r$ columns of U are omitted or the rightmost $n - r$ columns of V are omitted



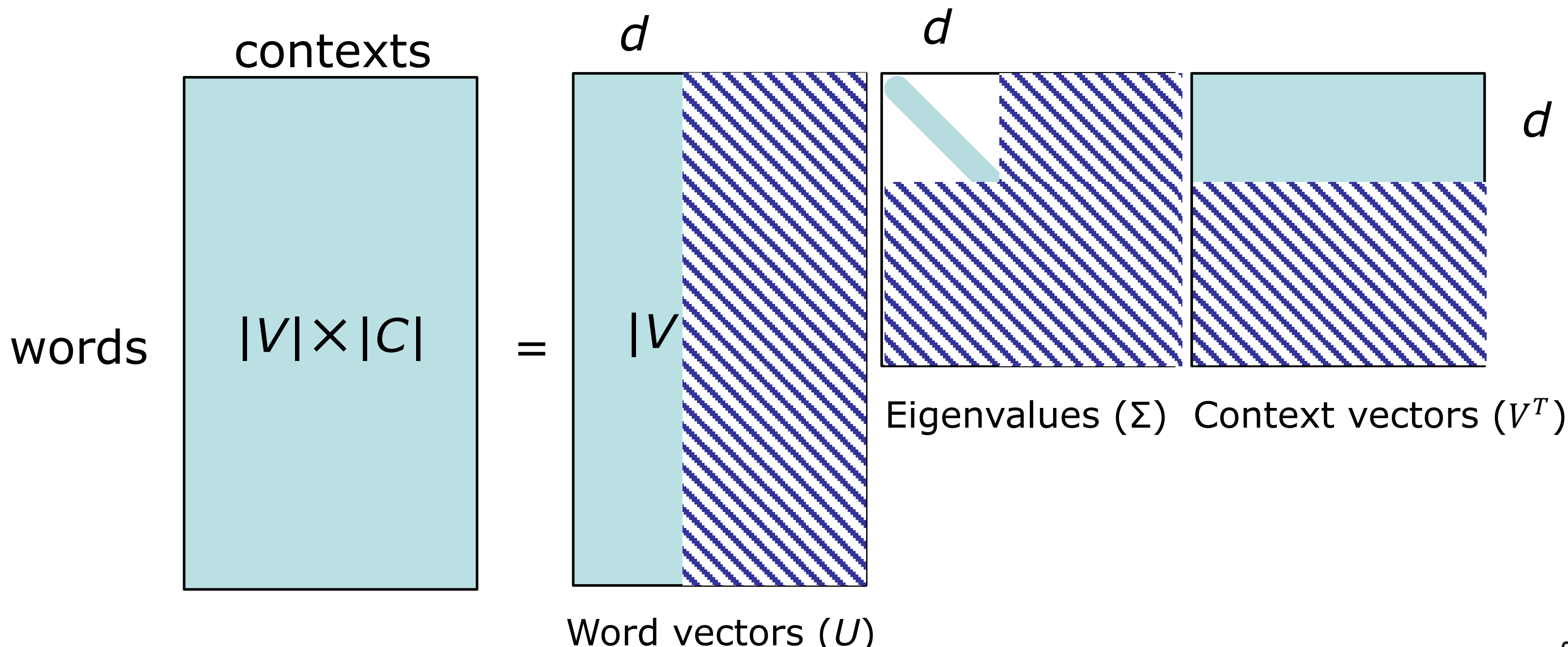
Applying SVD to Term-Context Matrix

- Start with a sparse PPMI matrix of the size $|V| \times |C|$ where $|V| > |C|$ (in practice $|V| = |C|$)
- Apply SVD



Applying SVD to Term-Context Matrix

- Keep only top d eigenvalues in Σ and set the rest to zero
- Truncate the U and V^T matrices based on the changes in Σ
- If we multiply the truncated matrices, we have a least-squares approximation of the original matrix
- Our dense semantic vectors is the truncated U matrix



