NLP

Deep Learning

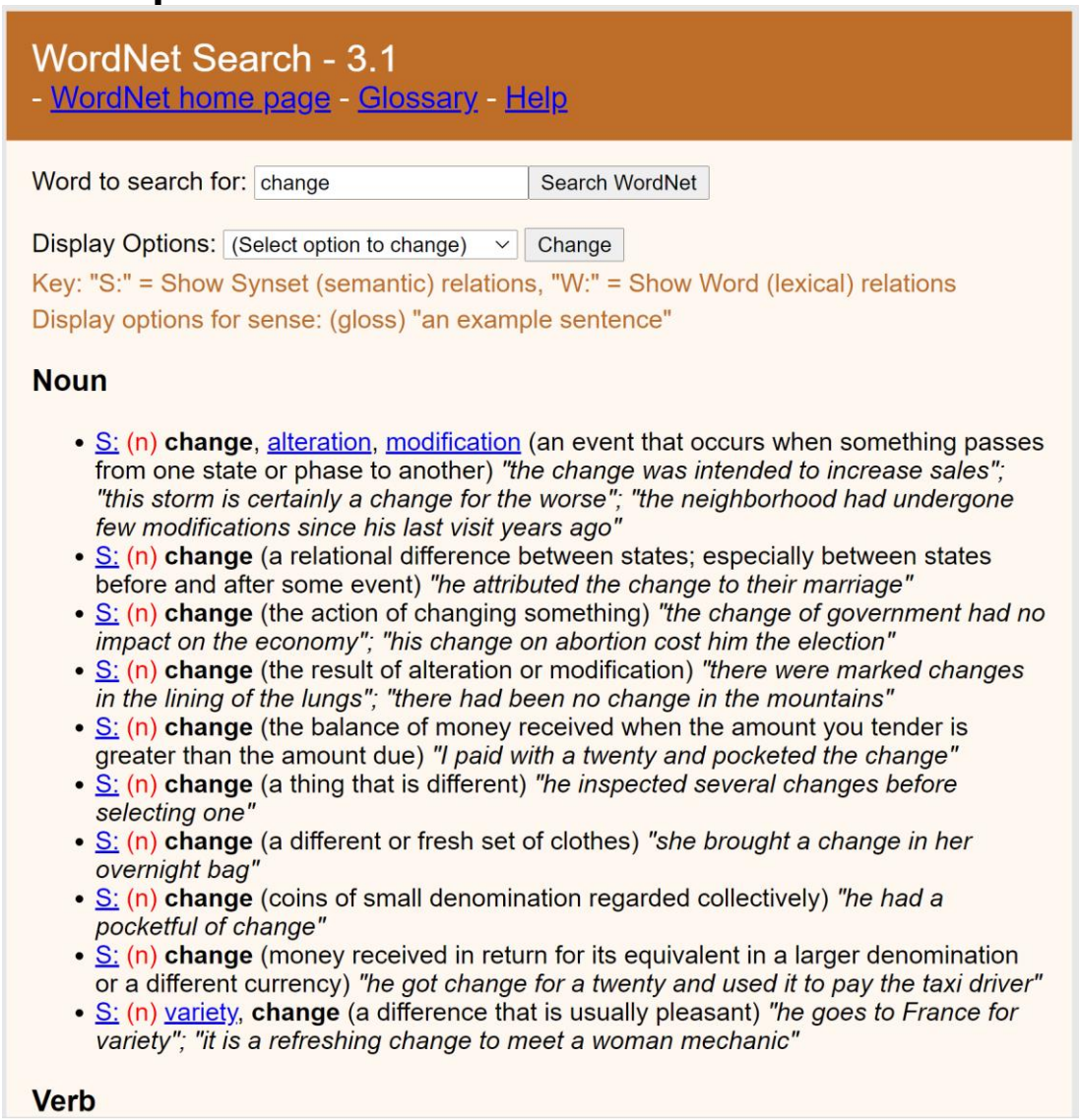
721.

1. Intro to Word Embeddings

Representing word meaning

- Image?
- Denotational Semantics
  - Signifier (symbol), Signified (idea or thing)
- Dictionary definitions
  - Webster's
  - Wiktionary
  - WordNet
- Issues with dictionaries
  - Incomplete synonyms (e.g., "spicy" as synonym of "hot")
  - Out of date (new words get created all the time)
  - Subjective
  - Manually created
  - Hard to build, esp. for different languages or domains
  - Hard to compute word similarity

WordNet example



WordNet Search - 3.1
- WordNet home page - Glossary - Help

Word to search for: change Search WordNet

Display Options: (Select option to change) Change

Key: "S:" = Show Synset (semantic) relations, "W:" = Show Word (lexical) relations

Display options for sense: (gloss) "an example sentence"

Noun

- S: (n) **change**, alteration, modification (an event that occurs when something passes from one state or phase to another) *"the change was intended to increase sales"; "this storm is certainly a change for the worse"; "the neighborhood had undergone few modifications since his last visit years ago"*
- S: (n) **change** (a relational difference between states; especially between states before and after some event) *"he attributed the change to their marriage"*
- S: (n) **change** (the action of changing something) *"the change of government had no impact on the economy"; "his change on abortion cost him the election"*
- S: (n) **change** (the result of alteration or modification) *"there were marked changes in the lining of the lungs"; "there had been no change in the mountains"*
- S: (n) **change** (the balance of money received when the amount you tender is greater than the amount due) *"I paid with a twenty and pocketed the change"*
- S: (n) **change** (a thing that is different) *"he inspected several changes before selecting one"*
- S: (n) **change** (a different or fresh set of clothes) *"she brought a change in her overnight bag"*
- S: (n) **change** (coins of small denomination regarded collectively) *"he had a pocketful of change"*
- S: (n) **change** (money received in return for its equivalent in a larger denomination or a different currency) *"he got change for a twenty and used it to pay the taxi driver"*
- S: (n) variety, **change** (a difference that is usually pleasant) *"he goes to France for variety"; "it is a refreshing change to meet a woman mechanic"*

Verb

https://wordnet.princeton.edu/

Representing word meaning

- Discrete symbols in traditional NLP
- Highly dimensional one-hot vectors

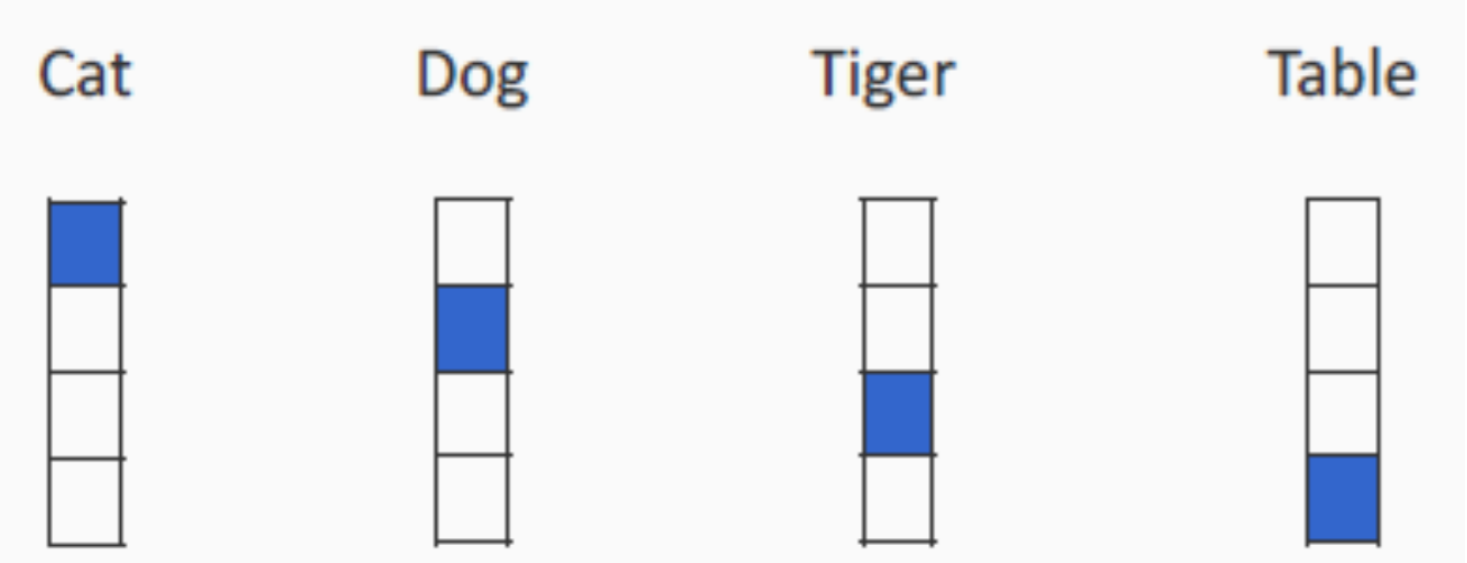
  Cat = [0 0 0 0 0 0 **1** 0 0 0 0 0 0 … 0]

  Kitten = [0 0 0 0 0 0 0 0 0 0 **1** 0 0 … 0]
- Problems with one-hot vectors
  - The vectors for “cat” and “kitten” are orthogonal
  - No practical notion of similarity
- Using word context
  - Distributional vectors
  - See earlier lecture

Representing word meaning

- Dense vectors (word embeddings)
  - distributed representations
- Co-occurrence counts
  - Window around the word
  - LSA/SVD (earlier lecture)
  - Very high space requirements
- Learning the embeddings
  - GloVe
  - Word2Vec

One-hot vs. distributed representations



[VS]

Word Embedding Example

pharmaceuticals

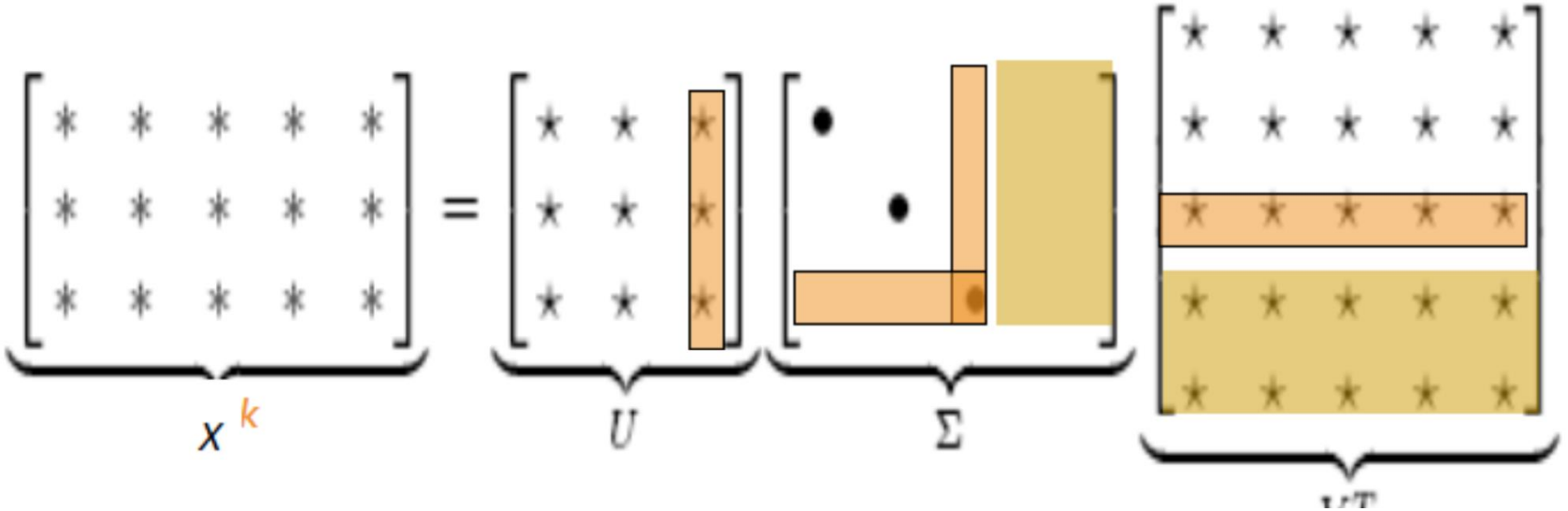
[-0.43337 0.10411 -0.49926 -0.06442 0.39005 0.65823 -1.3033 -0.39019 -0.41713 -0.038667 -0.37677 0.35303 -0.16425 -0.46671 -0.24358 0.10142 0.54664 0.065344 0.022733 -0.10422 -0.52839 0.20096 -0.23819 0.28856 0.098377 -0.12174 -0.035851 -0.92717 0.28408 0.22439 0.027598 0.47919 -0.51142 -0.062696 0.13002 0.20366 -0.050135 -0.4593 0.63779 -0.56294 -0.40989 -0.60342 0.8151 0.024929 -0.0096054 0.55748 0.35156 0.14476 -0.36327 -0.34856 0.1787 0.76398 0.27803 0.11999 0.42814 0.17844 0.092123 0.058811 -0.41114 -0.12101 -0.34001 -0.49946 -0.073549 0.15371 0.18034 0.34176 0.072738 -0.23442 -0.023682 -0.2499 -0.17334 -0.15008 -0.14599 0.51706 0.52797 -0.075328 0.13018 -0.069498 -0.15378 -0.31615 0.59434 -0.91396 -0.12803 0.32963 0.70337 -0.095882 -0.37066 0.16993 -0.62115 -0.76234 0.49005 -0.026823 -0.35171 -0.070227 0.19778 0.25563 -1.4504 -0.47122 -0.10107 -0.18279 -0.31553 0.090524 0.1975 0.073745 0.34809 -0.26728 -0.04808 -0.18467 0.18147 0.37255 0.26197 -0.0046708 0.51 -0.99408 -0.1942 -0.82518 0.59211 0.31112 0.3472 -0.066567 0.65975 -0.52254 -0.48302 0.30366 -0.35524 0.0022488 -0.89521 -0.096487 -0.36811 -0.1139 -0.039127 0.03701 0.18691 -0.28874 0.19926 -0.71229 0.21108 0.1768 0.27541 -0.72828 -0.74097 0.15007 -0.46696 0.52759 0.69806 0.28434 1.2781 0.033105 0.2153 -0.59069 -0.18089 -0.28775 -0.30792 -0.32764 -0.20838 -0.49774 0.2604 -0.26116 -0.29203 -0.18311 0.016024 0.26013 -0.4441 0.11857 0.61598 -0.25685 -0.49715 0.58277 -0.093157 -0.078187 -0.18587 -0.021307 0.52742 0.75704 0.091185 -0.41006 -0.26896 0.43715 0.13183 -0.49085 -0.84639 -0.22379 -0.094786 0.35858 -1.16 -0.019064 -0.29052 0.21588 0.026218 0.22063 -0.64061 0.89117 -0.14541 -0.47563 0.77044 -0.45668 0.49585 0.45303 -0.24904 0.24502 0.42608 0.0077214 -0.55742 0.17449 -0.2142 0.26996 -0.26239 0.18933 -0.66798 -0.004951 0.062785 0.45616 -0.77372 0.29266 -0.76515 0.2079 -0.52916 -0.13621 -0.60588 -0.049171 -0.21234 -0.071004 0.092045 0.87973 -1.0929 -0.028515 -0.28424 0.26105 0.2524 0.35996 -0.74328 -0.27066 -0.046789 0.081494 0.70996 0.18443 -0.10652 -0.32352 -0.026315 0.079707 0.14203 -0.21906 0.49254 -0.29718 0.25746 -1.108 -0.35566 -0.14188 -0.35727 -0.34243 -0.37111 1.0034 -0.13679 0.92309 -0.16716 -0.36536 0.019904 -0.33768 0.18646 0.4391 0.045023 -0.25726 -0.14073 0.77022 -0.18926 -0.27791 0.2978 -0.78997 0.052217 0.72518 0.0089245 0.1029 1.0213 -0.70057 0.31295 0.29313 -0.53556 0.75132 0.29351 -0.70482 -0.61882 -0.33732 0.60293 -0.33575 -0.12536 0.27659 -0.20361 0.12683 0.10469 -0.47956 0.187 0.38118 0.16238 -0.0484 0.43112 0.0089624 0.0051162 -0.67922 0.1709 -0.020472]

