IN4080 – 2022 FALL

NATURAL LANGUAGE PROCESSING

Jan Tore Lønning

Vectors, Distributions, Embeddings

Lecture 7, Oct 6

Today

- Lexical semantics
- Vector models of documents
- tf-idf weighting
- Word-context matrices
- Word embeddings with dense vectors
- Word2vec

The meaning of words

- Words (lecture 2)
  - Type – token
  - Word – lexeme – lemma
  - Meaning?

Look into the dictionary

lemma

sense

definition

pepper, *n.*

Pronunciation: Brit. /ˈpɛpə/ , U.S. /ˈpɛpər/

Forms: OE peopor (*rare*), OE pipcer (transmission error), OE pipor , OE pipur (*rare*

Frequency (in current use):

Etymology: A borrowing from Latin. Etymon: Latin *piper* .

< classical Latin *piper* , a loanword < Indo-Aryan (as is ancient Greek *πέπερι*); compare Sa

I. The spice or the plant.

1.

a. A hot pungent spice derived from the prepared fruits (peppercorns) of the pepper plant, *Piper nigrum* (see sense 2a), used from early times to season food, either whole or ground to powder (often in association with salt). Also (locally, chiefly with distinguishing word): a similar spice derived from the fruits of certain other species of the genus *Piper* ; the fruits themselves.

The ground spice from *Piper nigrum* comes in two forms, the more pungent *black pepper* , produced from black peppercorns, and the milder *white pepper* , produced from white peppercorns: see BLACK *adj.* and *n.* Special uses 5a, PEPPERCORN *n.* 1a, and WHITE *adj.* and *n.*[1] Special uses 7b(a).

2.

a. The plant *Piper nigrum* (family Piperaceae), a climbing shrub indigenous to South Asia and also cultivated elsewhere in the tropics, which has alternate stalked entire leaves, with pendulous spikes of small green flowers opposite the leaves, succeeded by small berries turning red when ripe. Also more widely: any plant of the genus *Piper* or the family Piperaceae.

b. Usu. with distinguishing word: any of numerous plants of other families having hot pungent fruits or leaves which resemble pepper (1a) in taste and in some cases are used as a substitute for it.

c. *U.S.* The California pepper tree, *Schinus molle*. Cf. PEPPER TREE *n.* 3.

3. Any of various forms of capsicum, esp. *Capsicum annuum* var. *annuum*. Originally (chiefly with distinguishing word): any variety of the *C. annuum* Longum group, with elongated fruits having a hot, pungent taste, the source of cayenne, chilli powder, paprika, etc., or of the

- A word with several senses is called polysemous
- If two different words look and sound the same, they are called homonyms

- How to tell: one word or several?
  - Common origin
  - But not waterproof/easy to see

Relations between senses

Term	Definition	Examples		

Relations between senses

Term	Definition	Examples		
Synonymy	Have the same meaning in all(?)/some(?) contexts	*sofa-couch, bus-coach* *big-large*		

Relations between senses

Term	Definition	Examples		
Synonymy	Have the same meaning in all(?)/some(?) contexts	*sofa-couch, bus-coach big-large*		
Antonym	Opposites with respect to a feature of meaning	*true-false, strong-weak, up-down*		

Relations between senses

Term	Definition	Examples		
Synonymy	Have the same meaning in all(?)/some(?) contexts	*sofa-couch, bus-coach big-large*		
Antonym	Opposites with respect to a feature of meaning	*true-false, strong-weak, up-down*		
Hyponym-hypernym	The <hyponym> is a type-of the <hypernym>	*rose →flower, cow →animal, car →vehicle*		

Relations between senses

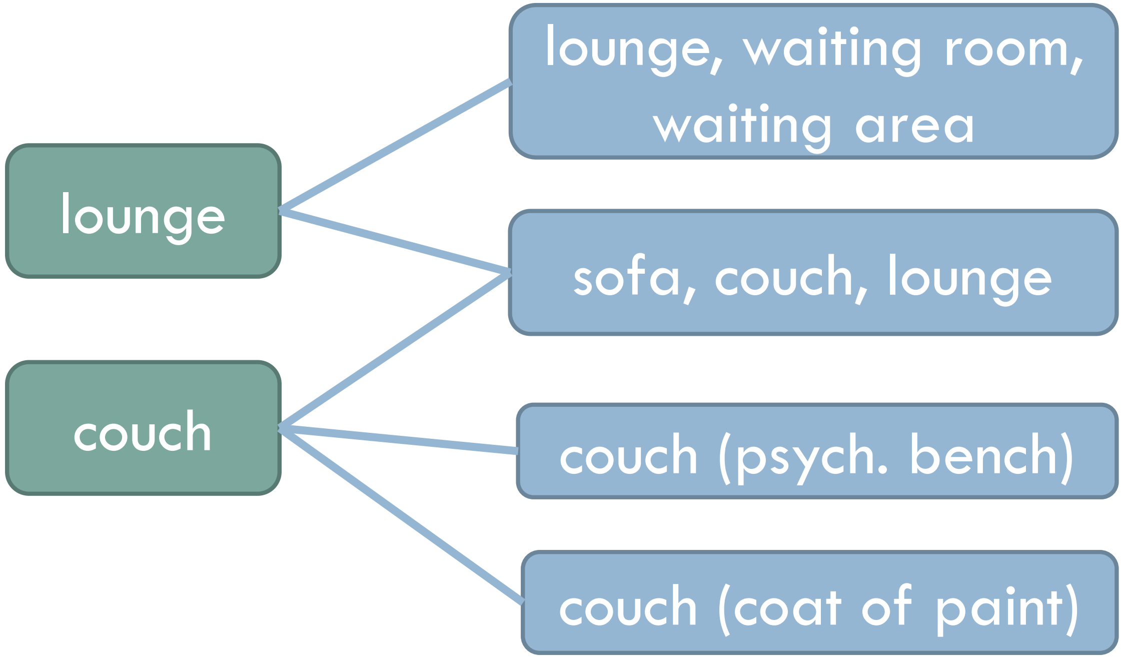
Term	Definition	Examples		
Synonymy	Have the same meaning in all(?)/some(?) contexts	*sofa-couch, bus-coach* *big-large*		
Antonym	Opposites with respect to a feature of meaning	*true-false, strong-weak, up-down*		
Hyponym-hypernym	The <hyponym> is a type-of the <hypernym>	*rose →flower, cow →animal, car →vehicle*		
Similarity		*cow-horse* *boy-girl*		

Relations between senses

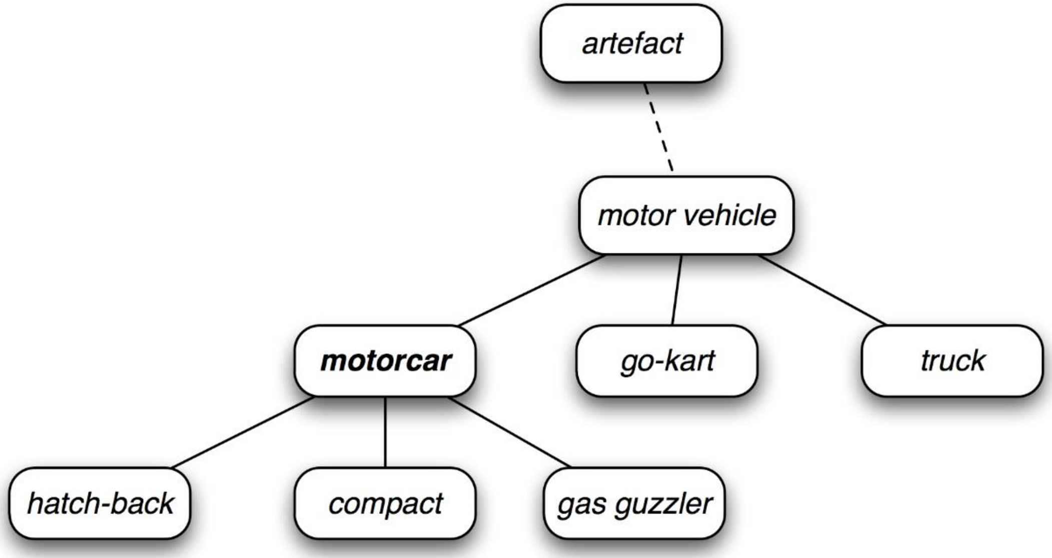
Term	Definition	Examples		
Synonymy	Have the same meaning in all(?)/some(?) contexts	*sofa-couch, bus-coach* *big-large*		
Antonym	Opposites with respect to a feature of meaning	*true-false, strong-weak, up-down*		
Hyponym-hypernym	The <hyponym> is a type-of the <hypernym>	*rose →flower, cow →animal, car →vehicle*		
Similarity		*cow-horse* *boy-girl*		
Related		*money-bank* *fish-water*		

Resources for lexical semantics: WordNet

- https://wordnet.princeton.edu
- To each word:
  - One or more synsets (synonymy)



- Hyponymy relations between the synsets



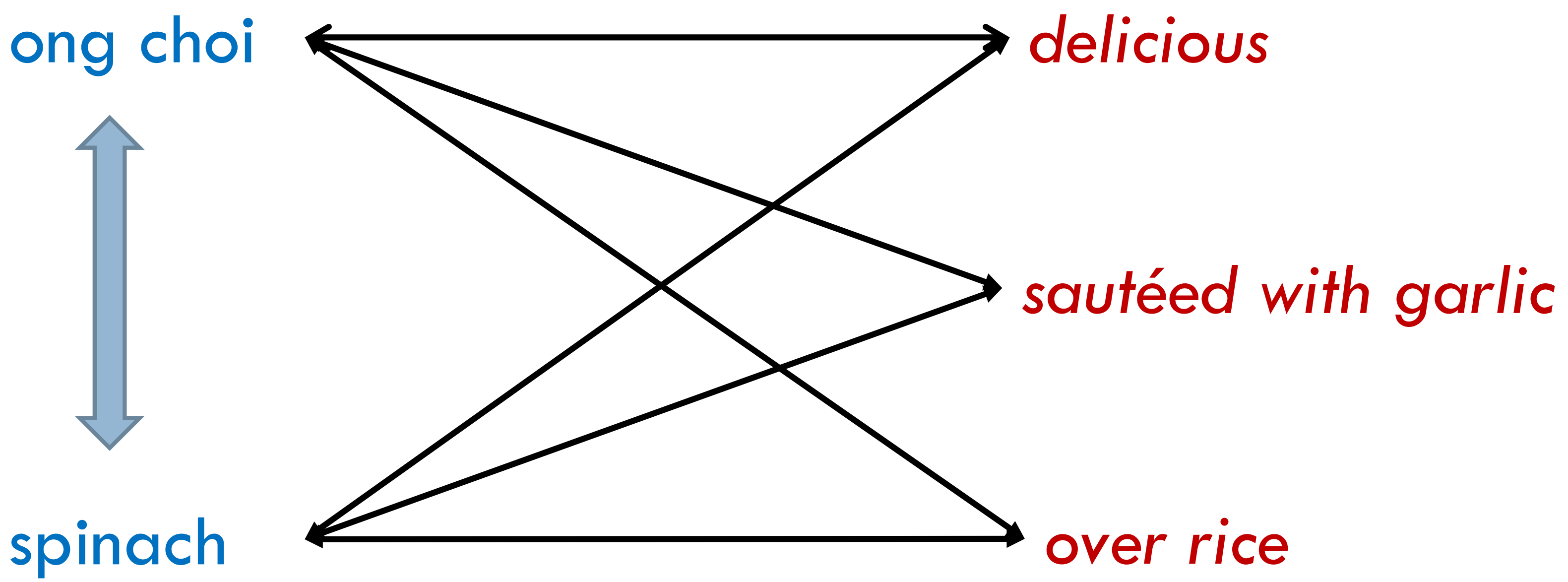
What does ongchoi mean?