Prediction instead of Counting

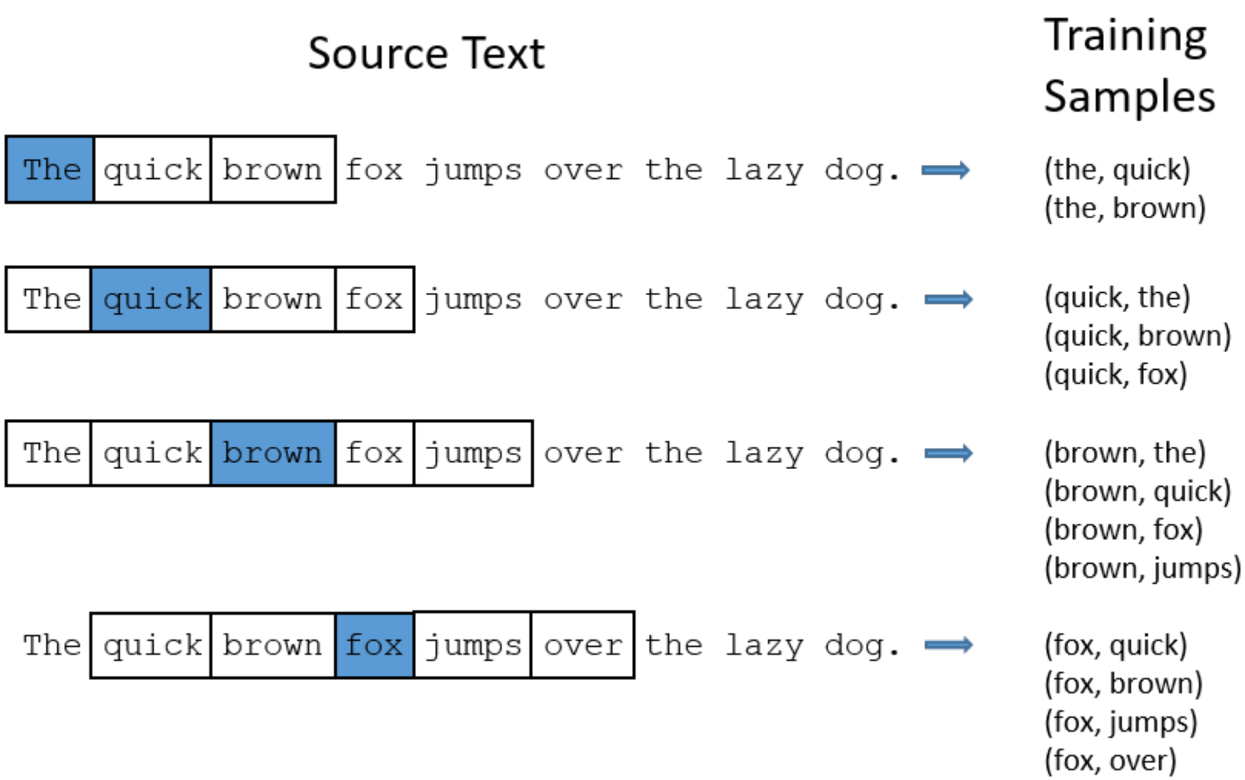
- Instead of counting, we want to predict the probability of occurrence of a word, given another word
- The prediction approach has roots in language modeling:
  - E.g.: I order a pizza with … (mashroom: 0.1, lizard: 0.001)
- We want to calculate the probability of appearance of a context word *c* in a window context given the word *w*:

$$P(c|w)$$

- Based on this probability, we define an objective function
- We aim to learn word representations by optimizing the error of the objective function on a training corpus
- **word2vec** [6,7] introduces an efficient and also effective method
- We study the **Skip-Gram** architecture, **CBOW** is very similar

Skip-Gram

- The Neural Network is trained by feeding it word pairs found in the text within a context window
- Below is an example with a window size of 2