[Glove 300d vector]

Dimensionality Reduction

Singular Value Decomposition of co-occurrence matrix X

Factorizes X into $U\Sigma V^T$, where U and V are orthonormal



Retain only k singular values, in order to generalize.

$\hat{X}$ is the best rank k approximation to X , in terms of least squares.

Classic linear algebra result. Expensive to compute for large matrices.

[Chris Manning]

Corpus:
I like deep learning. I like NLP. I enjoy flying.

In [15]:
```
import numpy as np
la = np.linalg
words = ["I","like","enjoy","deep","learning","NLP","flying","."]
import matplotlib.pyplot as plt
```

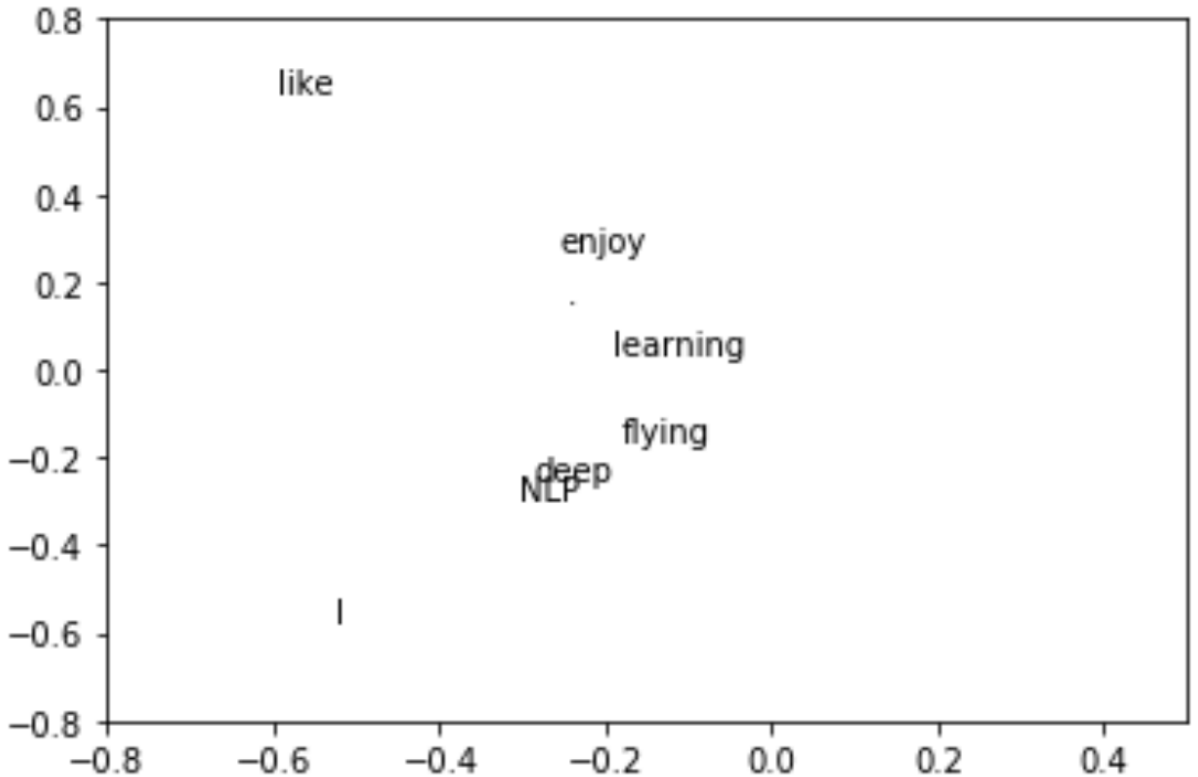
In [21]:
```
X = np.array ([[0,2,1,0,0,0,0,0],
               [2,0,0,1,0,1,0,0],
               [1,0,0,0,0,0,1,0],
               [0,1,0,0,1,0,0,0],
               [0,0,0,1,0,0,0,1],
               [0,1,0,0,0,0,0,1],
               [0,0,1,0,0,0,0,1],
               [0,0,0,0,1,1,1,0]])
U, s, Vh = la.svd(X, full_matrices=False)
```

Relation = bigrams

[Richard Socher and Chris Manning]

Corpus: I like deep learning. I like NLP. I enjoy flying.
Printing first two columns of U corresponding to the 2 biggest singular values

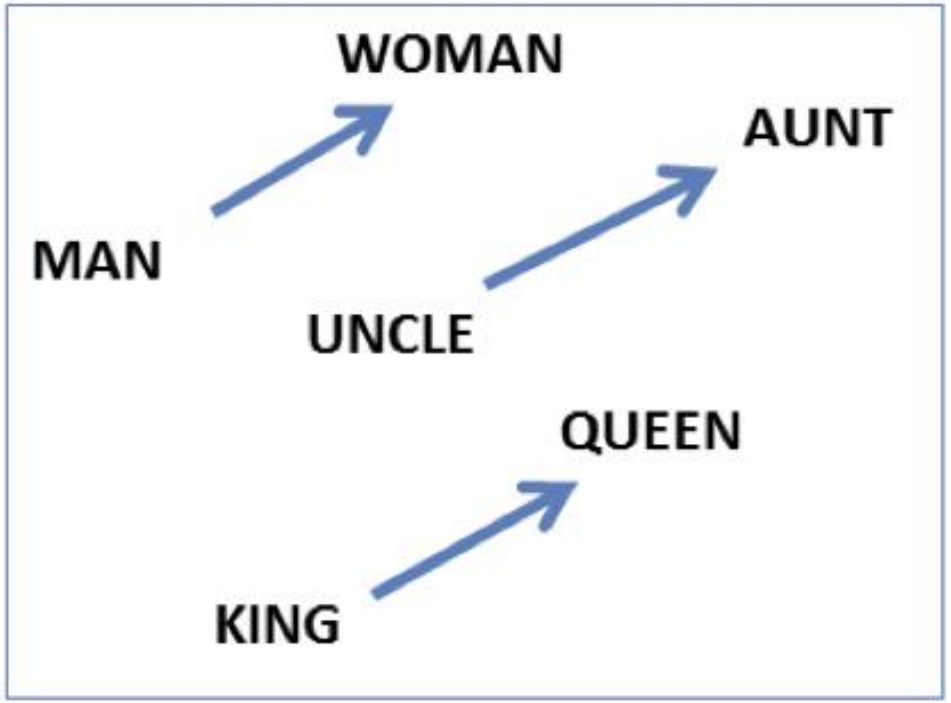
```
In [23]: plt.xlim(-.8,.5)
         plt.ylim(-.8,.8)

         for i in range(len(words)):
             plt.text(U[i,0],U[i,1],words[i])
```



[Richard Socher and Chris Manning]

Embeddings Are Magic, Part 1

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



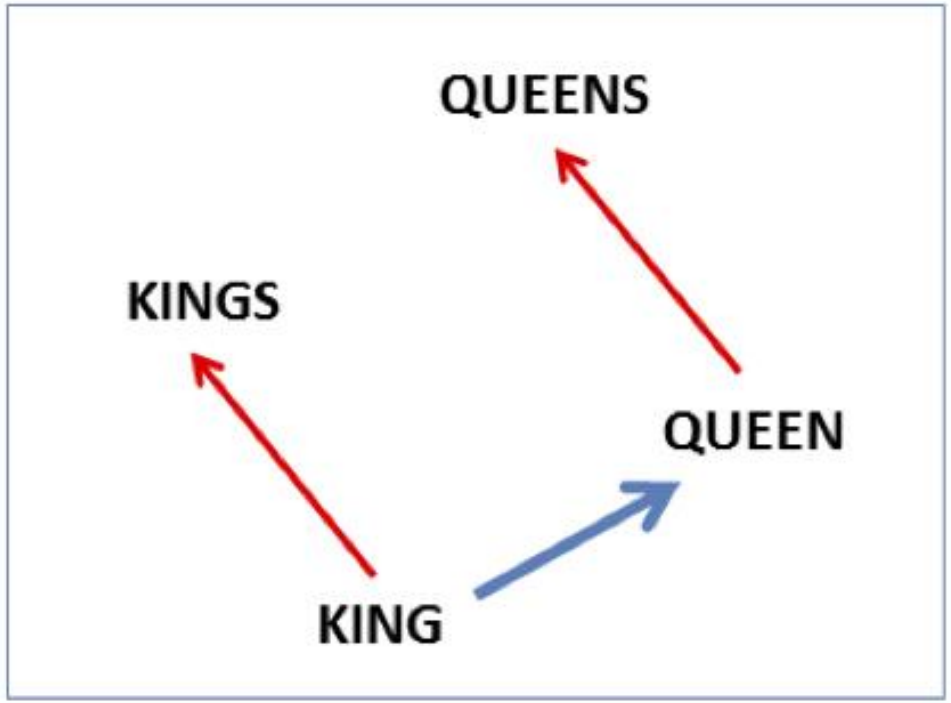
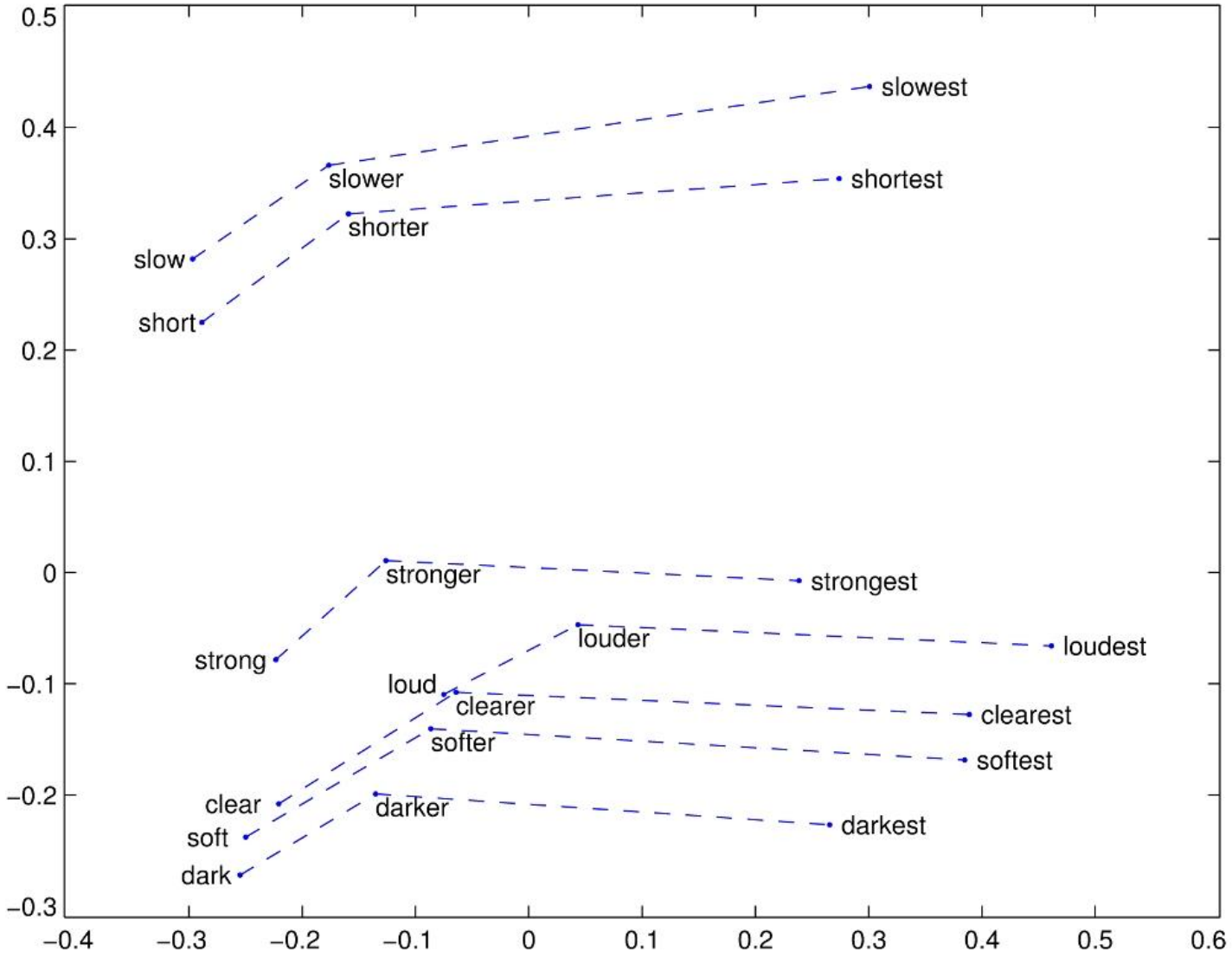