- Suppose you see these sentences:
  - *Ong choi is delicious sautéed with garlic.*
  - *Ong choi is superb over rice*
  - *Ong choi leaves with salty sauces*
- And you've also seen these:
  - *…spinach sautéed with garlic over rice*
  - *Chard stems and leaves are delicious*
  - *Collard greens and other salty leafy greens*
- Conclusion: Ongchoi is a leafy green like spinach, chard, or collard greens

Similar

Related

(first-order association, syntagmatic)



Similar

(second-order association, paradigmatic)

The distributional hypothesis

- Words that occur in similar contexts have similar meanings

Today

- Lexical semantics
- Vector models of documents
- tf-idf weighting
- Word-context matrices
- Word embeddings with dense vectors
- Word2vec

Shakespeare (from J & M)

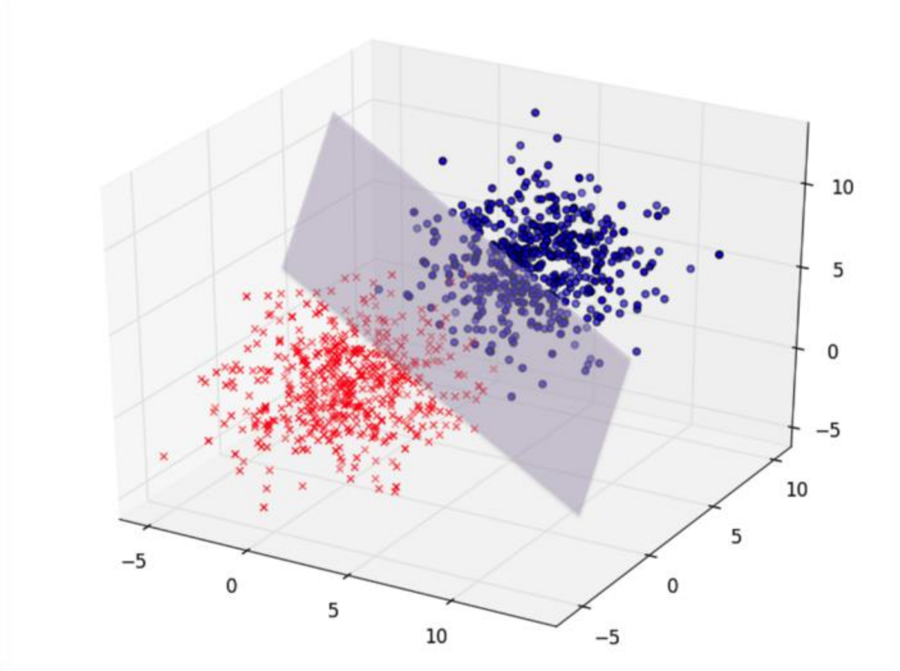
	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

- Vectors are similar for the two comedies
- Different than the historical dramas
- Comedies have more fools and wit and fewer battles.

- Notice similarity to text classification
- Mandatory 1B, multinomial
  - The document represented by a vector with the occurrences of 35,000 terms

Document classification

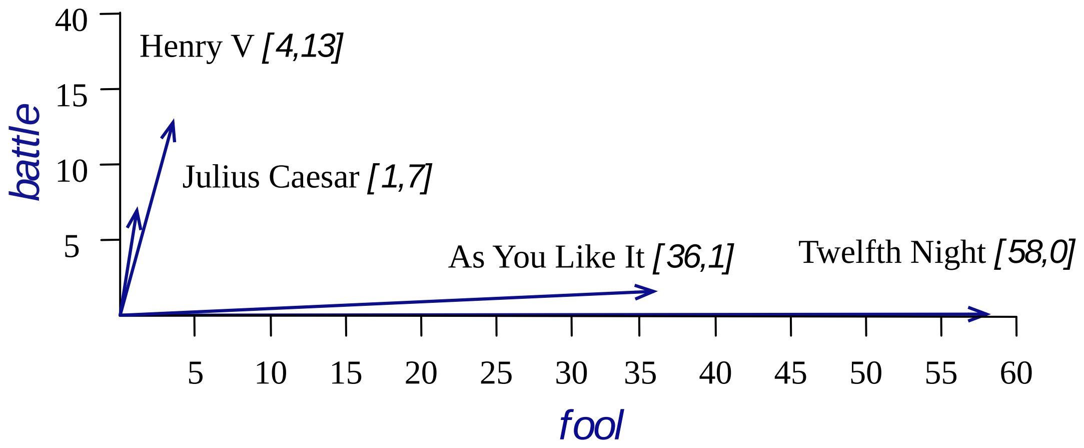
- The word vectors were used as basis for classification
- If two documents had the same vectors they were put in the same class
- Documents are similar = on the same side of the separating hyperplane



A problem to draw 35,000 dimensions

Information retrieval (IR)

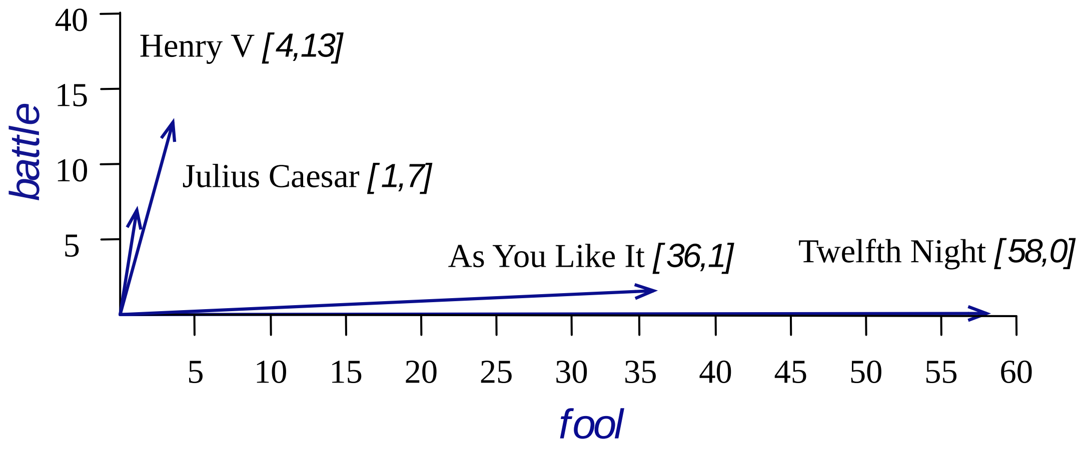
- Documents placed in the same *n*-dimensional space as in classification
- Retrieve documents similar to a given document



Cosine similarity

- Several possible ways to define similarity, e.g.,
  - Euclidean
  - Manhattan
- Most common: cosine
  - Do the arrows point in the same direction?

$$\cos(\vec{v},\vec{w}) = \frac{\vec{v}\cdot\vec{w}}{|\vec{v}||\vec{w}|} = \frac{\vec{v}}{|\vec{v}|}\cdot\frac{\vec{w}}{|\vec{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}}$$



Let us try: $\cos(v_1, v_2)$

Full vectors

	AYLI	TwNi	JuCa	HenV
AYLI	1.000	0.950	0.945	0.949
TwNi	0.950	1.000	0.809	0.822
JuCa	0.945	0.809	1.000	0.999
HenV	0.949	0.822	0.999	1.000

battles & fools

	AYLI	TwNi	JuCa	HenV
AYLI	1.000	1.000	0.169	0.321
TwNi	1.000	1.000	0.141	0.294
JuCa	0.169	0.141	1.000	0.988
HenV	0.321	0.294	0.988	1.000

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

Today

- Lexical semantics
- Vector models of documents
- tf-idf weighting
- Word-context matrices
- Word embeddings with dense vectors
- Word2vec

Inverse document frequency

- Are all words equally important?
- Intuition: A word occurring in a large proportion of documents is not a good discriminator.
- $idf_t = \log \frac{N}{df_t}$
  - df_t the number of documents containing t.

	Example: 1,000,000 documents, log_10							
df_t	1,000,000	100,000	10,000	1,000	100	10	1	0
idf_t	0	1	2	3	4	5	6	?

tf-idf

- Tf-idf weighting
- $tf_{t,d} \times idf_t$
  - $tf_{t,d}$ is the frequency of the term t in the document d

Variants of $tf\text{-}idf$

- $idf_t = \log \frac{N}{df_t}$
- scikit-learn:
  - $\log \frac{N}{df_t} + 1$
    - TfidfTransformer(smooth_idf=False)
  - $idf_t = \log \frac{N+1}{df_t+1} + 1$
    - TfidfTransformer(smooth_idf=True)
    - Here log_10 vs log_e matters
- and others in the literature
- Avoids log(0)

$tf_{t,d}$ - Alternatives

- Linear:
  - raw counts
  - relative frequencies (L1)
  - length normalization (L2)
  - all yield same cosine-similarity
- Sublinear
  - (J&M): $tf_{t,d} = \log(count(t,d) + 1)$
  - (sklearn) $tf_{t,d} = \log(count(t,d)) + 1$, 0 when $count(t,d) = 0$
    - TfidfTransformer(sub_linear=True)
  - Similar effect as log in idf_t

The effect of tf-idf

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	0.074	0	0.22	0.28
good	0	0	0	0
fool	0.019	0.021	0.0036	0.0083
wit	0.049	0.044	0.018	0.022

Figure 6.8 A tf-idf weighted term-document matrix for four words in four Shakespeare plays, using the counts in Fig. 6.2. For example the 0.049 value for *wit* in *As You Like It* is the product of tf $= \log_{10}(20+1) = 1.322$ and idf $= .037$. Note that the idf weighting has eliminated the importance of the ubiquitous word *good* and vastly reduced the impact of the almost-ubiquitous word *fool*.