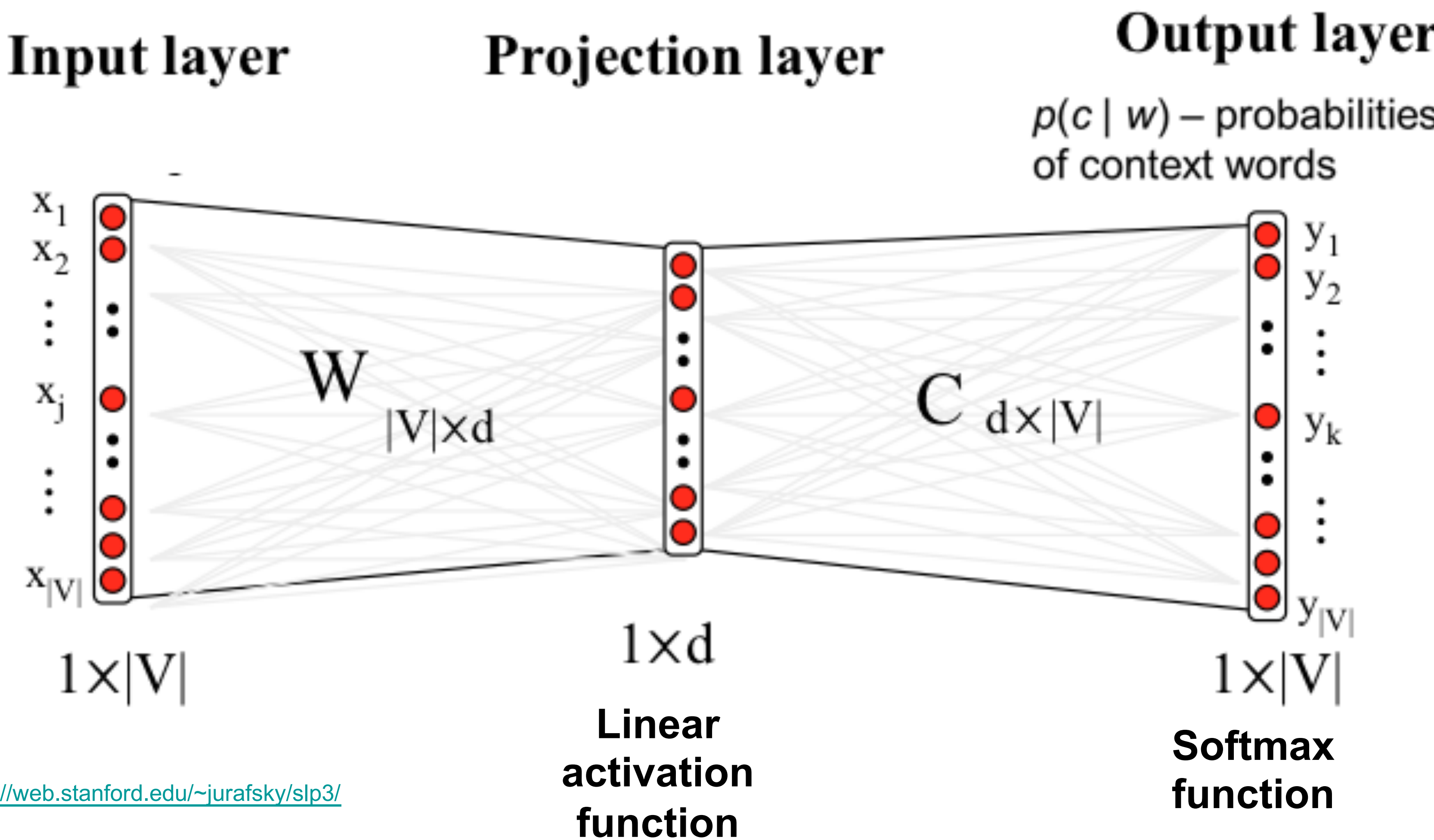


$w \in V$ and $c \in V$ are a word and its context

http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

A Neural Network Model for Prediction of Context Word

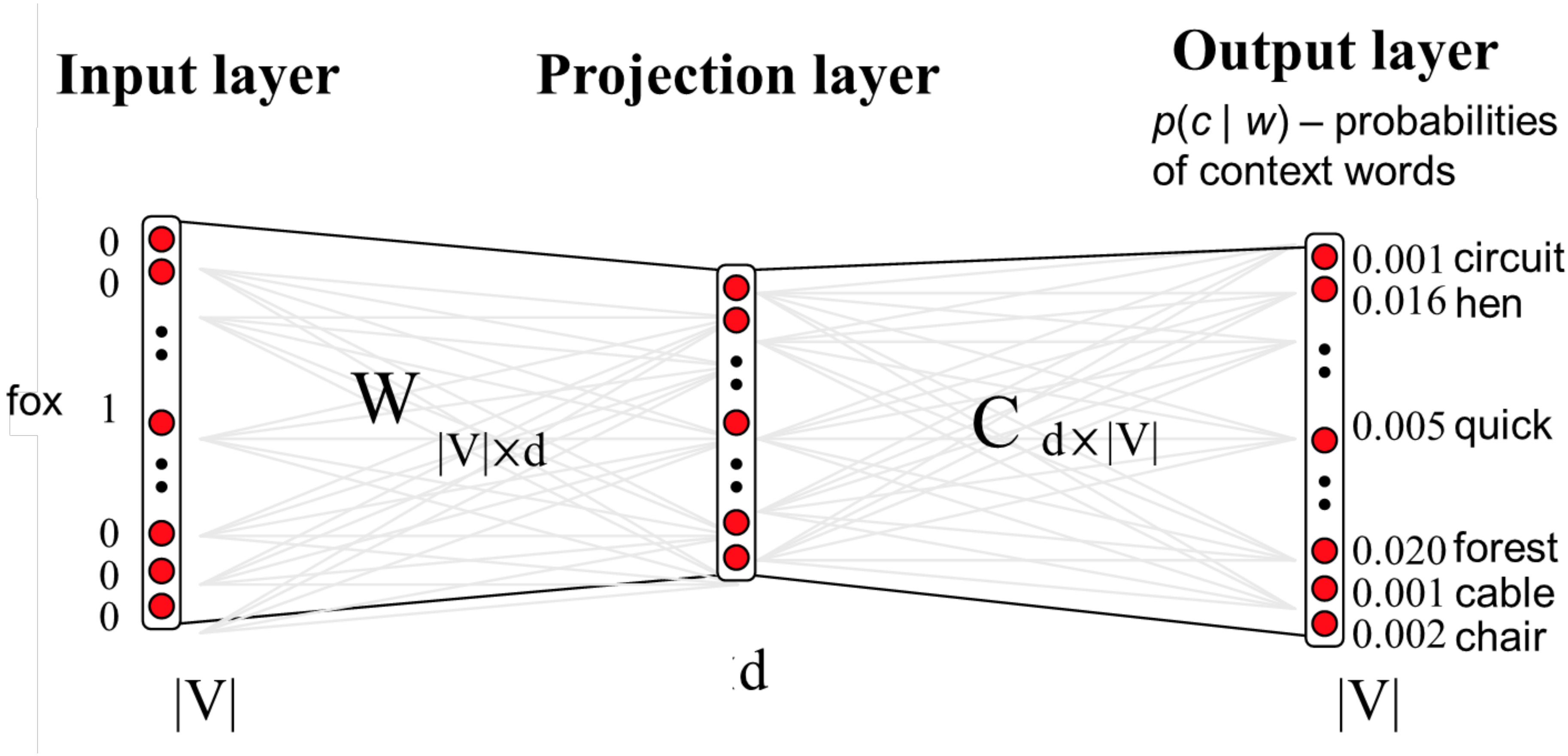
- The network predicts $P(c|w)$ i.e. *w* at input and *c at* output layer
- Two sets of vectors: word vectors *W* and context vector *C*



https://web.stanford.edu/~jurafsky/slp3/

The Prediction Results after Training

- After training, given the word *fox*, the network outputs the probability of appearance of every word in its window context



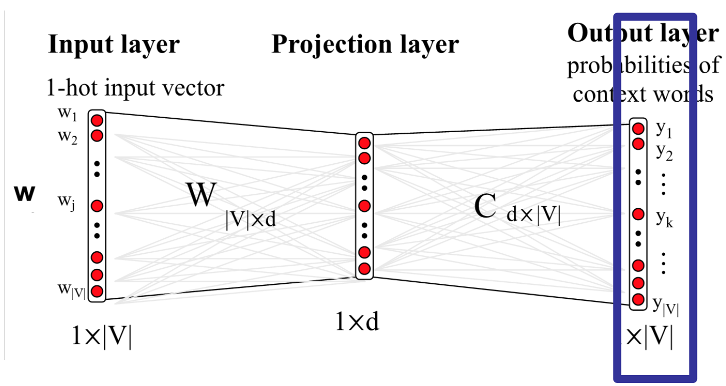
What is Softmax at the Output Layer

- Given the pair of *(w,c)*, the output value of the last layer in this network is in fact the dot product of the word vector to the context vector:

$$W_w \cdot C_c$$

- In order to turn this output into probability distribution, the outputs are normalised using the Softmax function:

$$p(c|w) = \frac{\exp(W_w \cdot C_c)}{\sum_{l \subset V} \exp(W_w \cdot C_l)}$$



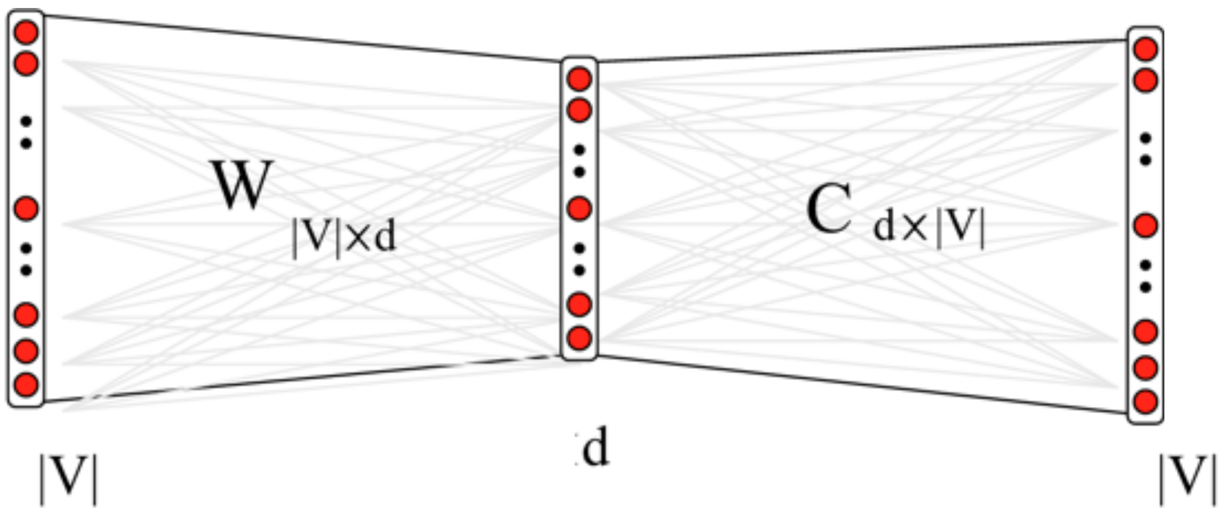
How to Train the Neural Network Model

1. The *W* and *C* vectors are randomly initialized
2. Slide the window over the corpus:

(w,c) = (fox, forest)

3. Input w with a one-hot vector
4. Calculate output layer for the context word:

$$p(\mathrm{c}|w) = p(\mathrm{forest}|\mathrm{fox}) = \frac{\exp(W_{\mathrm{fox}} \cdot C_{\mathrm{forest}})}{\sum_{l \subset V} \exp(W_{\mathrm{fox}} \cdot C_l)}$$



How to Train the Neural Network Model

4. Calculate the cross entropy cost function for each batch with *T* instances:

$$J = -\frac{1}{T}\sum_{1}^{T} \log p(c|w)$$

5. Minimize the cost function:
   - Need to increase $W_{\text{fox}} \cdot C_{\text{forest}}$
   - Update both W_{fox} and C_{forest} vectors by adding a portion of W_{fox} to C_{forest} and other way around

6. Continue training on the next *(w,c)* pairs:

(*w*,*c*)=(wolf, forest)

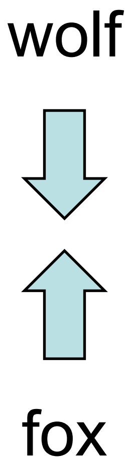
(*w*,*c*)=(resistor, circuit)

(*w*,*c*)=(wolf, tree)

(*w*,*c*)=(fox, tree)

Embedding Space

- Vectors associated with words that occur in the same context become more similar to each other



The Neural Network Prediction Model - Summary

- Prediction probability

$$p(c|w) = \frac{\exp(W_w \cdot C_c)}{\sum_{l \subset V} \exp(W_w \cdot C_l)}$$

- Cross entropy cost function

$$J = -\frac{1}{T}\sum_{1}^{T} \log p(c|w)$$

- Problem: the calculation of the denominator in the prediction probability is very expensive!