Image courtesy of Jurafsky & Martin

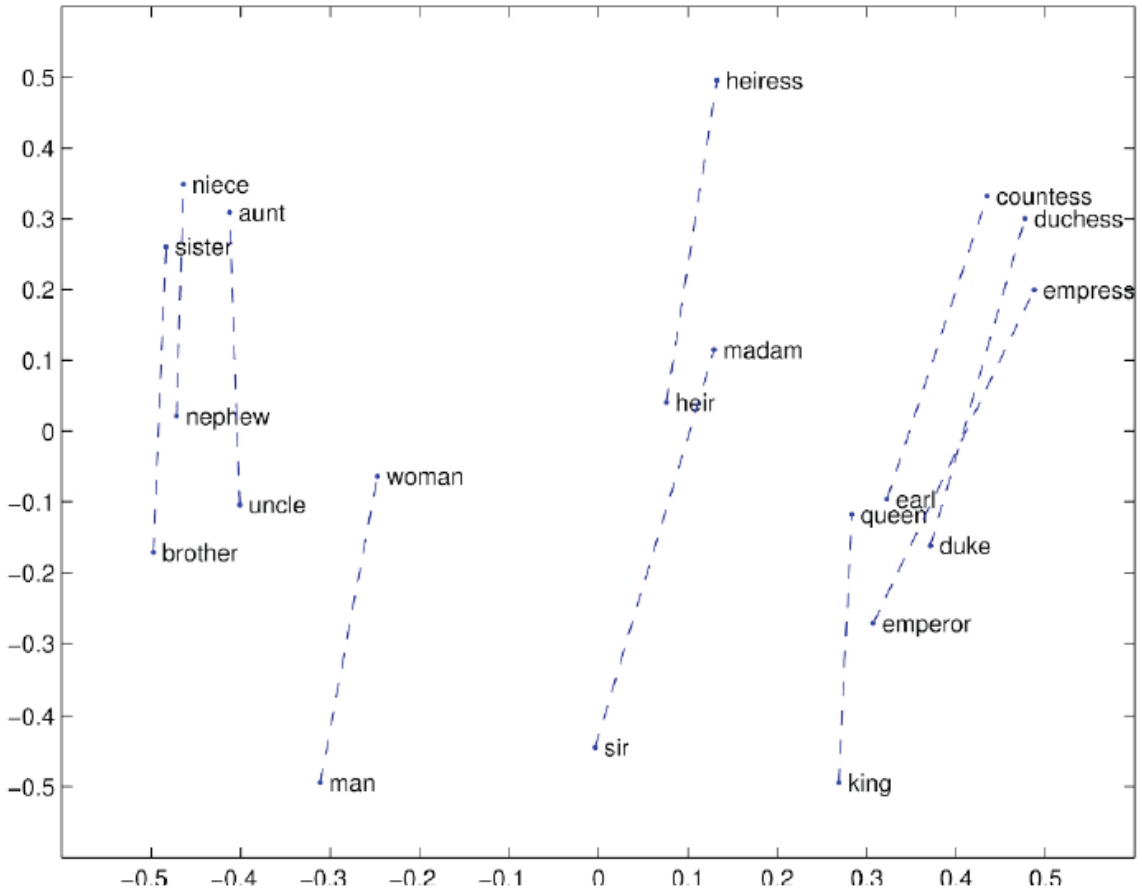
Embeddings Are Magic, Part 2

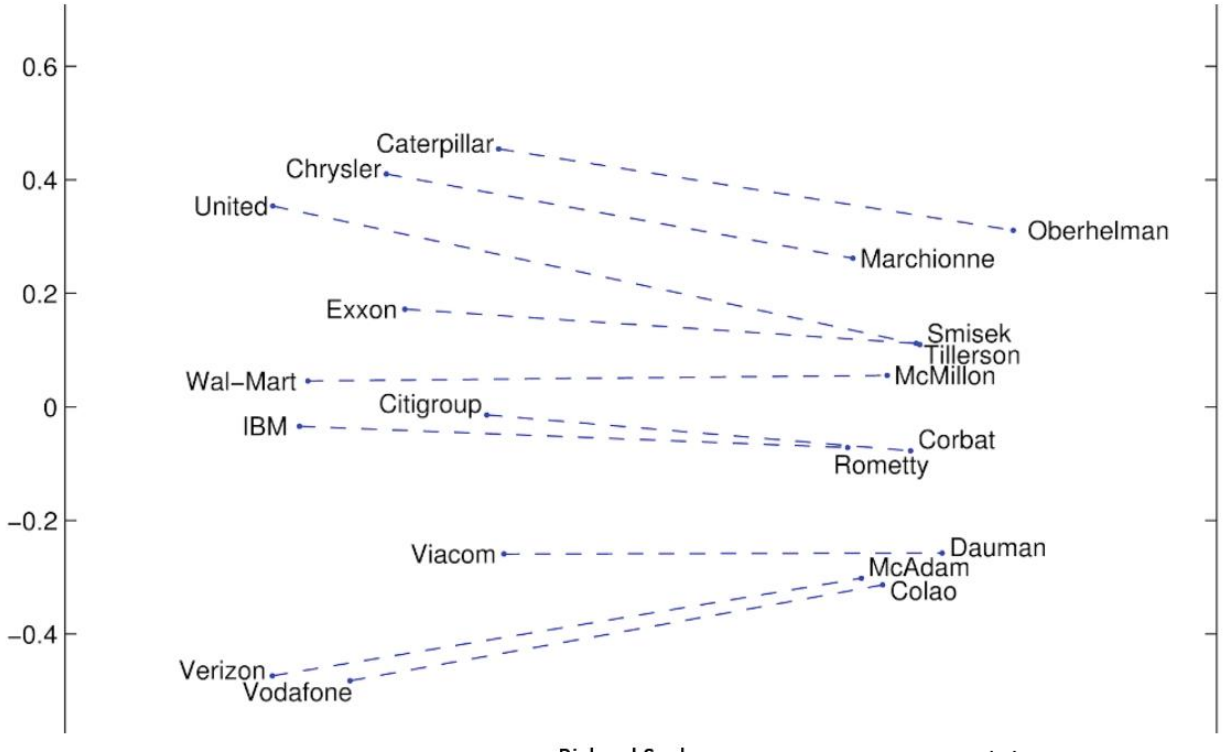


GloVe vectors for comparative and superlative adjectives

http://nlp.stanford.edu/projects/glove/images/comparative_superlative.jpg

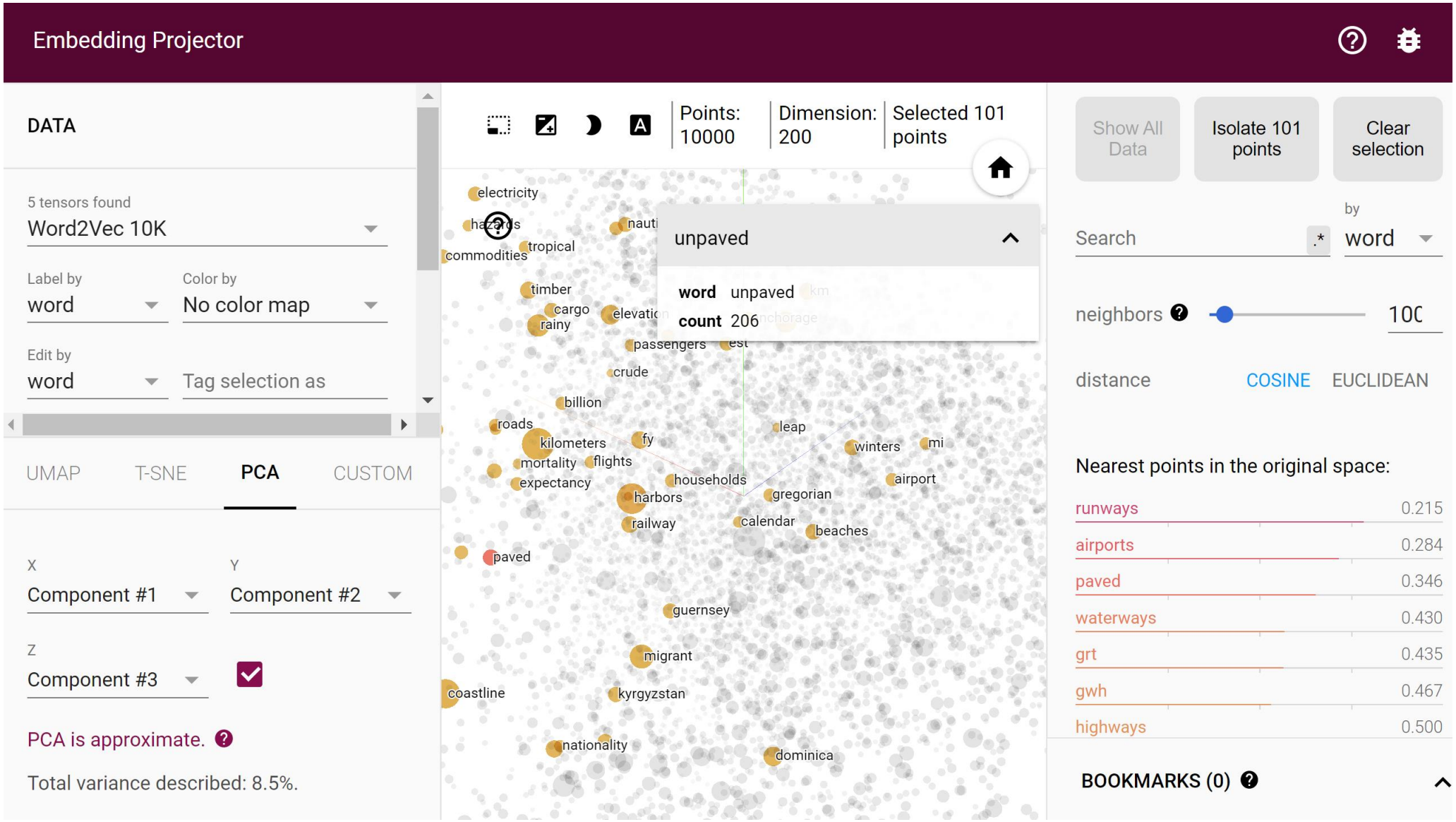
Embeddings Are Magic, Part 3





Examples from Richard Socher

Tensorflow Projector Demo



https://projector.tensorflow.org/

Deep Learning

721.