The effect of tf-idf

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	0.074	0	0.22	0.28
good	0	0	0	0
fool	0.019	0.021	0.0036	0.0083
wit	0.049	0.044	0.018	0.022

	AYLI	TwNi	JuCa	HenV
AYLI	1.000	0.577	0.860	0.861
TwNi	0.577	1.000	0.081	0.083
JuCa	0.860	0.081	1.000	1.000
HenV	0.861	0.083	1.000	1.000

- Maybe not the expected effect
- Probably because of few terms.

Today

- Lexical semantics
- Vector models of documents
- tf-idf weighting
- Word-context matrices
- Word embeddings with dense vectors
- Word2vec

Vector repr. of words 1: A vector of documents

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

battle is "the kind of word that occurs in Julius Caesar and Henry V"

fool is "the kind of word that occurs in comedies, especially Twelfth Night"

Vector repr. of words 2: Word-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	

Word-context matrix

Document-term matrix

- Objects: a set of documents, D
- Features: a set of terms,
  - $T = \{t_1, t_2, \dots, t_n\}$
- Each document d is identified with a vector
  - $(v_1, v_2, \dots, v_n)$
  - where v_i is calculated from the frequency of t_i in d.

Word-context matrix

- Objects: a vocabulary of words, V
- Features: a set of words,
  - $C = \{c_1, c_2, \dots, c_n\}$
- A set of texts, T
- A definition of the context of an occurrence of w in T
- Each word w in V is identified with a vector
  - $(v_1, v_2, \dots, v_n)$
  - where v_i is calculated from the frequency of c_i in all the contexts of w in T

Word-context matrix

Comments

- C=V, or C is a smaller set of the most frequent terms
  - To avoid too large repr.
- Context, alternatives:
  - A sentence
  - A window of *k* tokens on each side
  - A document
  - Defined by grammatical relations (after parsing)

Word-context matrix

- Objects: a vocabulary of words, V
- Features: a set of words,
  - $C = \{c_1, c_2, \dots, c_n\}$
- A set of texts, T
- A definition of the context of an occurrence of w in T
- Each word w in V is identified with a vector
  - $(v_1, v_2, \dots, v_n)$
  - where v_i is calculated from the frequency of c_i in all the contexts of w in T

So-far

- A word w can be represented by a context vector v_w where position j in the vector reflects the frequency of occurrences of w_j with w.
- Can be used for
  - studying similarities between words.
  - document similarities
- But the vectors are *sparse*
  - Long: 20-50,000
  - Many entries are 0
- Even though *car* and *automobile* get similar vectors, because both co-occur with e.g., *drive*, in the vector for *drive* there is no connection between the *car* element and the *automobile* element.

Today

- Lexical semantics
- Vector models of documents
- tf-idf weighting
- Word-context matrices
- Word embeddings with dense vectors
- Word2vec

Dense vectors

How?

- Shorter vectors.
  - (length 50-1000)
  - ``low-dimensional'' space
- Dense (most elements are not 0)
- Intuitions:
  - Similar words should have similar vectors.
  - Words that occur in similar contexts should be similar.

Properties

- Generalize better than sparse vectors.
- Input for deep learning
  - Fewer weights (or other weights)
- Capture semantic similarities better.
- Better for sequence modelling:
  - Language models, etc.

Word embeddings

- Each word is represented as a vector of reals
- Words are more or less similar
- A word can be similar to one word in some dimensions and other words in other dimensions

Dimensions

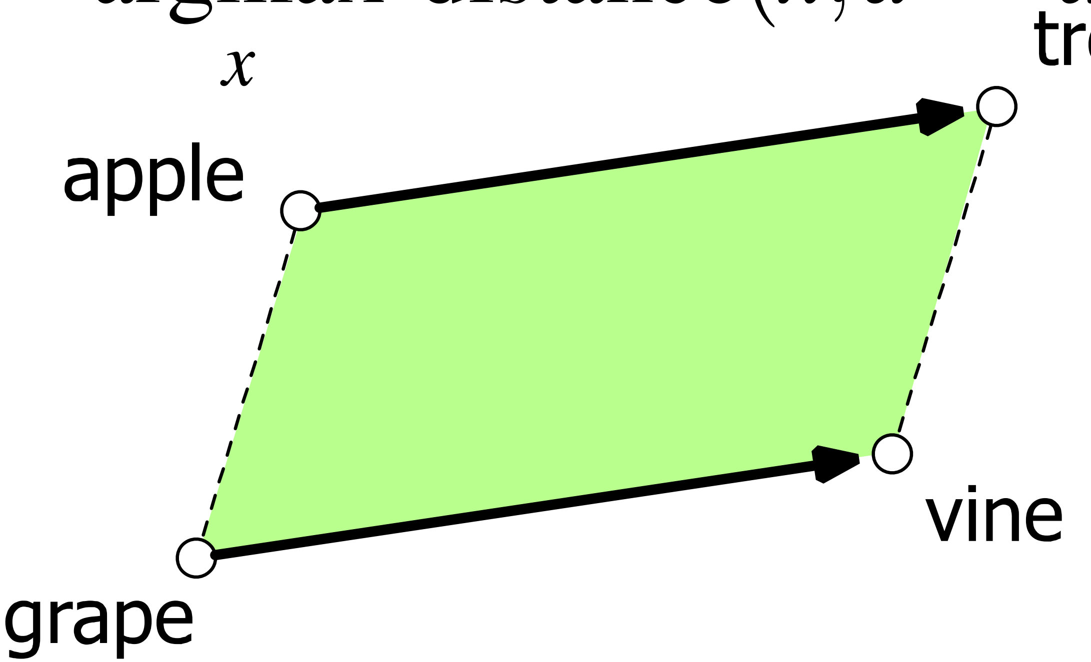
Word vectors	animal	domesticated	pet	fluffy
dog	-0.4	0.37	0.02	-0.34
cat	-0.15	-0.02	-0.23	-0.23
lion	0.19	-0.4	0.35	-0.48
tiger	-0.08	0.31	0.56	0.07
elephant	-0.04	-0.09	0.11	-0.06
cheetah	0.27	-0.28	-0.2	-0.43
monkey	-0.02	-0.67	-0.21	-0.48
rabbit	-0.04	-0.3	-0.18	-0.47
mouse	0.09	-0.46	-0.35	-0.24
rat	0.21	-0.48	-0.56	-0.37

Figure from https://medium.com/@jayeshbahire

Analogical relations

- The classic parallelogram model of analogical reasoning (Rumelhart and Abrahamson 1973)
- For a problem a:a*::b:b*, the parallelogram method is:

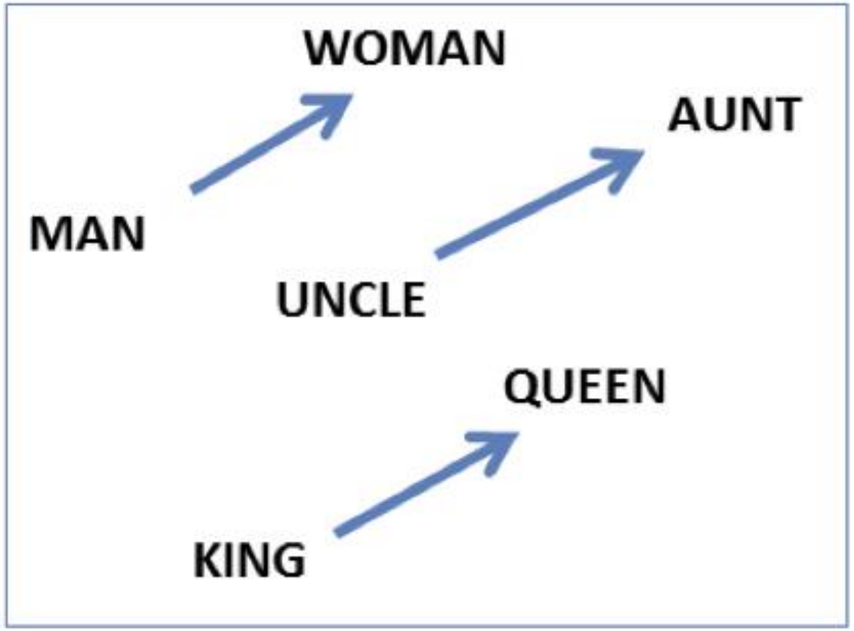
$$\hat{b}^* = \operatorname*{argmax}_x \text{distance}(x, a^* - a + b)$$

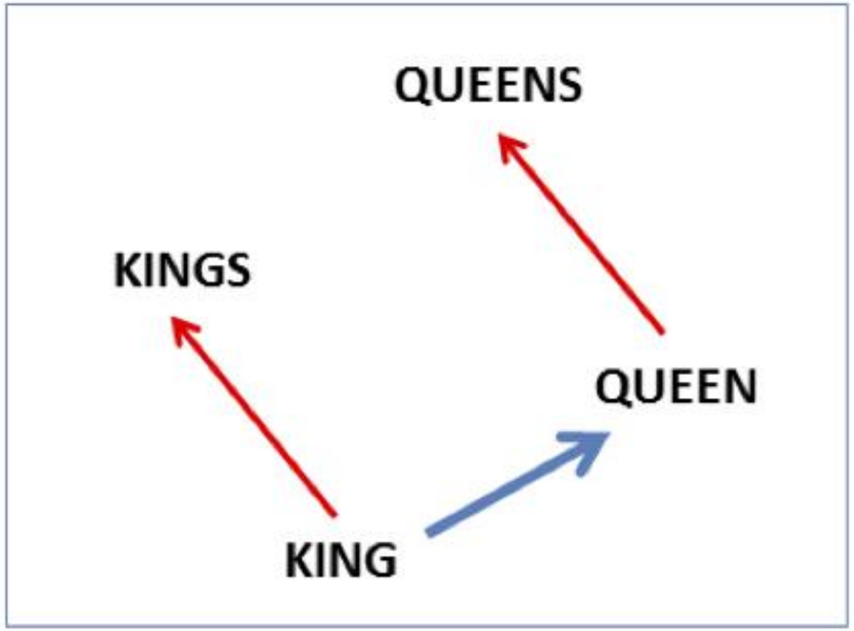


Analogy: Embeddings capture relational meaning!