- One approach to tackle the efficiency problem is using Negative Sampling, introduced in the word2vec toolbox

word2vec: Probability of a Genuine Co-occurrence

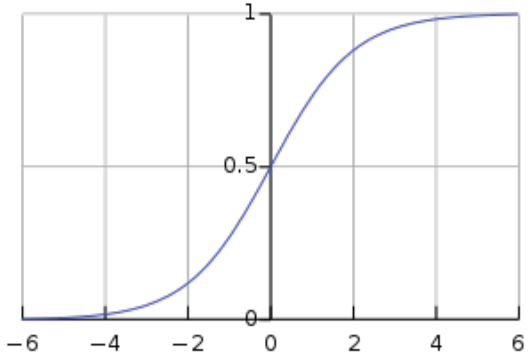
- Let's introduce a binary variable *y*, measuring how genuine the probability of co-occurrence of *w* and *c* is:

$$p(y = 1|w, c)$$

- This probability is estimated by the sigmoid function of the dot product of the word vector and context vector:

$$p(y = 1|w, c) = \frac{1}{1 + \exp(-W_w \cdot C_c)} = \sigma(W_w \cdot C_c)$$

- For example, we expect to have:
  - $p(y = 1|\text{fox}, \text{forest}) = 0.98$
  - $p(y = 0|\text{fox}, \text{forest}) = 1 - 0.98 = 0.02$
  - $p(y = 1|\text{fox}, \text{tree}) = 0.96$
  - $p(y = 1|\text{fox}, \text{chair}) = 0.01$
  - $p(y = 1|\text{fox}, \text{circuit}) = 0.001$



word2vec: Negative Sampling

- If we only use $p(y=1|w,c)$, we lack comparison or normalization over other words!!
- Instead of a complete normalization, we use Negative Sampling
- Negative Sampling intuition:

> The word *w* should attracts the context *c* when they appear in the same context and repeals some other context words $\check{c}$ that do not co-occur with *w* i.e. negative samples

- Since many words don't co-occur, any sampled word can be assumed as a negative sample
- We randomly sample *k* (2-20) words from the collection distribution
- We aim to increase $p(y=1|w,c)$ and decrease $p(y=1|w,\check{c})$

- For example with *k*=2

(w,c) = (fox, forest)

negative samples: [bluff, guitar]

$$p(y=1|\text{fox},\text{forest}) \uparrow$$

$$p(y=1|\text{fox},\text{bluff}) \downarrow \quad \Rrightarrow \quad p(y=0|\text{fox},\text{bluff}) \uparrow$$

$$p(y=1|\text{fox},\text{guitar}) \downarrow \quad \Rrightarrow \quad p(y=0|\text{fox},\text{guitar}) \uparrow$$

(w,c) = (wolf, forest)

negative samples: [blooper, film]

$$p(y=1|\text{wolf},\text{forest}) \uparrow$$

$$p(y=0|\text{wolf},\text{blooper}) \uparrow$$

$$p(y=0|\text{wolf},\text{film}) \uparrow$$

Random words from https://www.textfixer.com/tools/random-words.php

word2vec with Negative Sampling

- Genuine co-occurrence probability

$$p(y = 1|w, c) = \sigma(W_w \cdot C_c)$$

- Negative sampling of *k* context words $\check{c}$

$$p(y = 0|w, \check{c})$$

- Cost function

$$J = -\frac{1}{T}\sum_{1}^{T}\left[\log p(y = 1|w, c) + \sum_{i=1}^{k}\log p(y = 0|w, \check{c})\right]$$

↑ co-occurrence probability

↑ Negative sampling

word2vec with Negative Sampling

(w,c) = (fox, forest)

negative samples: [bluff, guitar]

$p(y = 1|\text{fox}, \text{forest}) \uparrow$

$p(y = 0|\text{fox}, \text{bluff}) \uparrow$

$p(y = 0|\text{fox}, \text{guitar}) \uparrow$

(w,c) = (wolf, forest)

negative samples: [blooper, film]