2. Word2Vec

word2vec

word2vec (Mikolov et al., 2013a)

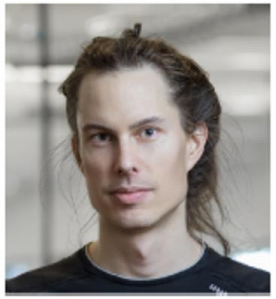
Efficient Estimation of Word Representations in Vector Space

Tomas Mikolov
Google Inc., Mountain View, CA
tmikolov@google.com

Kai Chen
Google Inc., Mountain View, CA
kaichen@google.com

Greg Corrado
Google Inc., Mountain View, CA
gcorrado@google.com

Jeffrey Dean
Google Inc., Mountain View, CA
jeff@google.com



word2vec (Mikolov et al., 2013b)

Distributed Representations of Words and Phrases and their Compositionality

Tomas Mikolov
Google Inc.
Mountain View
mikolov@google.com

Ilya Sutskever
Google Inc.
Mountain View
ilyasu@google.com

Kai Chen
Google Inc.
Mountain View
kai@google.com

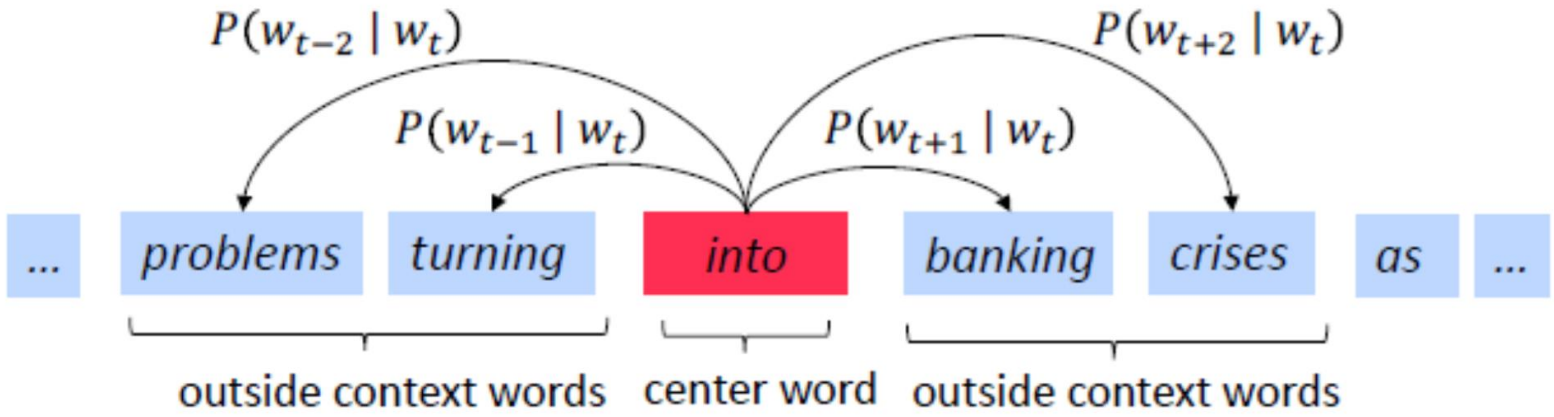
Greg Corrado
Google Inc.
Mountain View
gcorrado@google.com

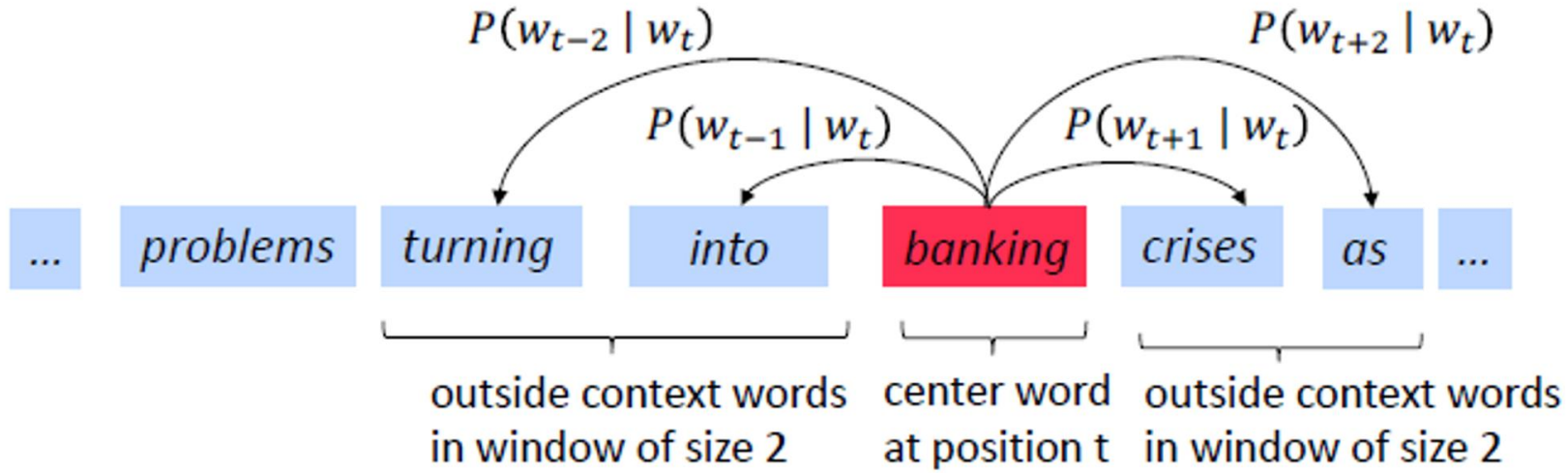
Jeffrey Dean
Google Inc.
Mountain View
jeff@google.com

word2vec

- Predicting adjacent words, instead of counting
- Words that have similar embeddings will also have similar meanings
- Is word *w* likely to appear near word “cat”?
- No need for hand-labeled data
- Idea based on Bengio et al. 2003 and Collobert et al. 2011 on neural language models

Word2vec (skipgram version)





[Chris Manning]

Objective function

For each position $t = 1, \ldots, T$, predict context words within a window of fixed size m, given center word w_j.

Likelihood = $L(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \le j \le m \\ j \neq 0}} P(w_{t+j} \mid w_t; \theta)$

θ is all variables to be optimized

sometimes called *cost* or *loss* function

The objective function $J(\theta)$ is the (average) negative log likelihood:

$$J(\theta) = -\frac{1}{T} \log L(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \neq 0}} \log P(w_{t+j} \mid w_t; \theta)$$

Minimizing objective function $\Leftrightarrow$ Maximizing predictive accuracy

[Chris Manning]

Minimizing the objective function

- We want to minimize the objective function:

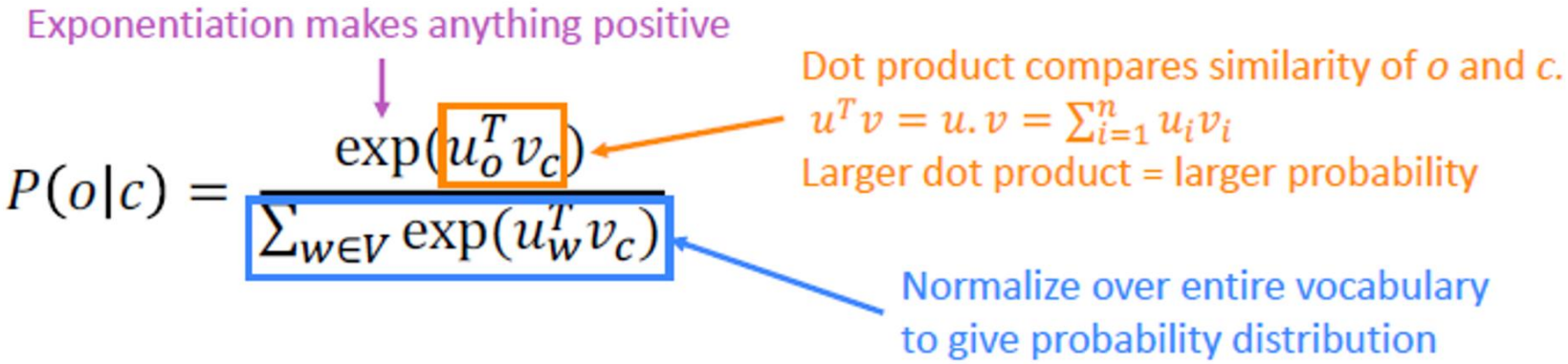
$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\le j\le m \\ j\neq 0}} \log P\left(w_{t+j} \mid w_t; \theta\right)$$

- Question: How to calculate $P\left(w_{t+j} \mid w_t; \theta\right)$?
- Answer: We will *use two* vectors per word *w*:
  - v_w when *w* is a center word
  - u_w when *w* is a context word
- Then for a center word *c* and a context word *o*:

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w\in V}\exp(u_w^T v_c)}$$

[Chris Manning]

Converting the similarities to probabilities



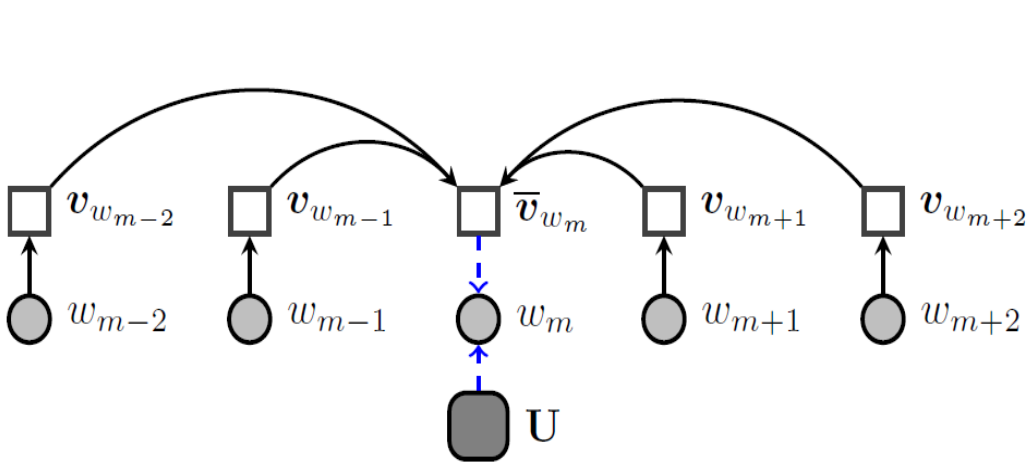
- This is an example of the **softmax function** $\mathbb{R}^n \to \mathbb{R}^n$

$$\text{softmax}(x_i) = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)} = p_i$$

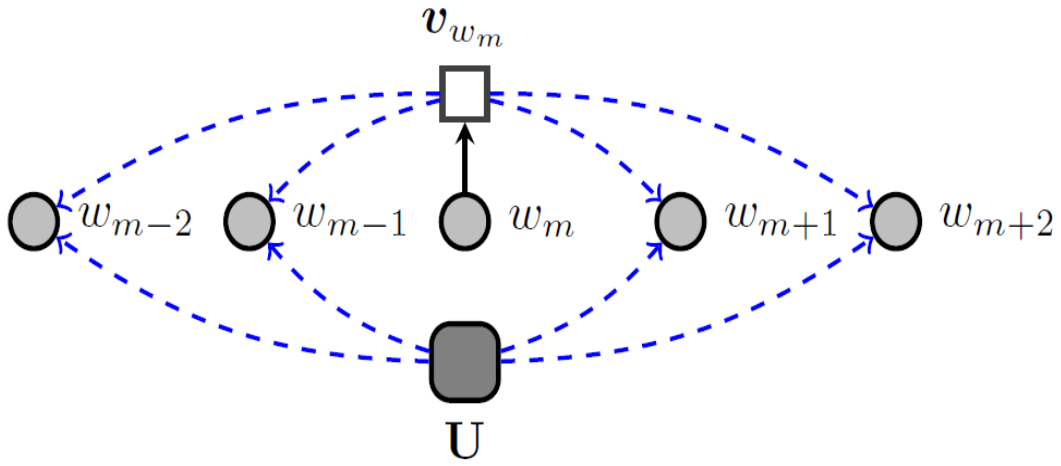
- The softmax function maps arbitrary values x_i to a probability distribution p_i
  - "max" because amplifies probability of largest x_i
  - "soft" because still assigns some probability to smaller x_i
  - Frequently used in Deep Learning

[Chris Manning]

CBOW vs. Skipgram



(a) Continuous bag-of-words (CBOW)

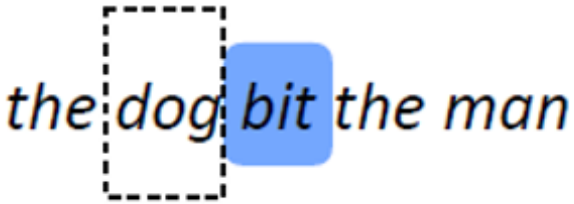


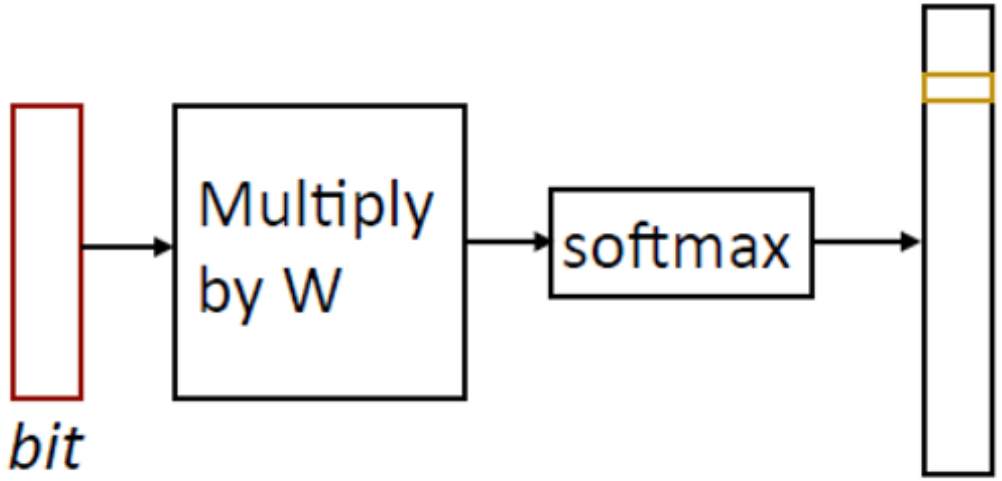
(b) Skipgram

Figure 14.3: The CBOW and skipgram variants of WORD2VEC. The parameter $\mathbf{U}$ is the matrix of word embeddings, and each $\boldsymbol{v}_m$ is the context embedding for word w_m.