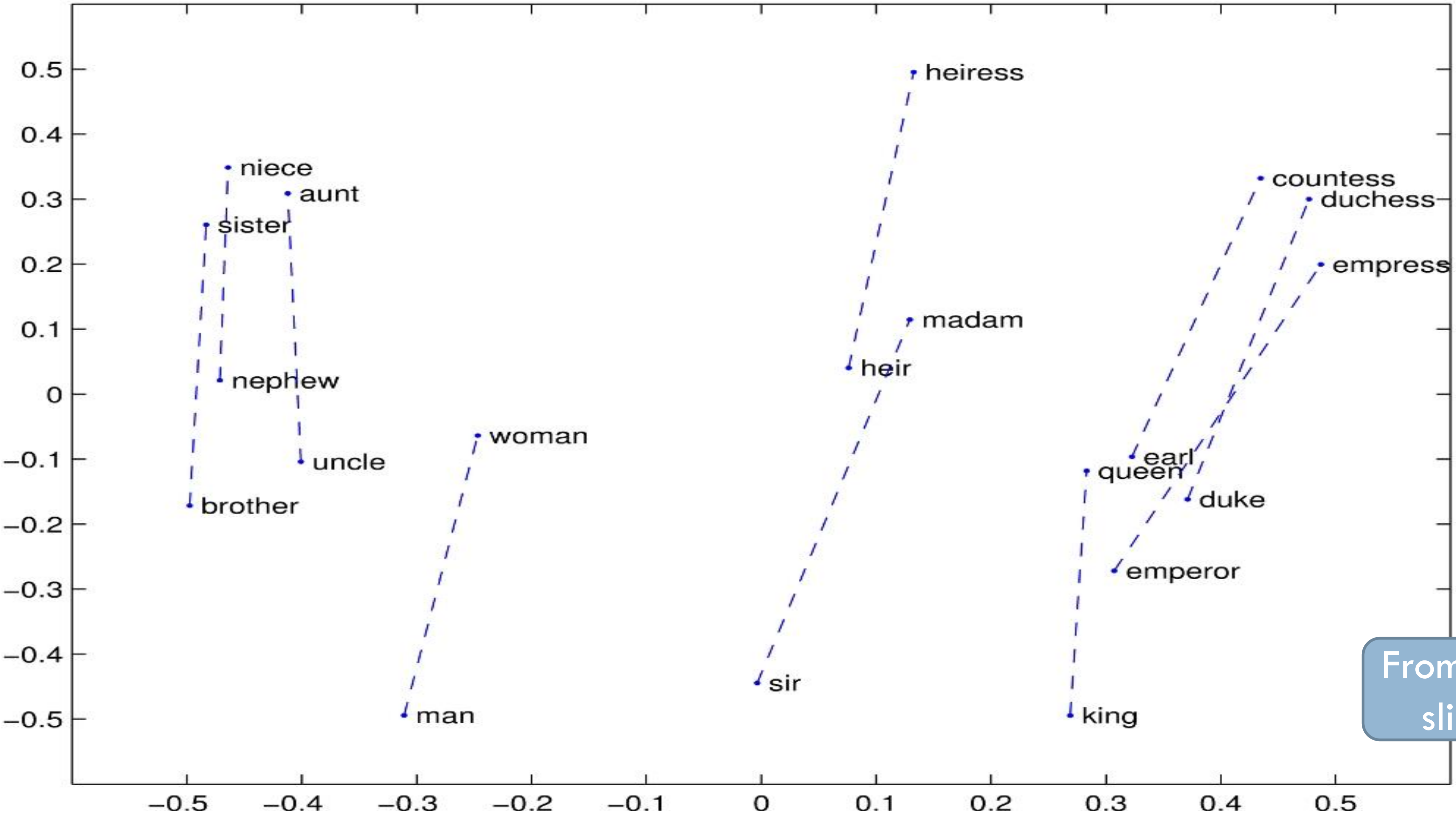
vector('*king*') - vector('*man*') + vector('*woman*') $\approx$ vector('queen')

vector('*Paris*') - vector('*France*') + vector('*Italy*') $\approx$ vector('Rome')