$p(y = 1|\text{wolf}, \text{forest}) \uparrow$

$p(y = 0|\text{wolf}, \text{blooper}) \uparrow$

$p(y = 0|\text{wolf}, \text{film}) \uparrow$

word2vec with Negative Sampling

(w,c) = (fox, forest)

negative samples: [bluff, guitar]

$p(y = 1|\text{fox}, \text{forest}) \uparrow$ W_{fox} attracts C_{forest}

$p(y = 0|\text{fox}, \text{bluff}) \uparrow$ W_{fox} repeals C_{bluff}

$p(y = 0|\text{fox}, \text{guitar}) \uparrow$ W_{fox} repeals C_{guitar}

(w,c) = (wolf, forest)

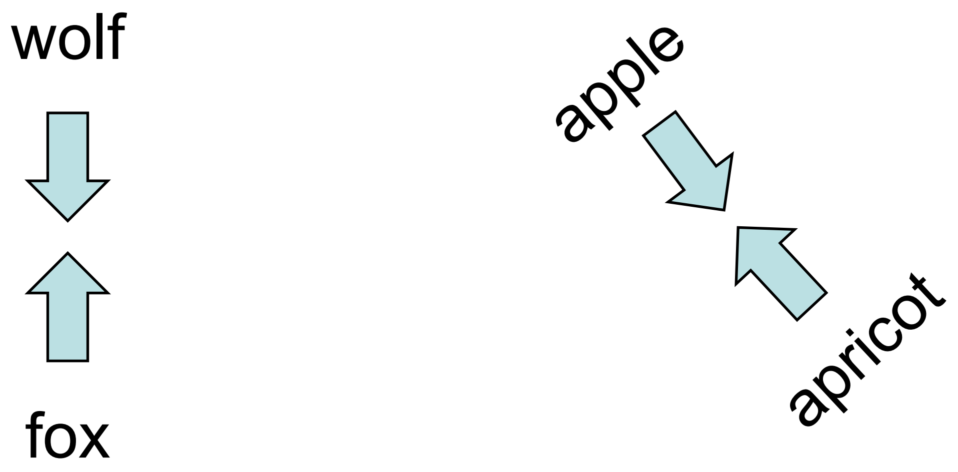
negative samples: [blooper, film]

$p(y = 1|\text{wolf}, \text{forest}) \uparrow$ W_{wolf} attracts C_{forest}

$p(y = 0|\text{wolf}, \text{blooper}) \uparrow$ W_{wolf} repeals C_{bloopers}

$p(y = 0|\text{wolf}, \text{film}) \uparrow$ W_{wolf} repeals C_{film}

- Eventually words with similar contexts (like *fox* and *wolf* or *apple* and *apricot*) become more similar to each other and different from the rest



- Very frequent words dominant the model and influence the performance of the vectors. Solutions:
- Subsampling
  - When creating the window, remove the words with frequency *f* higher than *t* with the following probability

$$p = 1 - \sqrt{\frac{t}{f}}$$

- Context Distribution Smoothing
  - Dampens the values of the collection distribution for negative sampling with $f^{3/4}$ $f = 10000 \rightarrow f^{3/4} = 1000$
  - Prevents domination of very frequent words in sampling

References

[1] Jurafsky, Dan, and James H. Martin. *Speech and language processing*. Vol. 3. London: Pearson, 2014.

[2] Exploration of a Threshold for Similarity based on Uncertainty in Word Embedding. *Navid Rekabsaz, Mihai Lupu, Allan Hanbury, Guido Zuccon* In Proceedings of the European Conference on Information Retrieval Research

[3] Navigating the semantic horizon using relative neighborhood graph. *Amaru Cuba Gyllensten and Magnus Sahlgren. In Proceedings of EMNLP 2015*.

[4] Generalizing Translation Models in the Probabilistic Relevance Framework *Navid Rekabsaz, Mihai Lupu, Allan Hanbury, Guido Zuccon* Proceedings of ACM International Conference on Information and Knowledge Management (CIKM 2016)

[5] Kulkarni, Vivek, et al. "Statistically significant detection of linguistic change." *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2015.

[6] Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." *Advances in neural information processing systems*. 2013.

[7] Mikolov, Tomas, et al. "Efficient estimation of word representations in vector space." *arXiv preprint arXiv:1301.3781* (2013).

Thanks!

Questions?

@NRekabsaz rekabsaz@ifs.tuwien.ac.at