Skipgrams

- Predict one word of context from word





gold = *dog*

$$P(w'|w) = \text{softmax}(We(w))$$

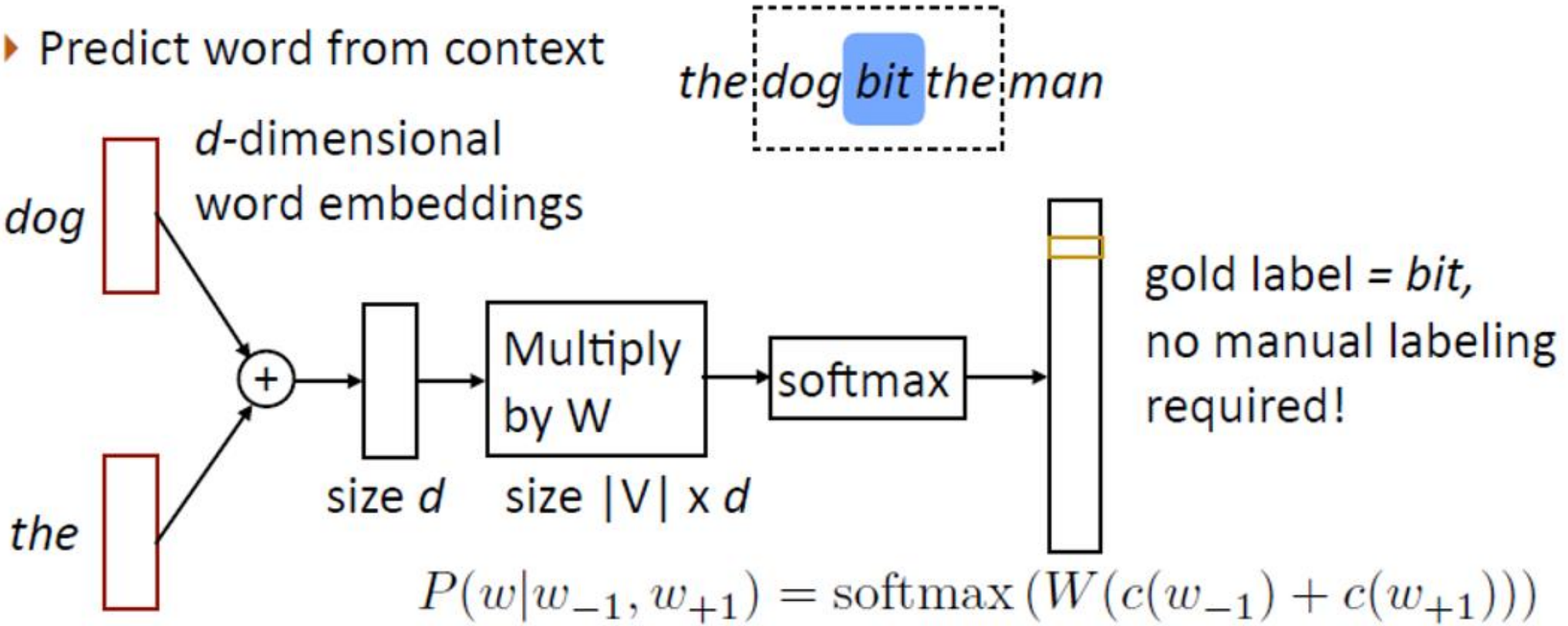
- Another training example: *bit* -> *the*
- Parameters: *d* x |V| vectors, |V| x *d* output parameters (W) (also usable as vectors!)

Mikolov et al. (2013)

[Greg Durrett]

Continuous bag of words (CBOW)

- Predict word from context



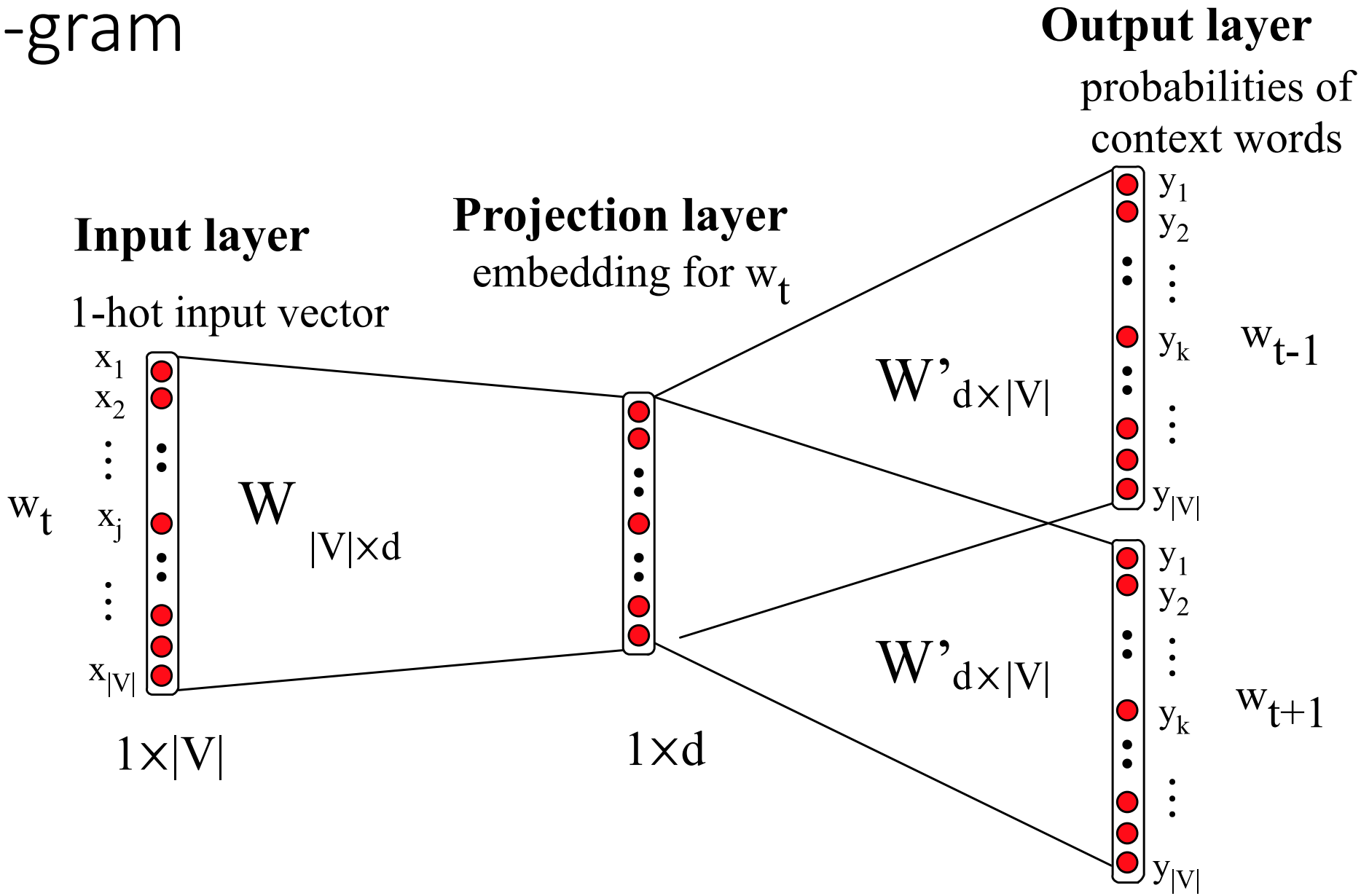
$$P(w|w_{-1}, w_{+1}) = \text{softmax}\left(W(c(w_{-1}) + c(w_{+1}))\right)$$

- Parameters: d x $|V|$ (one d-length vector per voc word),
  $|V|$ x d output parameters (W)

Mikolov et al. (2013)

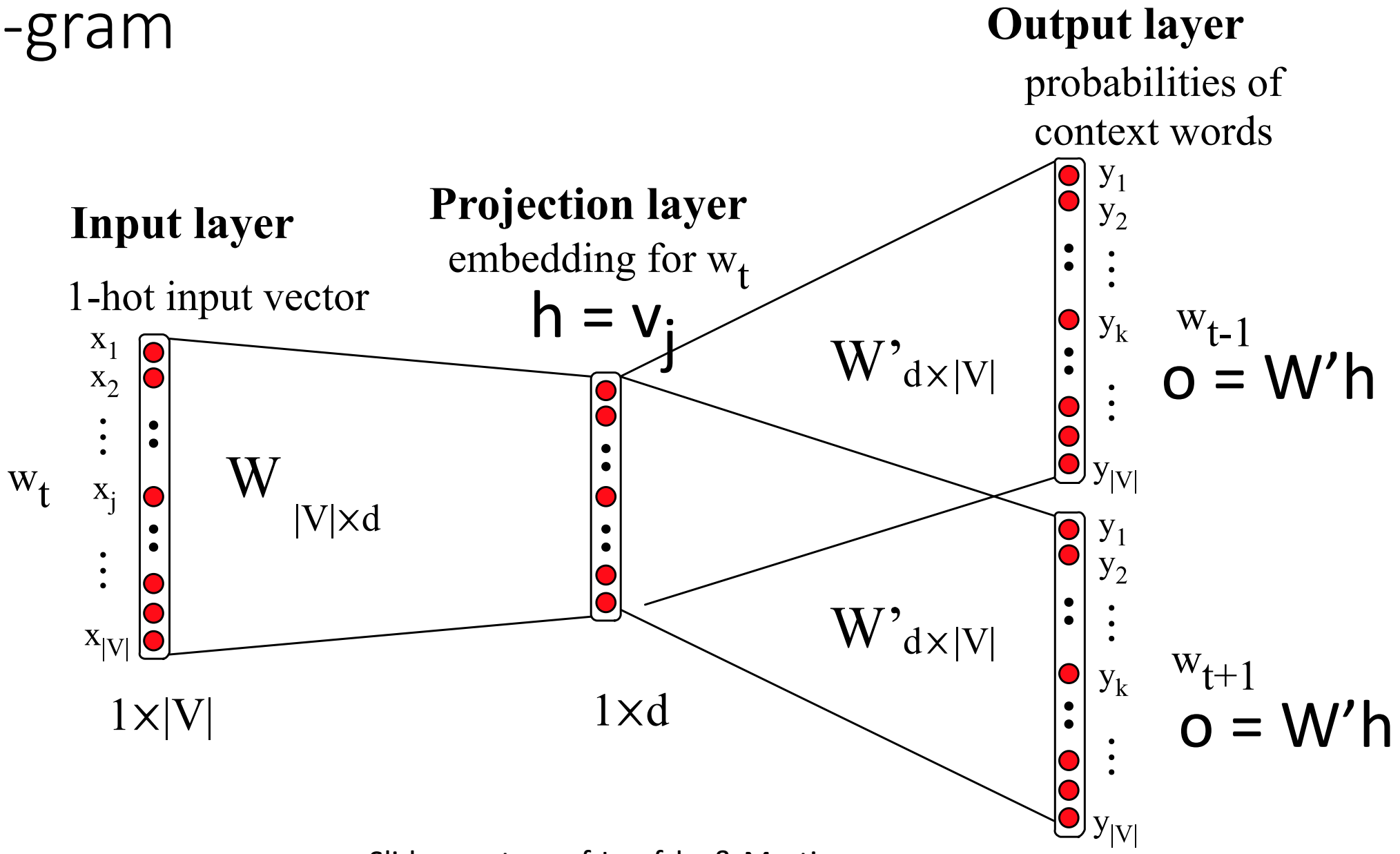
[Greg Durrett]

Skip-gram



Slide courtesy of Jurafsky & Martin

Skip-gram



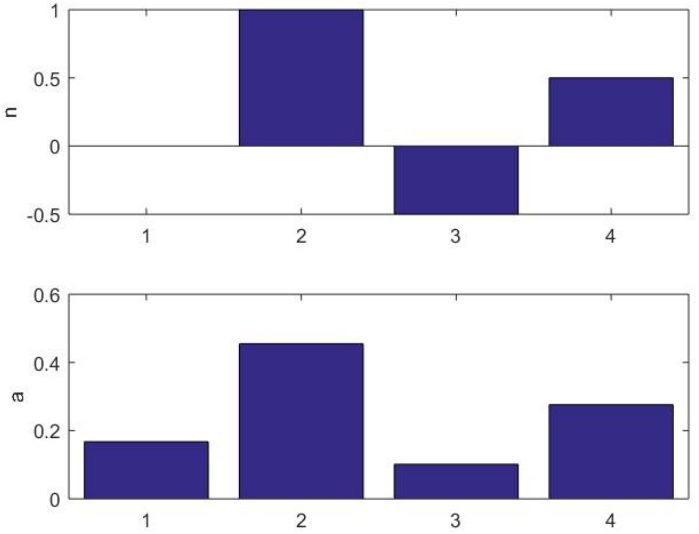
Slide courtesy of Jurafsky & Martin

Similarity Computation

- Using the dot product of the two vectors
- To convert a similarity to a probability, use softmax

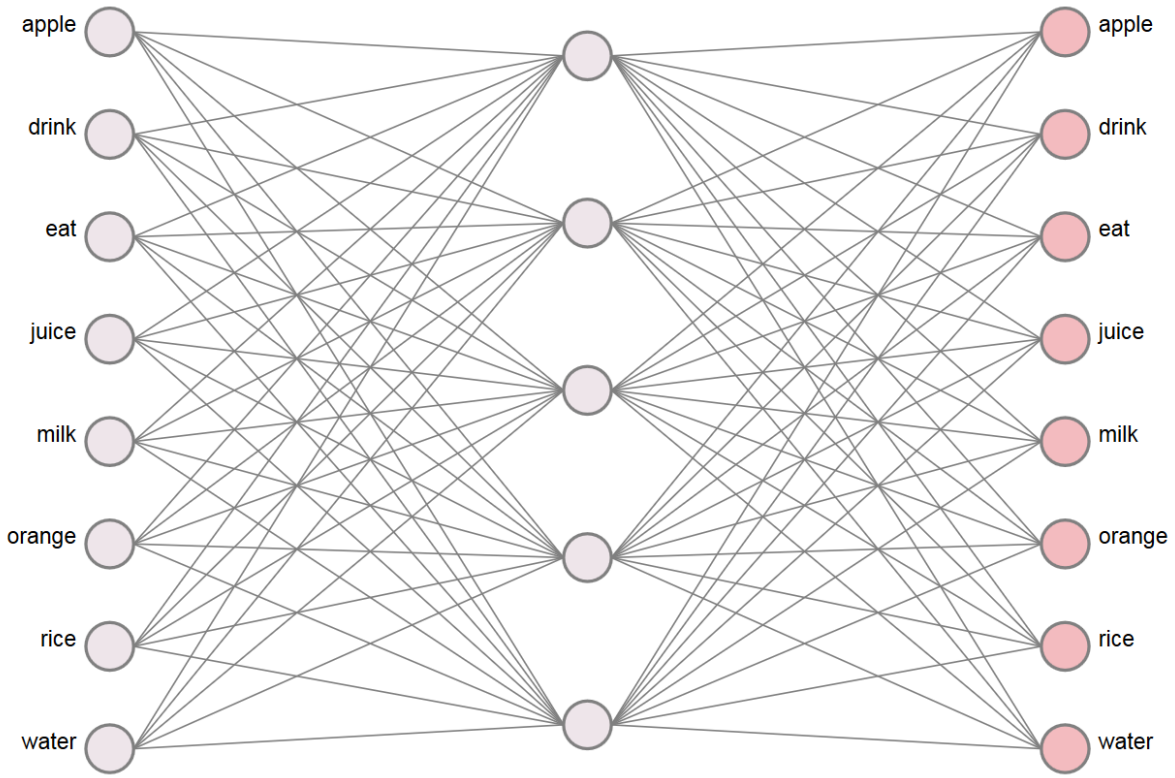
$$p(w_k|w_j) = \frac{\exp(c_k v_j)}{\sum_i \exp(c_i v_j)}$$