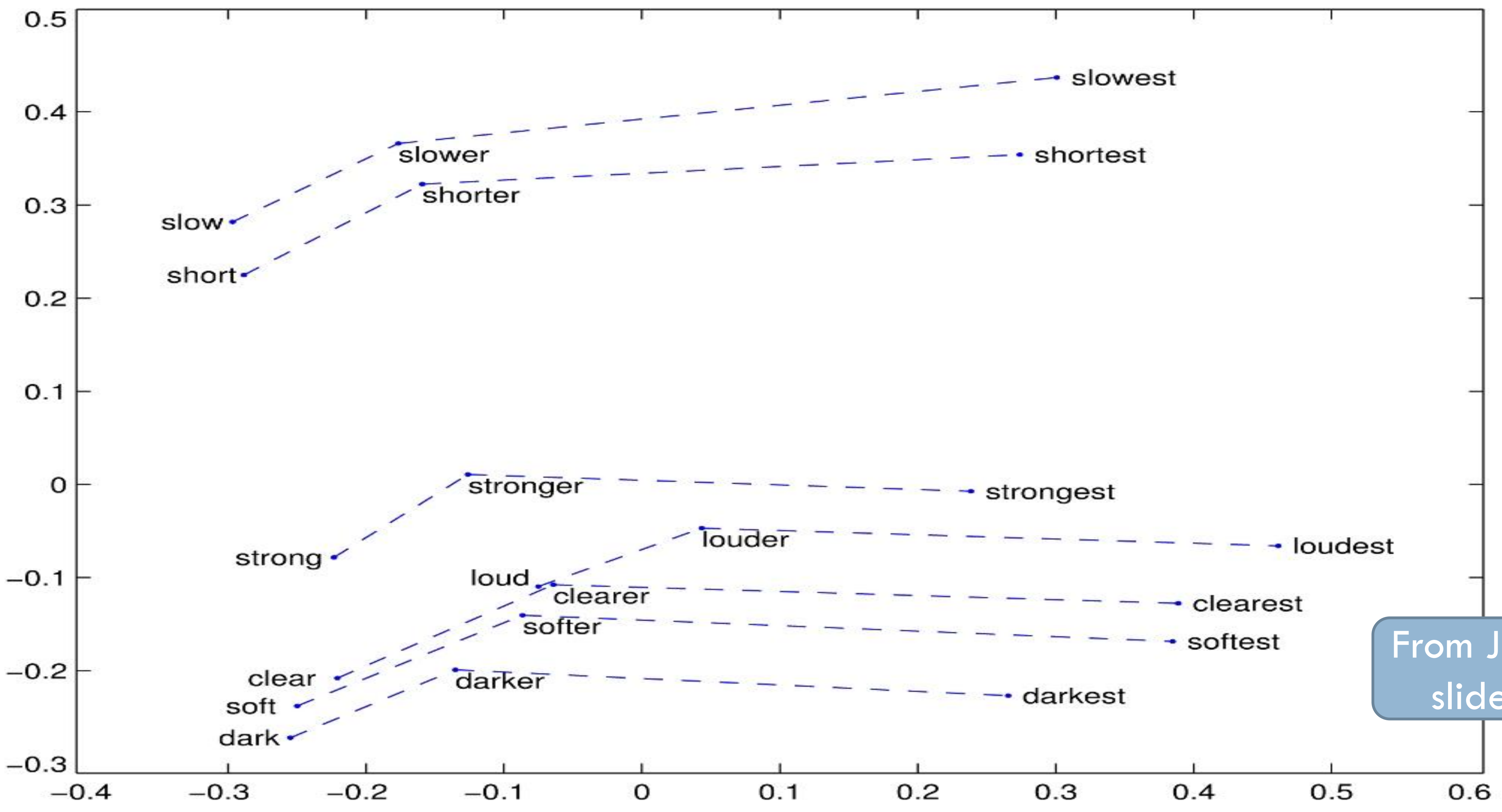




From J&M slides



From J&M slides

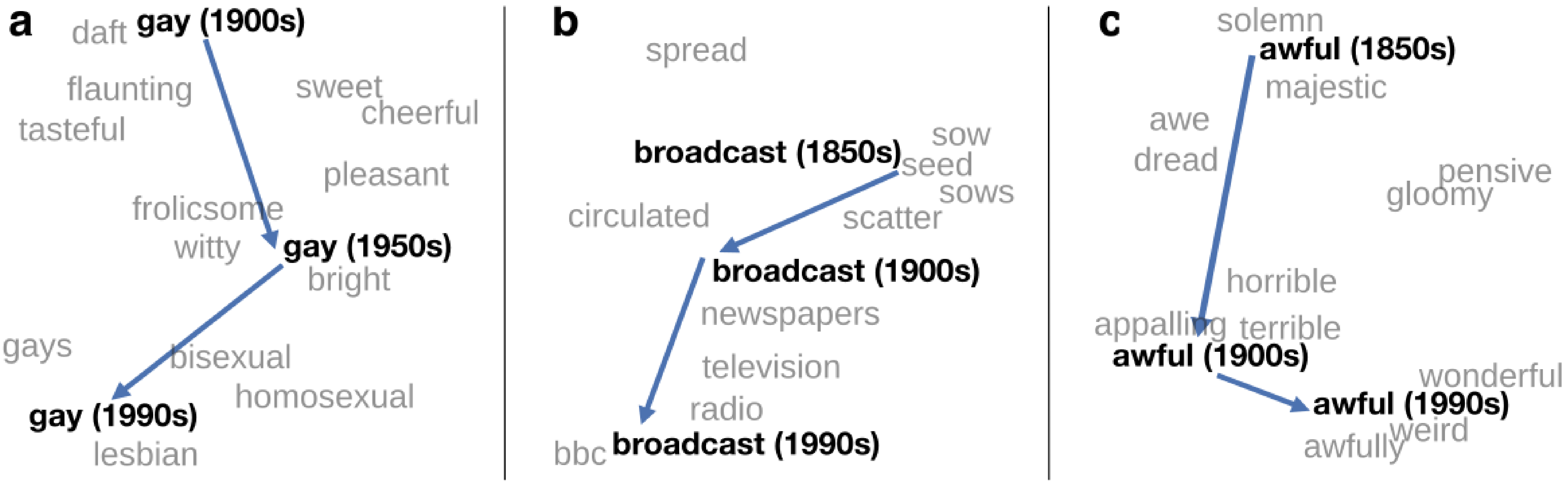


From J&M slides

Demo

- http://vectors.nlpl.eu/explore/embeddings/en/

Track change of meaning of words



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

From J&M slides

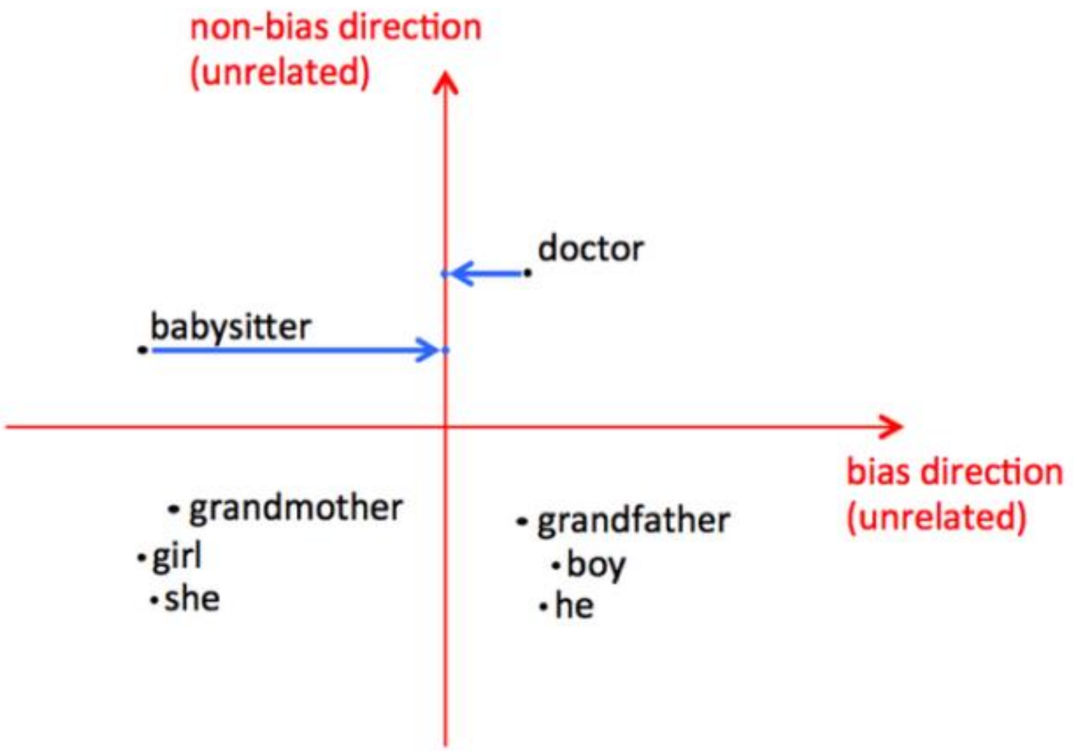
Bias

- *Man is to computer programmer as woman is to homemaker*.
- Different adjectives associated with:
  - male and female terms
  - typical black names and typical white names
- Embeddings may be used to study historical bias

Debiasing (research topic)

- Goal: neutralize the biases
- Some positive results
- But also reports that is is not fully possible

- Is debiasing a goal?
- When should we (not) debias?



https://vagdevik.wordpress.com/2018/07/08/debiasing-word-embeddings/

Evaluation of embeddings

- Extrinsic evaluation:
  - Evaluate contribution as part of an application
- Intrinsic evaluation:
  - Evaluate against a resource
- Some datasets
  - WordSim-353:
    - Broader "semantic relatedness"
  - SimLex-999:
    - Narrower: similarity
    - Manually annotated for similarity

Word1	Word2	POS	Sim-score
old	new	A	1.58
smart	intelligent	A	9.2
plane	jet	N	8.1
woman	man	N	3.33
word	dictionary	N	3.68
create	build	V	8.48
get	put	V	1.98
keep	protect	V	5.4

Part of SimLex-999

Today

- Lexical semantics
- Vector models of documents
- tf-idf weighting
- Word-context matrices
- Word embeddings with dense vectors
- Word2vec

Idea

- Instead of counting, use a neural network to learn a LM
- Simplest form: a bigram model:
  - For a given word w_{i-1}, try to predict the next word w_i
  - i.e. try to estimate $P(w_i|\, w_{i-1})$

Model

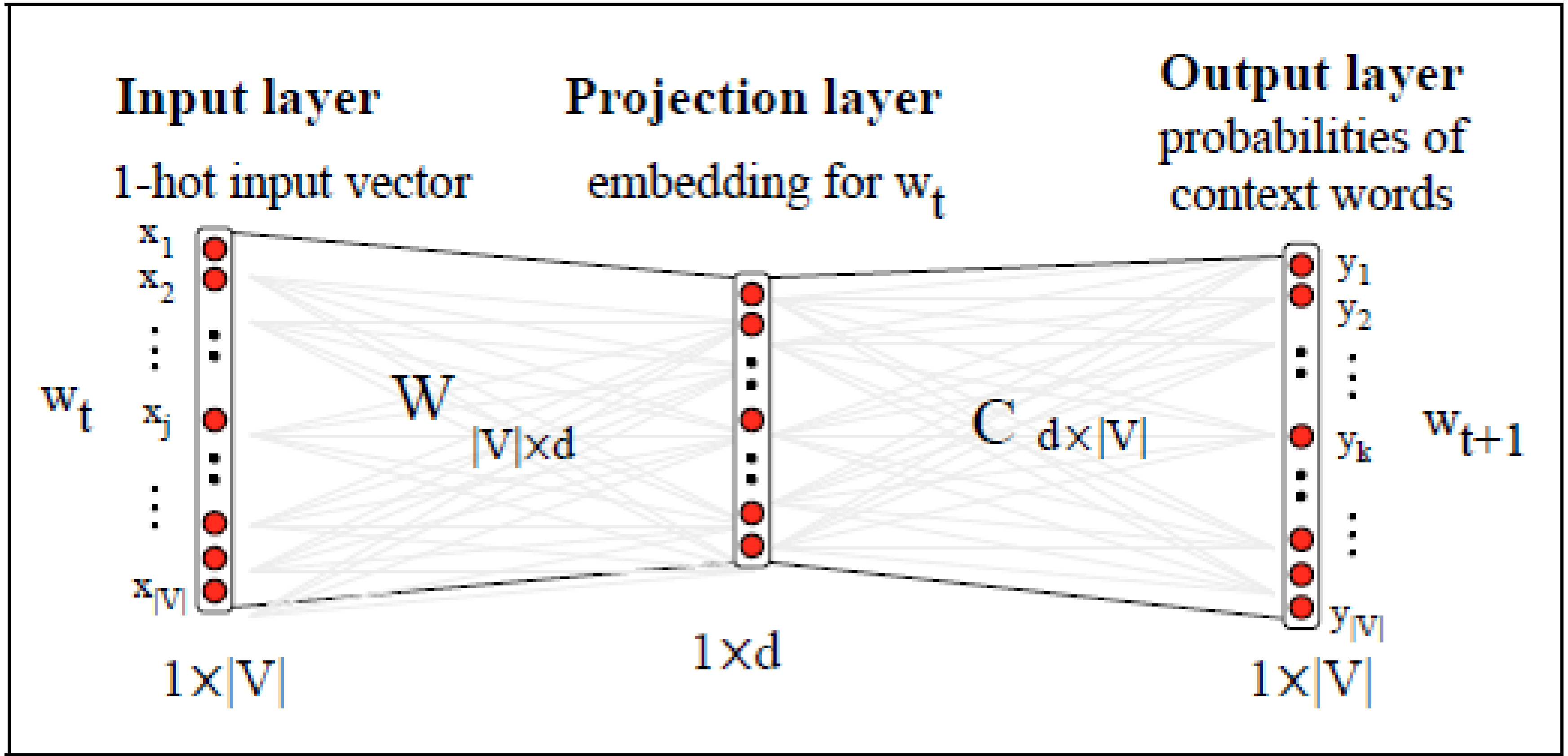
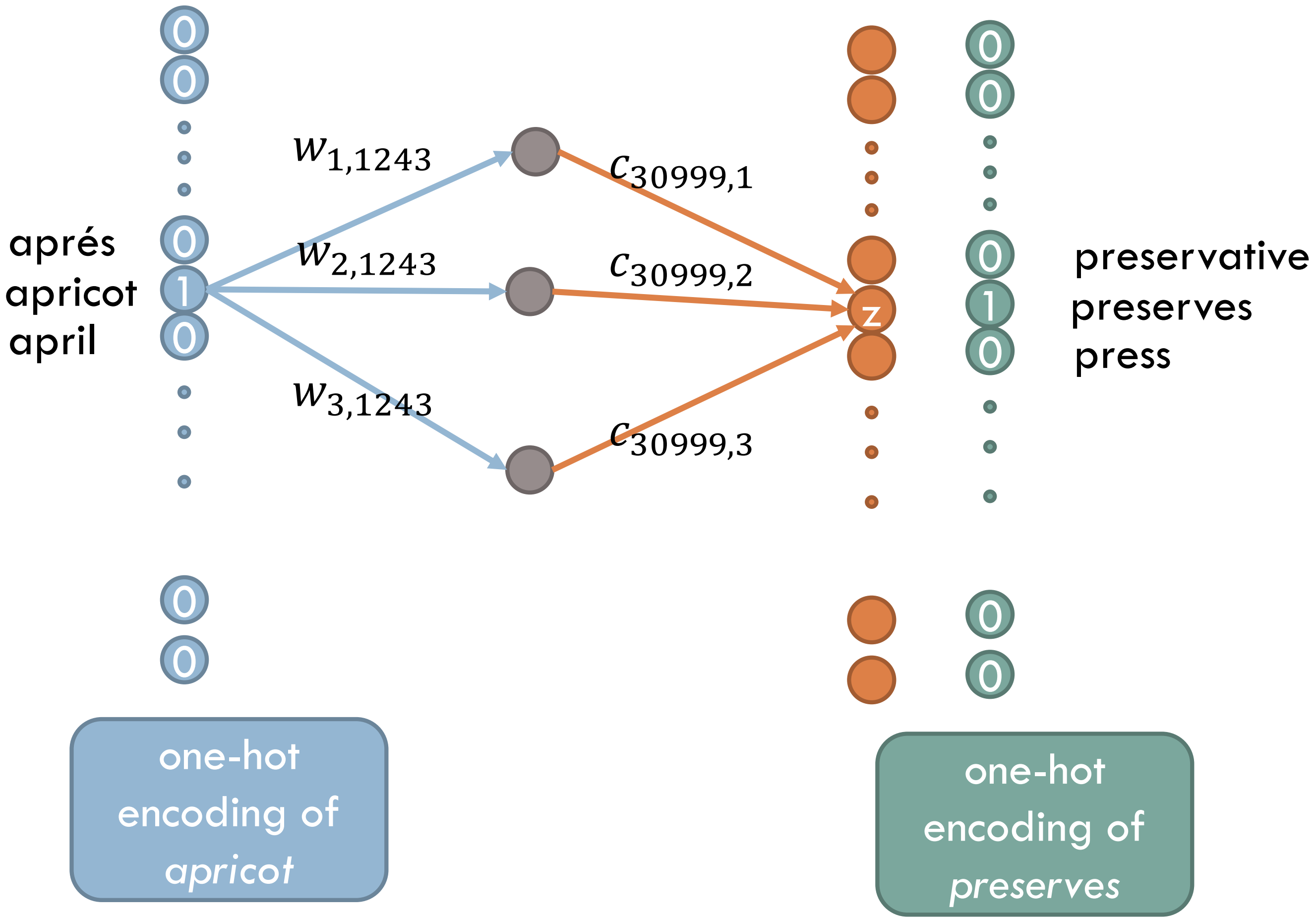


Figure 16.5 The skip-gram model viewed as a network (Mikolov et al. 2013, Mikolov et al. 2013a).

From J&M 3.ed. 2018 Ch. 16

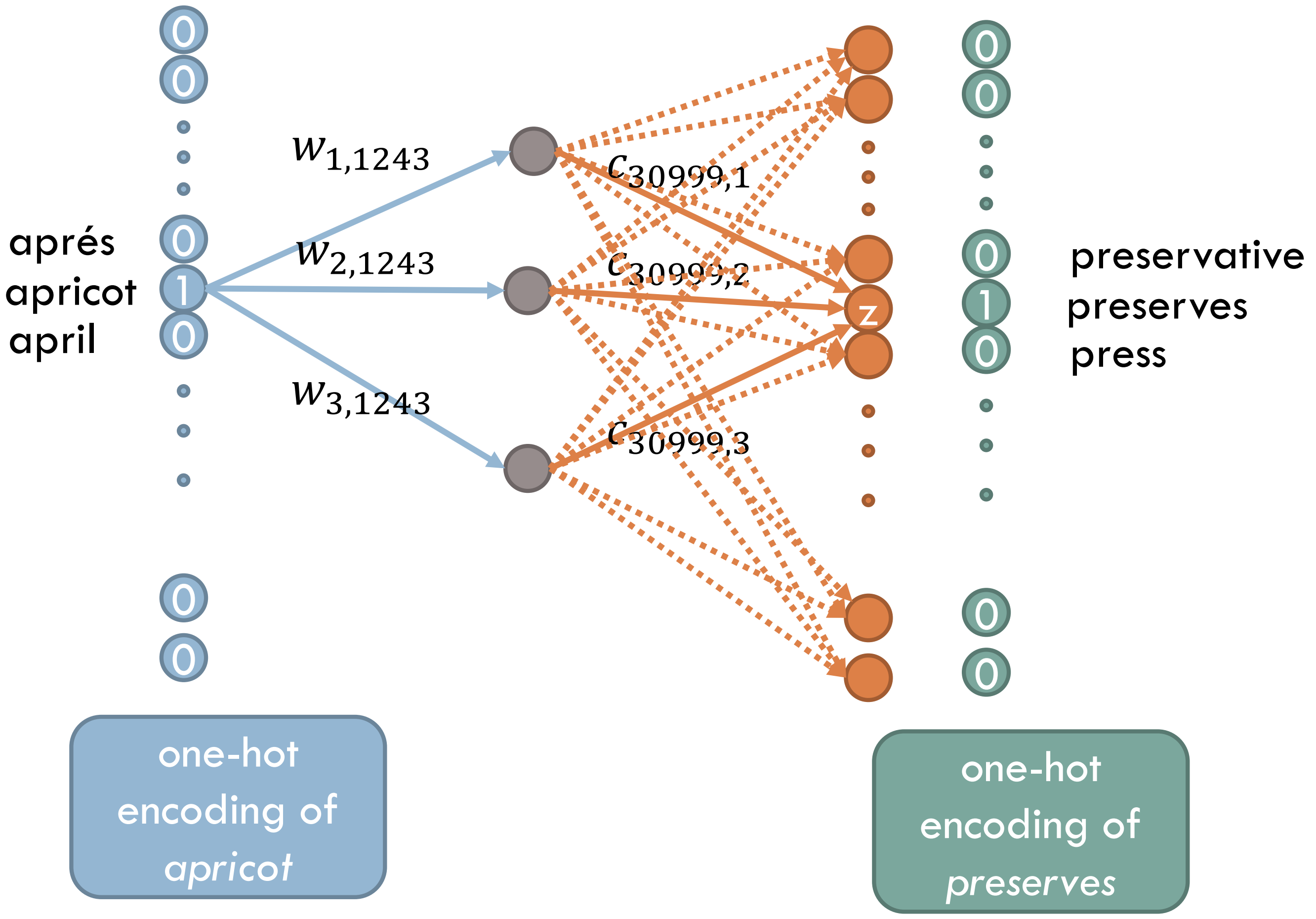
Model: zoom in

- *apricot* is word 1243
  - word-embedding:
  - $\boldsymbol{w} = (w_{1,1243}, \ldots w_{d,1243})$
- *preserves* is word 30999
  - context-embedding:
  - $\boldsymbol{c} = (c_{30999,1}, \ldots c_{30999,d})$
- $z = \boldsymbol{w} \cdot \boldsymbol{c} = \sum_{i=1}^{d} w_{i,1243} c_{i,30999}$



Model

- Input and output word are represented by sparse one-hot vectors
- Dim *d* typically 50-300
- Idea for training:
  - Consider all possible next words for w' for this word
  - Use softmax to get a probability distribution of all next words



Embeddings from this

- Idea: Use the weight matrix $W_{|V|\times d}$ as embeddings, i.e.:
- Represent word j by $(w_{j,1}, w_{j,2}, \ldots, w_{j,d})$ = the weights that sends this word to the hidden layer
- Why? since similar words will predict more or less the same words, they will get similar embeddings

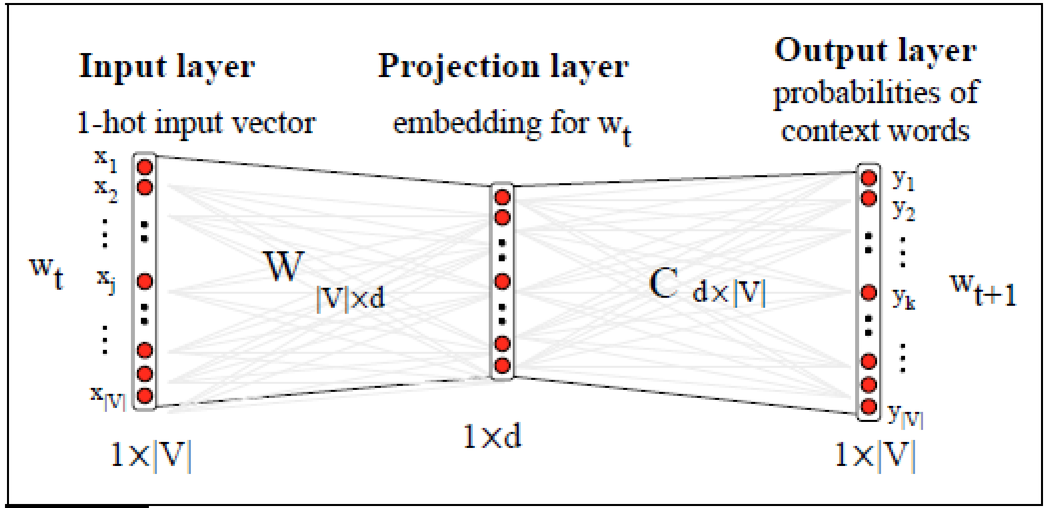


Figure 16.5 The skip-gram model viewed as a network (Mikolov et al. 2013, Mikolov et al. 2013a).

Observations

- Since two words that are similar are predicted by the same words, there will also be similarities between similar words in $C_{d\times|V|}$
- This will help the training of $W_{|V|\times d}$
- We could alternatively use $C_{d\times|V|}$ as the embeddings

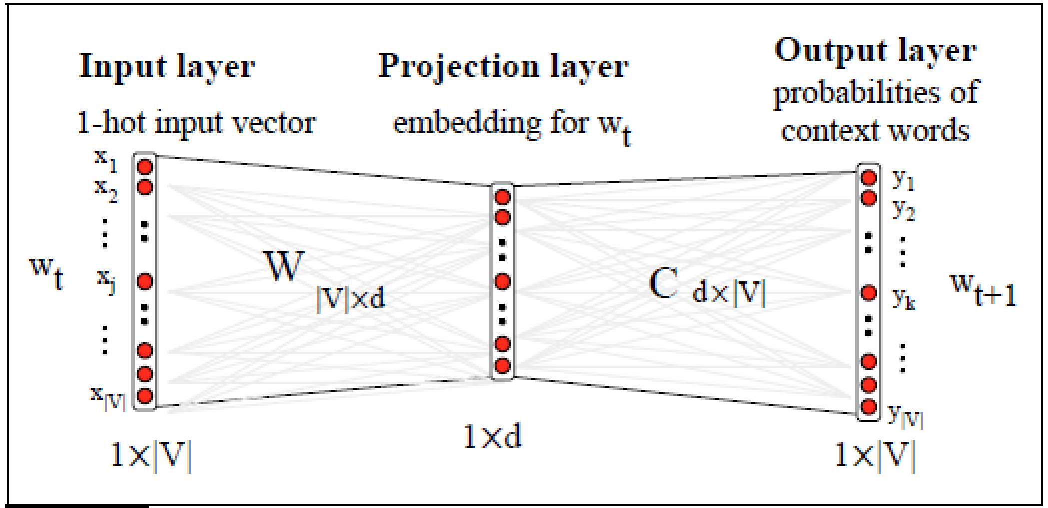
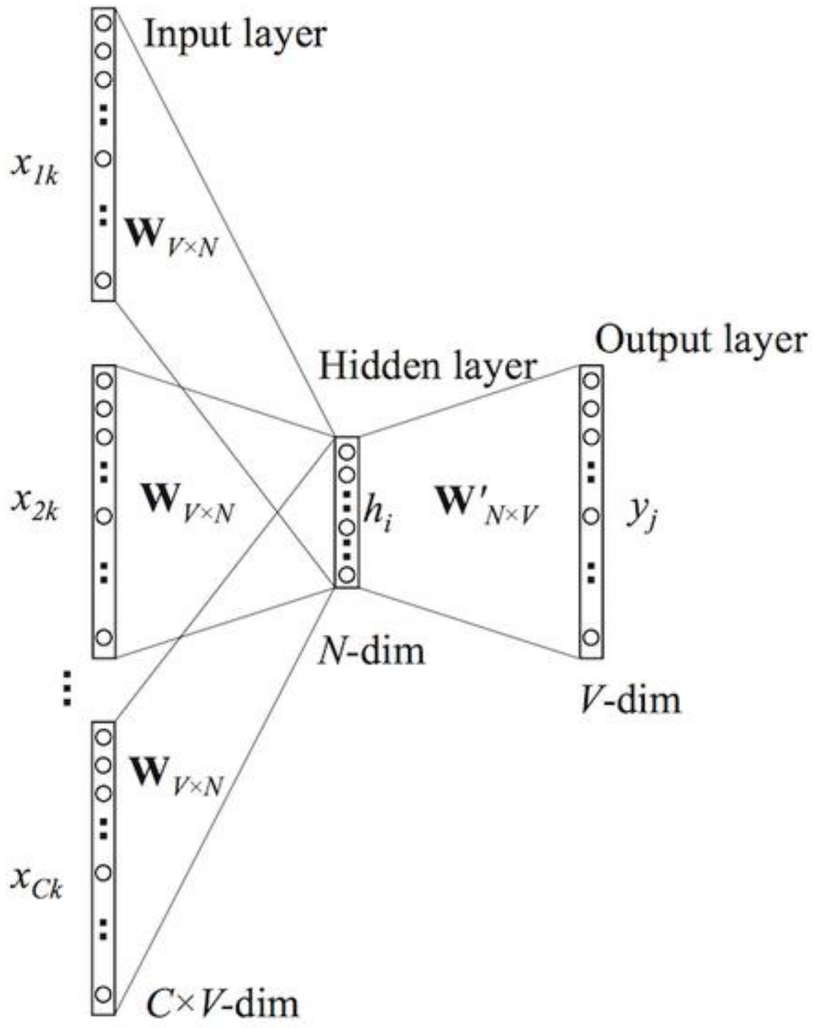