- Too many words in the denominator
- In practice, use negative sampling
  - the denominator is only computed for a few words
  - (in a few slides)



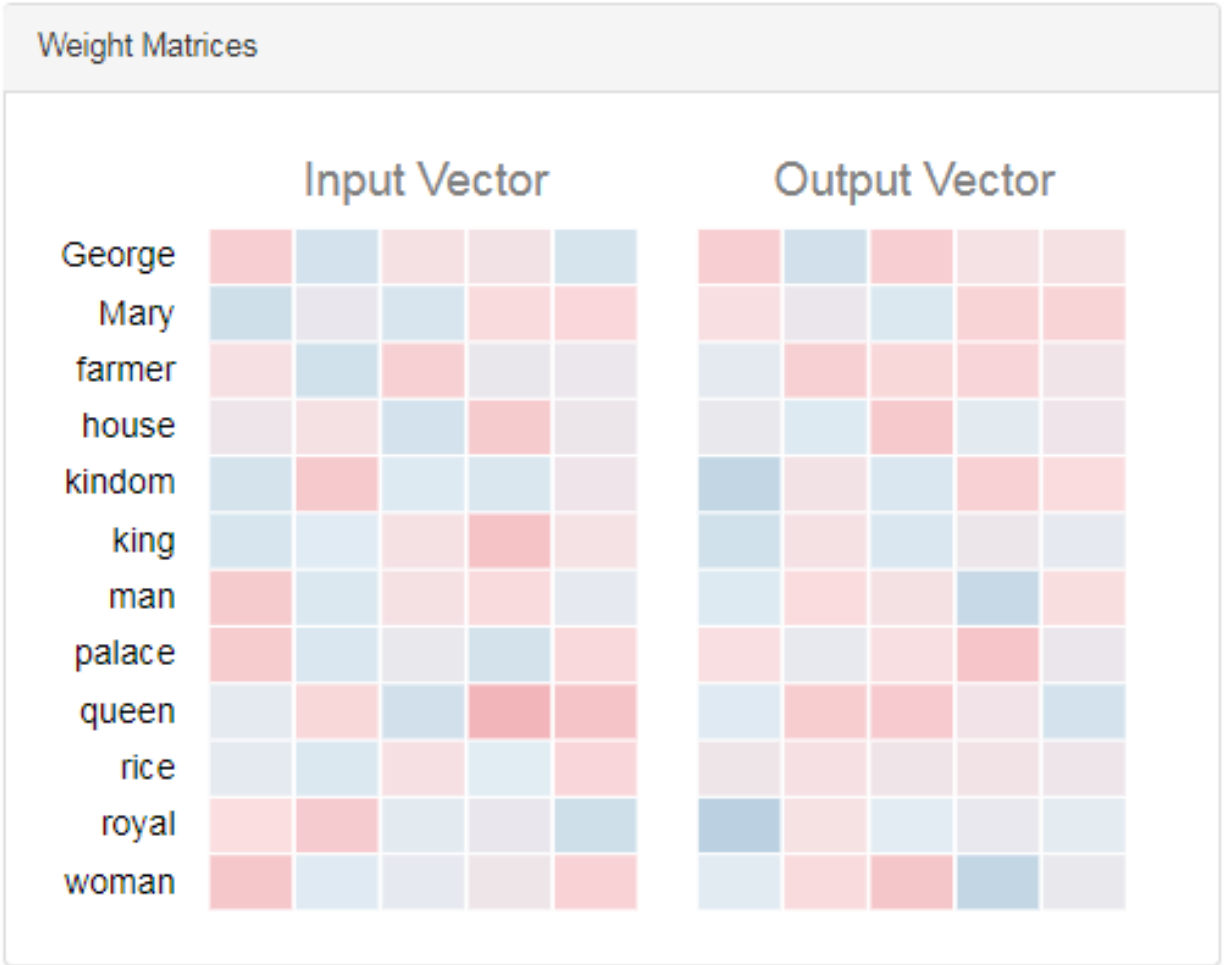
WEVI (Xin Rong)

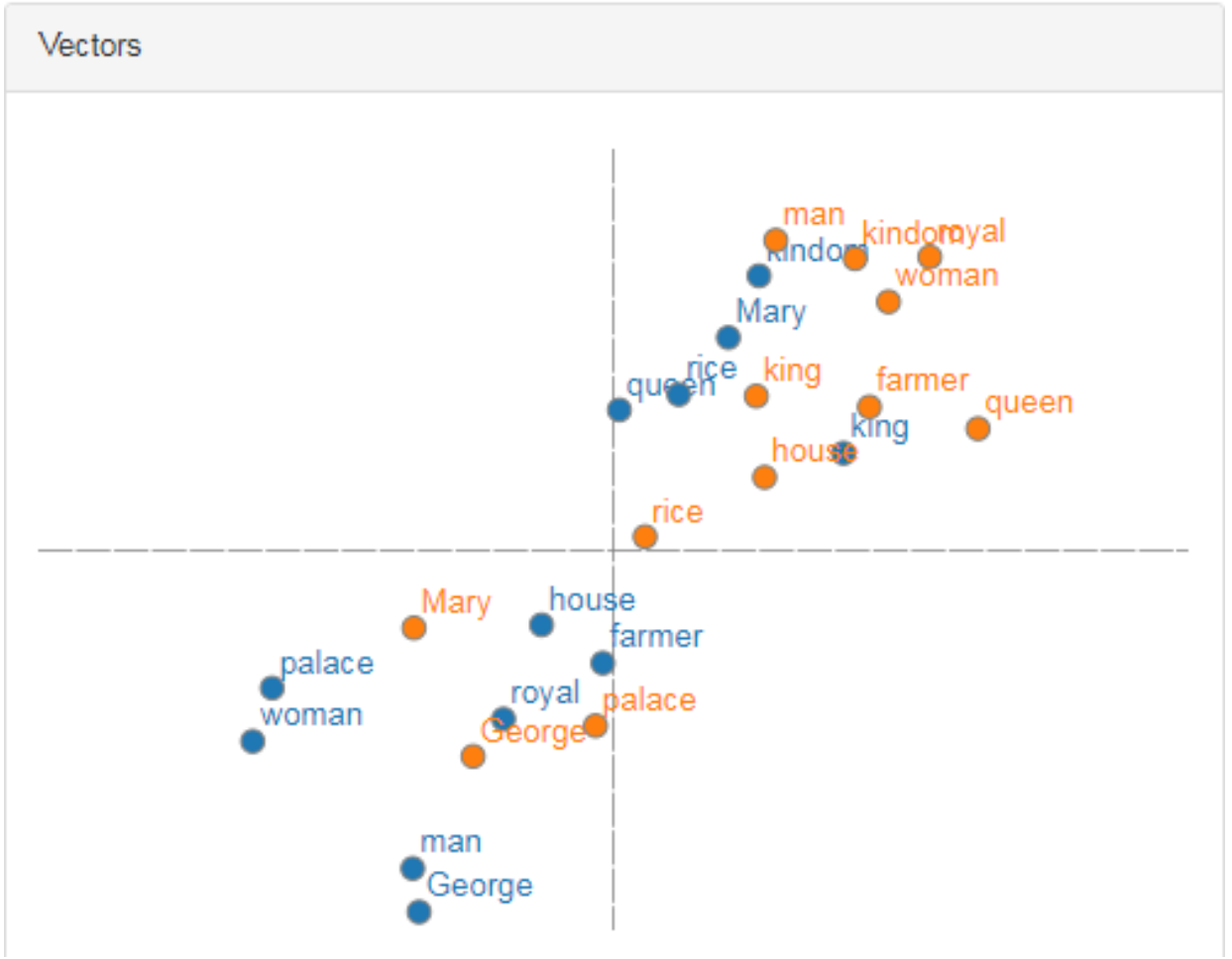
https://ronxin.github.io/wevi/



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

WEVI (Xin Rong)





Notes

- Notes on context:
  - Small – syntax
  - Large – semantics

- Usefulness
  - Very – tagging, parsing, text classification
  - Less – machine translation
  - Not – language modeling

[Graham Neubig]

Links

- https://code.google.com/p/word2vec/
  - includes the models and pre-trained embeddings
  - Pre-trained is good, because training takes a lot of data
- Gensim: Python library that works with word2vec
  - https://radimrehurek.com/gensim/

Deep Learning

3. Training Embeddings

CBOW

$$\overline{\boldsymbol{v}}_m = \frac{1}{2h} \sum_{n=1}^{h} \boldsymbol{v}_{w_{m+n}} + \boldsymbol{v}_{w_{m-n}}.$$

The CBOW model optimizes an approximation to the corpus log-likelihood,

$$\log \mathrm{p}(\boldsymbol{w}) \approx \sum_{m=1}^{M} \log \mathrm{p}(w_m \mid w_{m-h}, w_{m-h+1}, \ldots, w_{m+h-1}, w_{m+h}) \quad [14.15]$$

$$= \sum_{m=1}^{M} \log \frac{\exp\left(\boldsymbol{u}_{w_m} \cdot \overline{\boldsymbol{v}}_m\right)}{\sum_{j=1}^{V} \exp\left(\boldsymbol{u}_j \cdot \overline{\boldsymbol{v}}_m\right)} \quad [14.16]$$

$$= \sum_{m=1}^{M} \boldsymbol{u}_{w_m} \cdot \overline{\boldsymbol{v}}_m - \log \sum_{j=1}^{V} \exp\left(\boldsymbol{u}_j \cdot \overline{\boldsymbol{v}}_m\right). \quad [14.17]$$

Every word has two embeddings: v_w (when w is the center word) and u_w (when w is a context word).

Skipgram

In the CBOW model, words are predicted from their context. In the **skipgram** model, the context is predicted from the word, yielding the objective:

$$\log \mathrm{p}(\boldsymbol{w}) \approx \sum_{m=1}^{M} \sum_{n=1}^{h_m} \log \mathrm{p}(w_{m-n} \mid w_m) + \log \mathrm{p}(w_{m+n} \mid w_m) \quad [14.18]$$

$$= \sum_{m=1}^{M} \sum_{n=1}^{h_m} \log \frac{\exp(\boldsymbol{u}_{w_{m-n}} \cdot \boldsymbol{v}_{w_m})}{\sum_{j=1}^{V} \exp(\boldsymbol{u}_j \cdot \boldsymbol{v}_{w_m})} + \log \frac{\exp(\boldsymbol{u}_{w_{m+n}} \cdot \boldsymbol{v}_{w_m})}{\sum_{j=1}^{V} \exp(\boldsymbol{u}_j \cdot \boldsymbol{v}_{w_m})} \quad [14.19]$$

$$= \sum_{m=1}^{M} \sum_{n=1}^{h_m} \boldsymbol{u}_{w_{m-n}} \cdot \boldsymbol{v}_{w_m} + \boldsymbol{u}_{w_{m+n}} \cdot \boldsymbol{v}_{w_m} - 2 \log \sum_{j=1}^{V} \exp\left(\boldsymbol{u}_j \cdot \boldsymbol{v}_{w_m}\right). \quad [14.20]$$

In the skipgram approximation, each word is generated multiple times; each time it is conditioned only on a single word. This makes it possible to avoid averaging the word vectors, as in the CBOW model. The local neighborhood size h_m is randomly sampled from a uniform categorical distribution over the range $\{1, 2, \ldots, h_{\text{max}}\}$; Mikolov et al. (2013) set $h_{\text{max}} = 10$. Because the neighborhood grows outward with h, this approach has the effect of weighting near neighbors more than distant ones. Skipgram performs better on most evaluations than CBOW (see § 14.6 for details of how to evaluate word representations), but CBOW is faster to train (Mikolov et al., 2013).

Summary

- Start with V random 300-dimensional vectors as initial embeddings
- Use logistic regression, the second most basic classifier used in machine learning after Naïve Bayes
  - 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.

[Diyi Yang]

Deep Learning

4. Negative Sampling

Negative Sampling

- Take (word, context) pairs and classify them as "real" or not. Create random negative examples by sampling from unigram distribution

  (*bit, the*) => +1

  (*bit, cat*) => -1

  (*bit, a*) => -1

  (*bit, fish*) => -1

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

words in similar contexts select for similar *c* vectors

- *d* x |V| vectors, *d* x |V| context vectors (same # of params as before)
- Objective = $\log P(y=1|w,c) - \frac{1}{k}\sum_{i=1}^{n} \log P(y=0|w_i,c)$

sampled

Mikolov et al. (2013)

[Greg Durrett]

Binary Log Loss with Negative Sampling
(Mikolov et al., 2013)

- like what we saw earlier with similarity modeling, now with skip-gram scoring function:

$$\min_{\boldsymbol{\theta}} \sum_{1 \le t \le |\mathcal{T}|} \sum_{-c \le j \le c, j \ne 0} -\log \sigma(\boldsymbol{\phi}_{x_{t+j}}^\top \boldsymbol{\psi}_{x_t}) - \sum_{x \in \text{NEG}} \log\left(1 - \sigma(\boldsymbol{\phi}_x^\top \boldsymbol{\psi}_{x_t})\right)$$

KG2

Binary Log Loss with Negative Sampling
(Mikolov et al., 2013)

- like what we saw earlier with similarity modeling, now with skip-gram scoring function:

$$\min_{\boldsymbol{\theta}} \sum_{1 \le t \le |\mathcal{T}|} \sum_{-c \le j \le c, j \ne 0} -\log \sigma(\boldsymbol{\phi}_{x_{t+j}}^\top \boldsymbol{\psi}_{x_t}) - \sum_{x \in \text{NEG}} \log\left(1 - \sigma(\boldsymbol{\phi}_x^\top \boldsymbol{\psi}_{x_t})\right)$$

Note: this is not actually what the word2vec code does. The code samples a window size *d* (up to the max window size *c*), then samples a *j* value between –*d* and *d* (such that *j* does not equal 0)

Deep Learning

5. Hierarchical Softmax

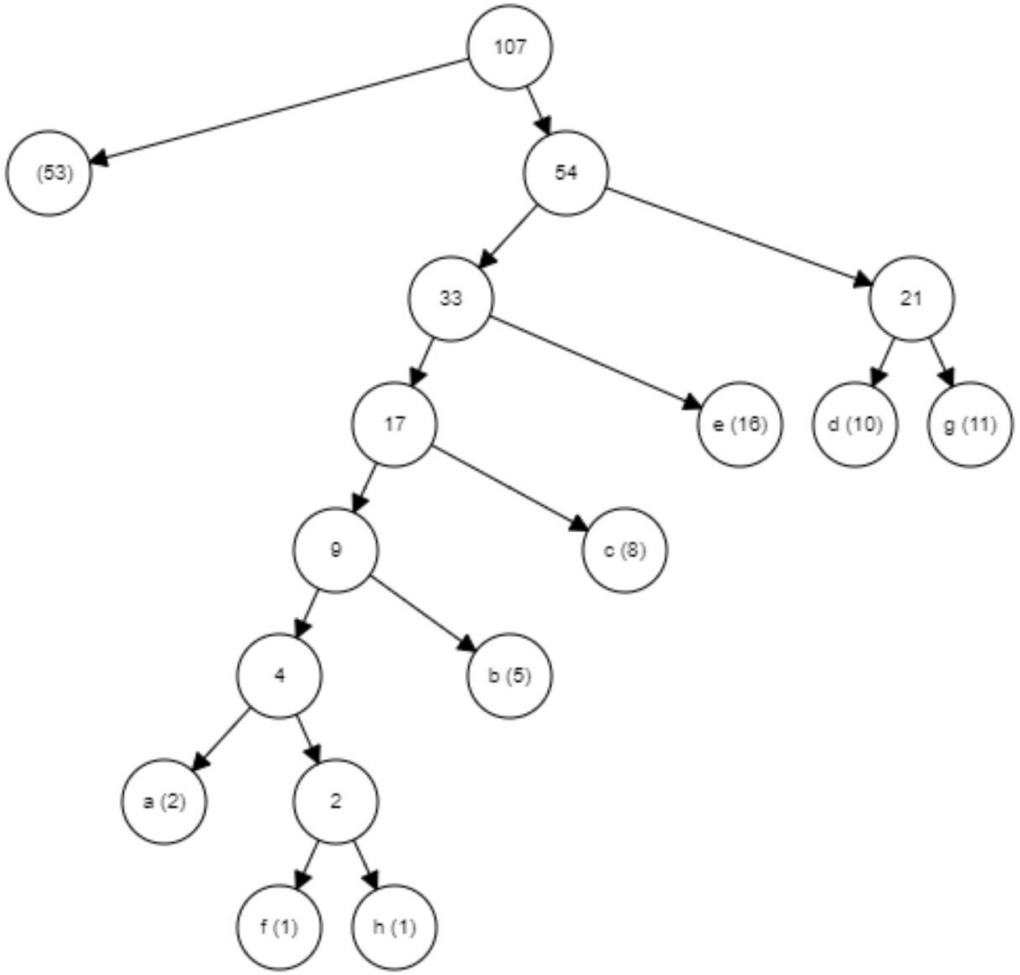
Hierarchical softmax

- Use a binary tree
- Each word is a leaf node
- Each word has a unique path associated with it
- Only need to learn output vectors for the internal nodes

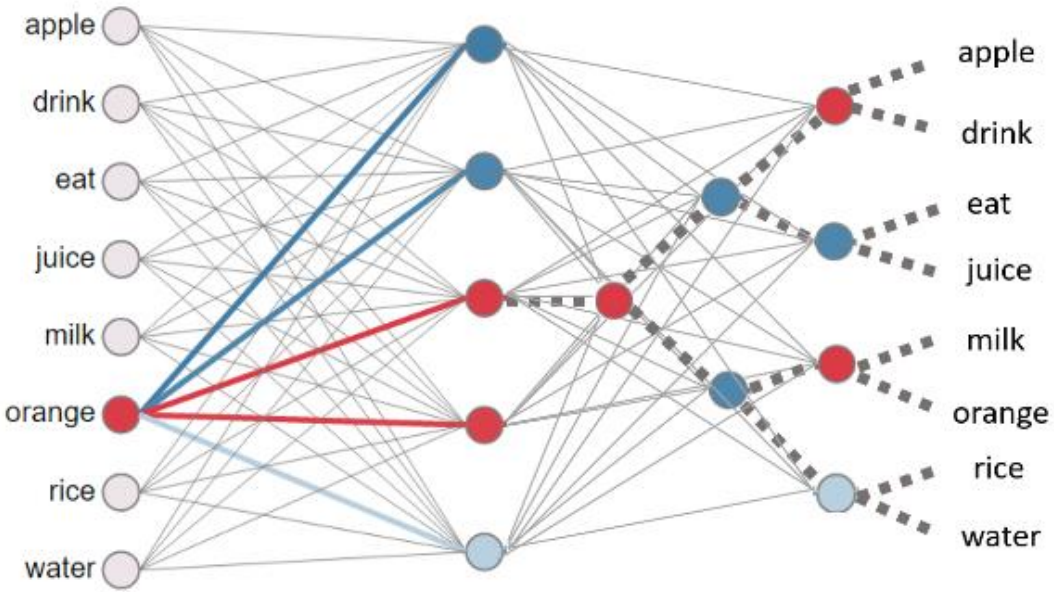
Huffman tree

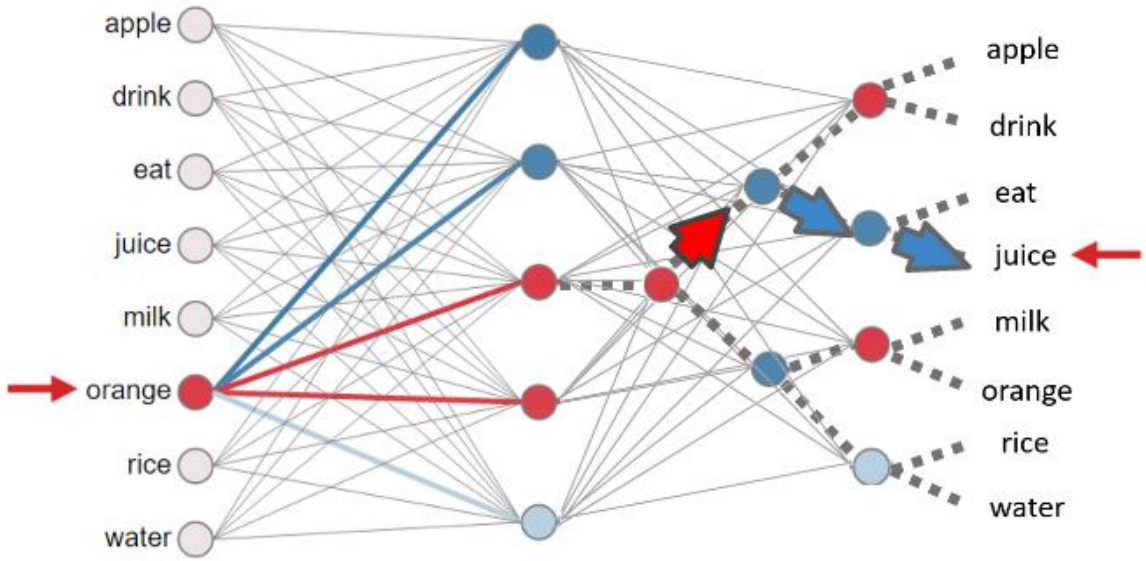
- Depth is O(log |V|)

a	2
b	5
c	8
d	10
e	16
f	1
g	11
h	1



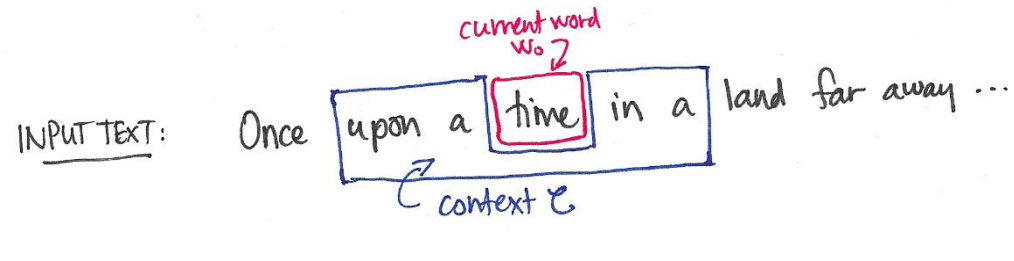
https://people.ok.ubc.ca/ylucet/DS/Huffman.html

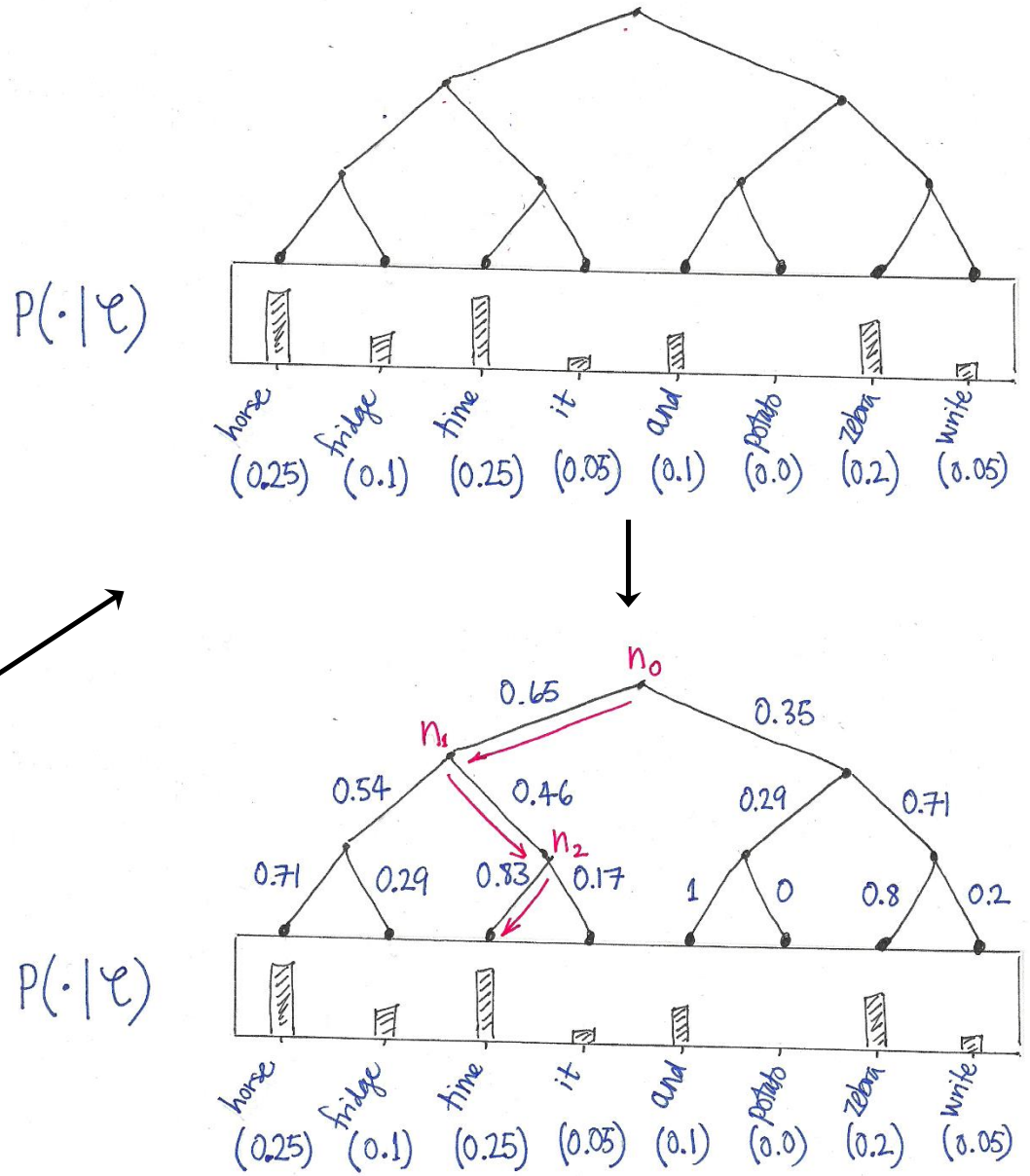




[Thanaki 2017]