Figure 16.5 The skip-gram model viewed as a network (Mikolov et al. 2013, Mikolov et al. 2013a).

CBOW

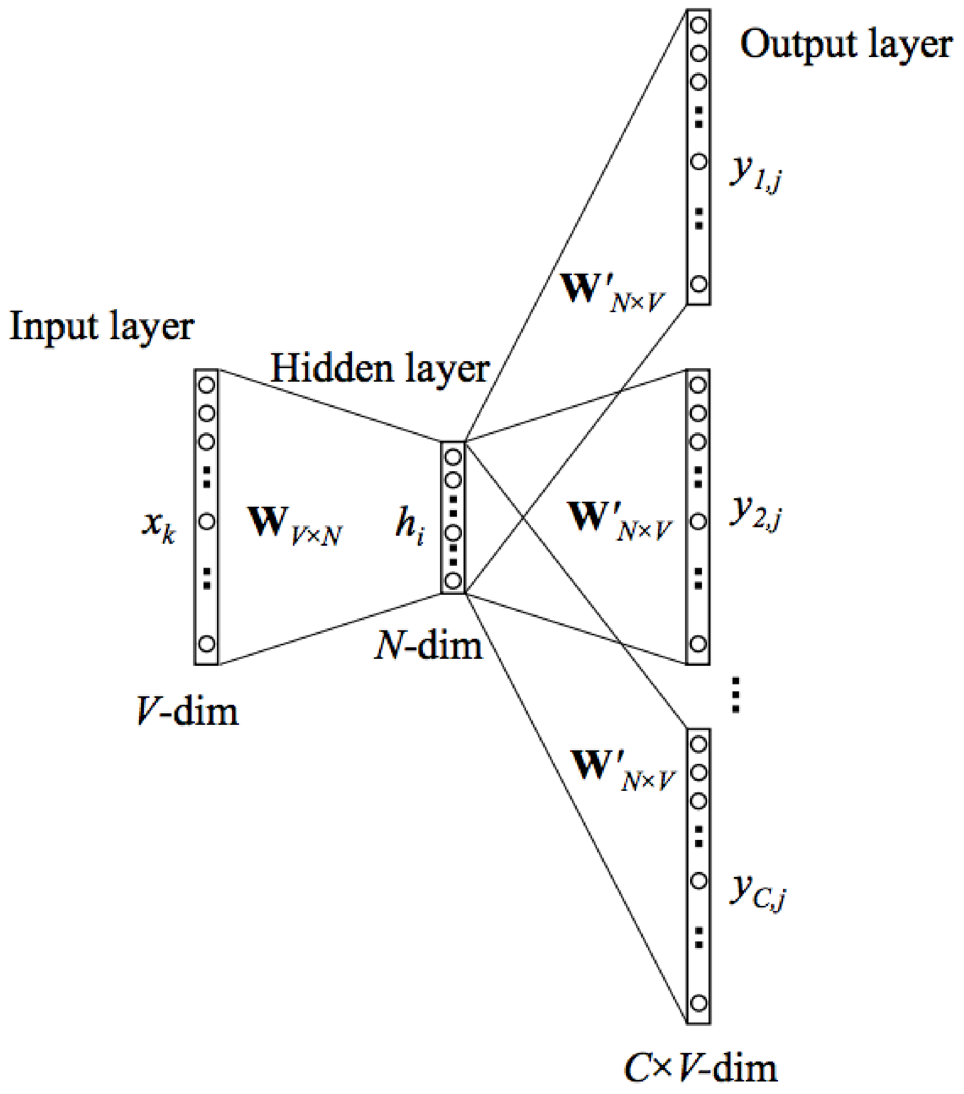
- We could generalize to predicting from a number of preceding words, e.g. 3, as indicated in the figure.
- Observe this is order-independent
- Continuous bag of words model (CBOW):
  - Predict w_t from a window $(w_{t-k}, \dots, w_{t-1}, w_{t+1}, \dots, w_{t+k})$



https://commons.wikimedia.org/wiki/File:Cbow.png

Skip-gram

- From w_t predict all the words in a window $(w_{t-k}, \dots, w_{t-1}, w_{t+1}, \dots, w_{t+k})$
- Assume independence of the context words, i.e. from w_t predict each of the words w in $\{w_{t-k}, \dots, w_{t-1}, w_{t+1}, \dots, w_{t+k}\}$
- The size of the window will influence which embeddings you get



https://commons.wikimedia.org/wiki/File:Skip-gram.png

Skip-gram model

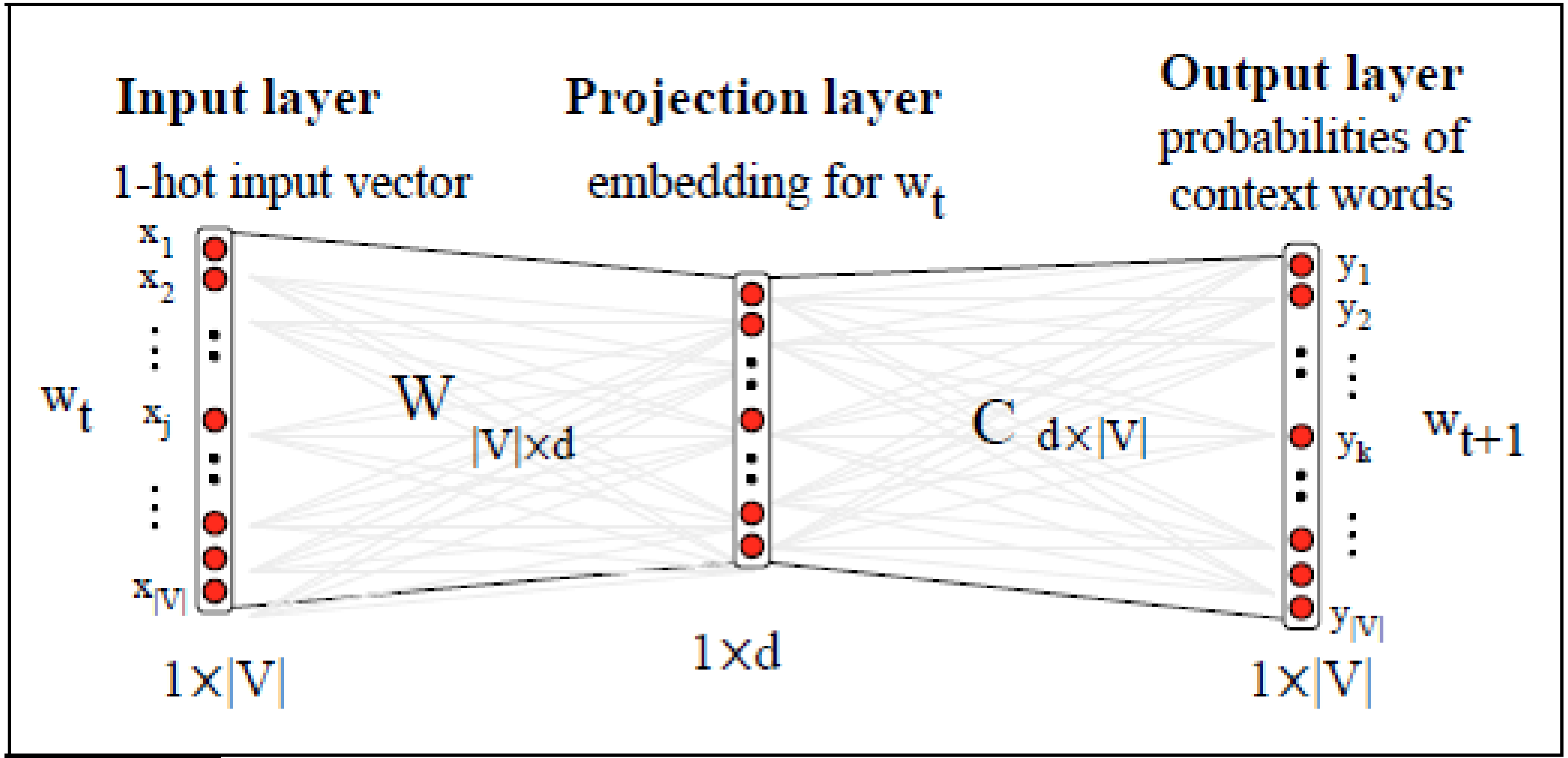
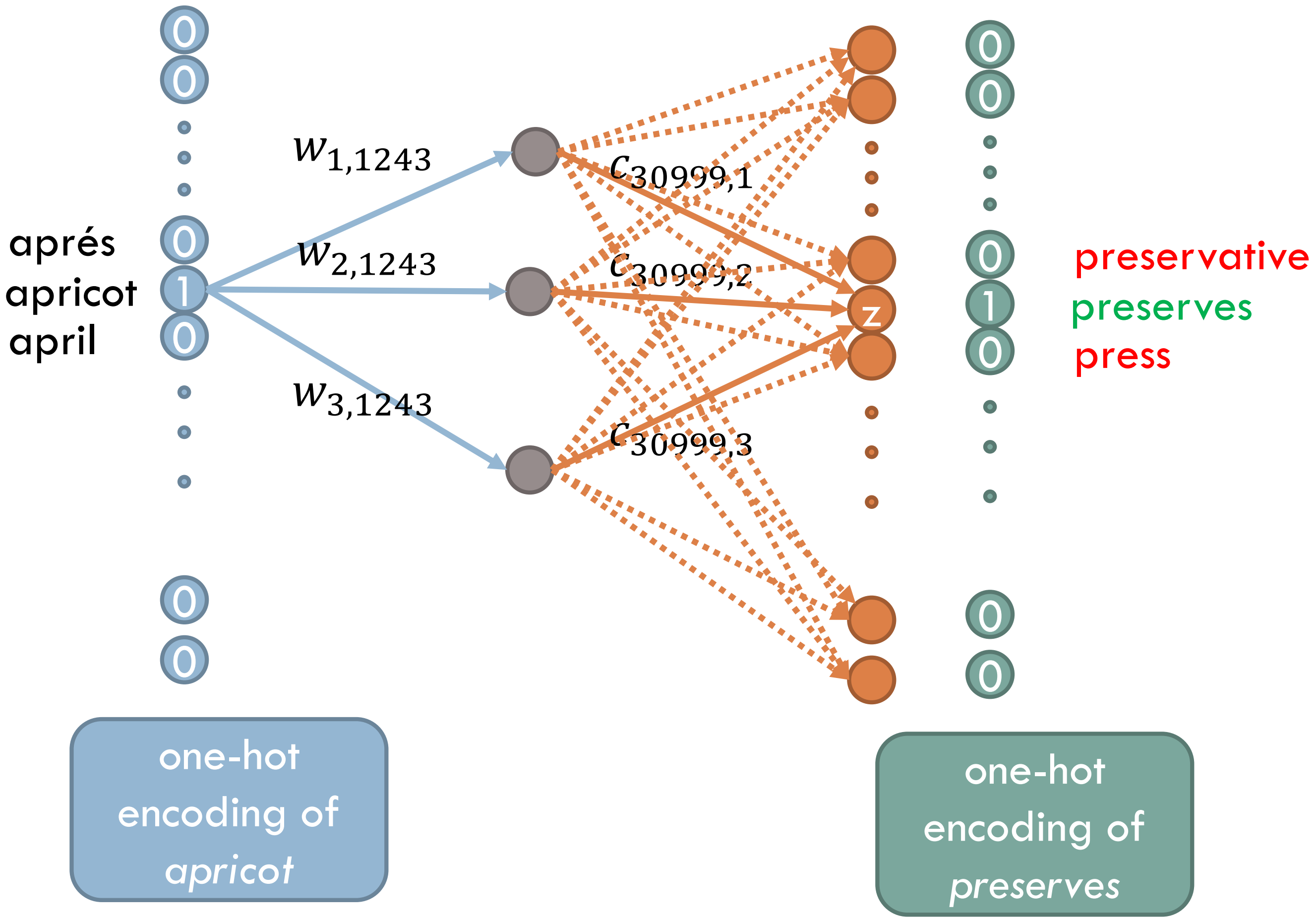


Figure 16.5 The skip-gram model viewed as a network (Mikolov et al. 2013, Mikolov et al. 2013a).

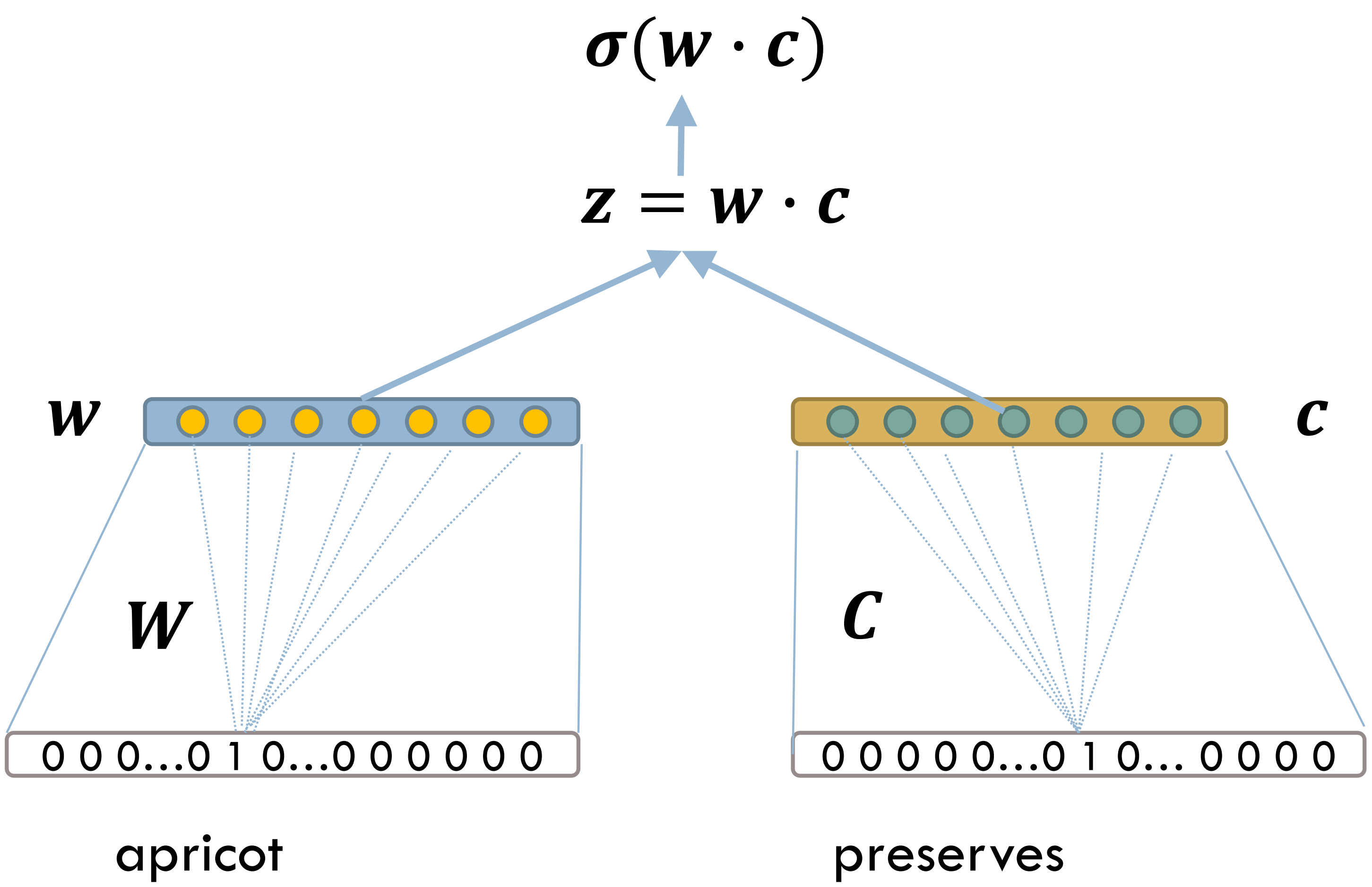
From J&M 3.ed. 2018 Ch. 16

Softmax is expensive

- The use of softmax is expensive
- For one observation, *apricot* *preserves*, one must change all the $c_{i,j}$-s to
  - increase the probability for *preserves*
  - decrease all the other probabilities
- $d \times |C|$, say $300 \times 50{,}000$



Prediction as classification



- To predict preserves from apricot, corresponds to a classification task where
  - class(apricot, preserves)=+
  - class(apricot, w)= – for all other w

Skip-gram with negative sampling

1. Treat the target word and a neighboring context word as a positive example.
2. Randomly sample other words in the lexicon to get negative samples
   - sample accordance to frequency
   - adjusted for high-frequent and low-frequent words: $P_\alpha(w) = \frac{count(w)^\alpha}{\sum_{w'} count(w')^\alpha}$
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 preserves or a ...
- c1 c2 t c3 c4

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

Skip-Gram Training Data

- ... lemon, a tablespoon of apricot preserves or a ...
- c1 c2 t c3 c4
- For each positive example, we'll create *k* negative examples.
  - Using *noise* words: Any random word that isn't *t*

positive examples +

t	c
apricot	tablespoon
apricot	of
apricot	preserves
apricot	or

negative examples -

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

Learning

- Like Logistic Regression
- Start with randomly initialized weights for W and C
- For the training items (w, c), calculate $\hat{y} = \sigma(\boldsymbol{c} \cdot \boldsymbol{w}) = \frac{1}{1+e^{-\boldsymbol{c}\cdot\boldsymbol{w}}}$
- Compare to the gold labels using cross-entropy loss
  - The gold label is 1 if c is a context word and 0 if c is a negative example
  - This is like Logistic regression
- Use the derivative of the loss with respect to **c**: $\frac{\partial}{\partial \boldsymbol{c}} Lce$ to update **c**
- and the derivative of the loss with respect to **w** to update **w**

Update equations in SGD

- We skip the derivation, but these are the resulting update equations

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

- $\hat{y} = \sigma(\boldsymbol{c} \cdot \boldsymbol{w})$
- Similar to the logistic regression, where we update weights
- Her we update both the w-s and the c-s.

Result

- We learn two separate embedding matrices W and C
- We can use W as representations for the words
  - (or combine with C in some ways)

- What have we learned:
  - If two words *w1* and *w2* occur in similar contexts
    - = with the same (or similar) context words, e.g. c,
  - then both *w1* and *w2* should have a large cosine with c,
    - hence get similar vectors.

Use of embeddings

- Embeddings are used as representations for words as input in all kinds of NLP tasks using deep learning:
  - Text classification
  - Language models
  - Named-entity recognition
  - Machine translation
  - etc.
- These embeddings are nowadays called static
- Since 2018, Transformers:
  - The embedding of each word depends on the context
  - Superior results in all tasks
- IN5550, Spring

Resources

- gensim
  - Easy-to-use tool for training own models
- Word2wec
  - https://code.google.com/archive/p/word2vec/
- https://fasttext.cc/
- https://nlp.stanford.edu/projects/glove/
- http://vectors.nlpl.eu/repository/
  - Pretrained embeddings, also for Norwegian