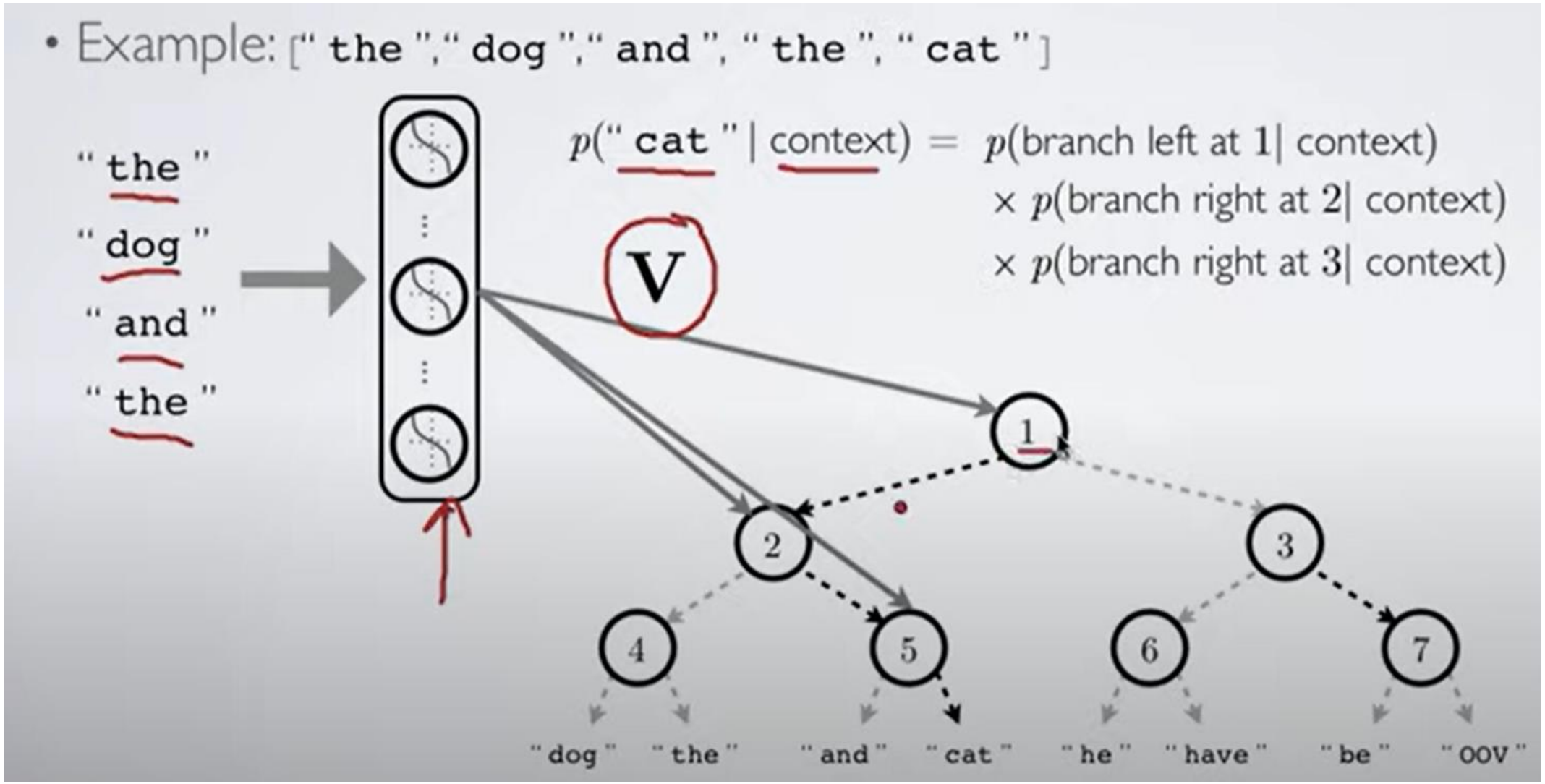
Hierarchical softmax





http://building-babylon.net/2017/08/01/hierarchical-softmax/

Example



Video by Hugo Larochelle https://www.youtube.com/watch?v=B95LTf2rVWM

NLP

Deep Learning

6. Evaluating Embeddings

Evaluating Embeddings

- Nearest Neighbors
- Information Retrieval
- Analogies
  - (a:b)::(c:?)

$$d = \arg\max_i \frac{(x_b - x_a + x_c)^T x_i}{||x_b - x_a + x_c||}$$

```
Minsk Belarus                    Rabat Morocco
Denver Colorado                  Sacramento California
professional professionally      most mostly
Moldova Moldovan                 Greece Greek
```

https://github.com/nicholas-leonard/word2vec/blob/master/questions-words.txt

Similarity Data Sets

Dataset	Word pairs	Reference
RG	65	Rubenstein and Goodenough (1965)
MC	30	Miller and Charles (1991)
WS-353	353	Finkelstein et al. (2002)
YP-130	130	Yang and Powers (2006)
MTurk-287	287	Radinsky et al. (2011)
MTurk-771	771	Halawi et al. (2012)
MEN	3000	Bruni et al. (2012)
RW	2034	Luong et al. (2013)
Verb	144	Baker et al. (2014)
SimLex	999	Hill et al. (2014)

[Table from Faruqui et al. 2016]

Semantic-syntactic word relationship (SSWR)

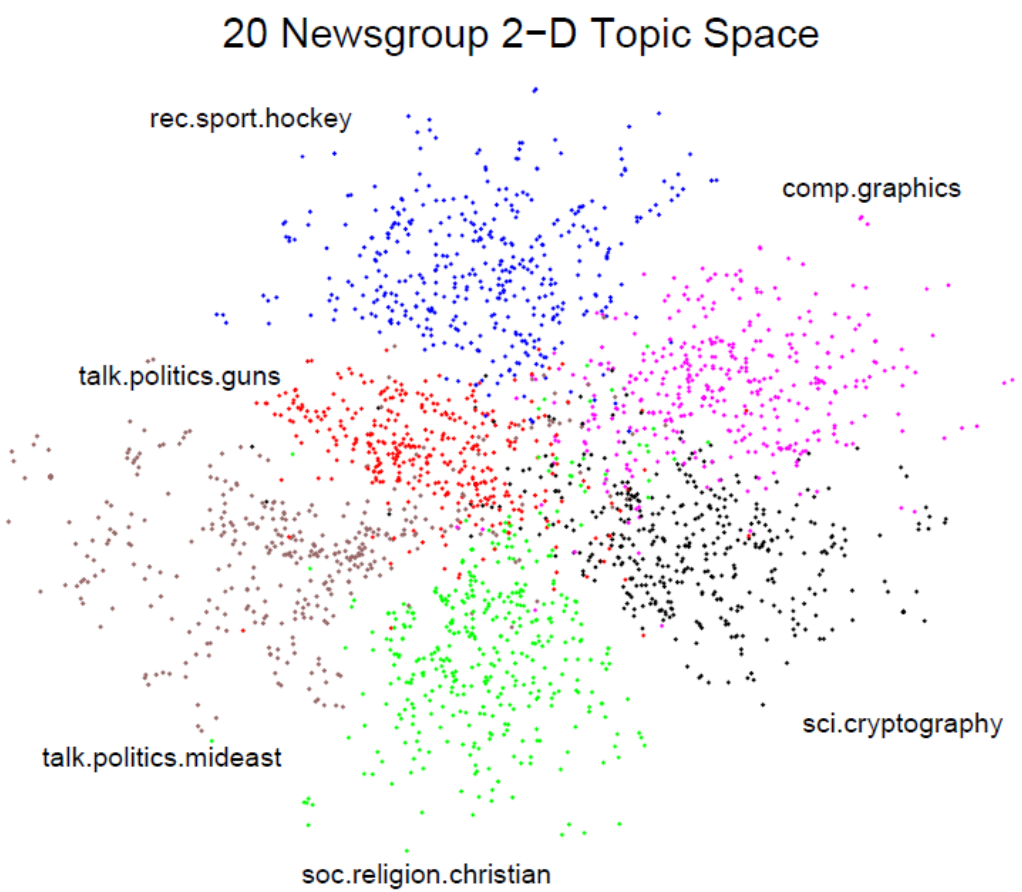
Table 1: *Examples of five types of semantic and nine types of syntactic questions in the Semantic-Syntactic Word Relationship test set.*

Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	brother	sister	grandson	granddaughter
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks

https://github.com/nicholas-leonard/word2vec/blob/master/questions-words.txt

[Mikolov et al. 2013]

Semantic Hashing



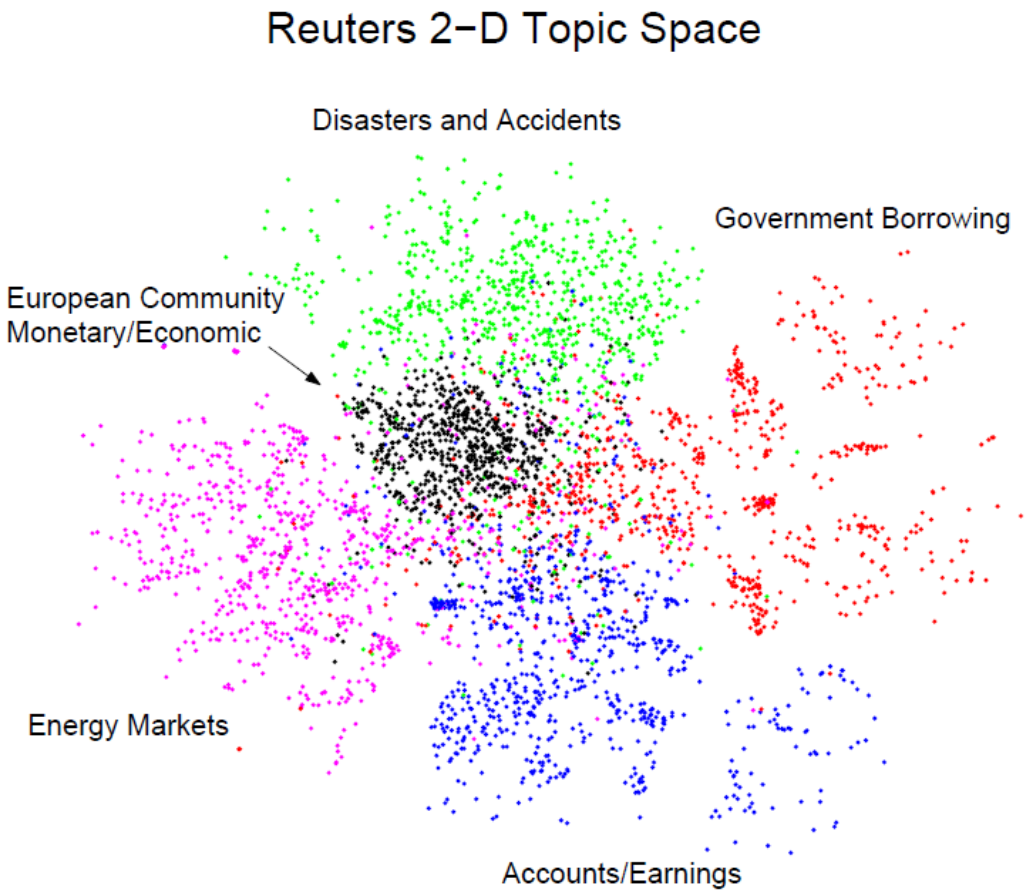


Figure 5: A 2-dimensional embedding of the 128-bit codes using stochastic neighbor embedding for the 20 Newsgroups data (left panel) and the Reuters RCV2 corpus (right panel). See in color for better visualization.

[Salakhutdinov and Hinton 2007]

Intrinsic evaluation

• magician	wizard	9.02
• midday	noon	9.29
• furnace	stove	8.79
• food	fruit	7.52
• food	rooster	4.42
• coast	hill	4.38
• money	withdrawal	6.88
• money	laundering	5.65
• money	operation	3.31
• tiger	jaguar	8
• tiger	feline	8
• tiger	carnivore	7.08
• tiger	mammal	6.85

http://www.gabrilovich.com/resources/data/wordsim353/wordsim353.html

simlex-999

• happy	mad	A	0.95
• large	huge	A	9.47
• hard	tough	A	8.05
• new	fresh	A	6.83
• sharp	dull	A	0.6
• quick	rapid	A	9.7
• dumb	foolish	A	6.67
• wonderful	terrific	A	8.63
• strange	odd	A	9.02
• dreary	dull	A	8.25
• sad	terrible	A	5.4
• dirty	cheap	A	1.6
• elastic	flexible	A	7.78

• beg	ask	V	6
• arrange	organize	V	8.27
• reduce	shrink	V	8.02
• acknowledge	speak	V	4.67
• give	borrow	V	2.22
• kill	defend	V	2.63
• disappear	shrink	V	5.8
• deliver	carry	V	3.88
• breathe	choke	V	1.37
• acknowledge	notify	V	5.3
• become	seem	V	2.63
• pretend	seem	V	4.68
• accomplish	become	V	4

Felix Hill, Roi Reichart and Anna Korhonen. *Computational Linguistics.* 2015 https://fh295.github.io/simlex.html

Intrinsic evaluation of different embeddings

Model	Size	WS353	MC	RG	SCWS	RW
SVD	6B	35.3	35.1	42.5	38.3	25.6
SVD-S	6B	56.5	71.5	71.0	53.6	34.7
SVD-L	6B	65.7	72.7	75.1	56.5	37.0
CBOW†	6B	57.2	65.6	68.2	57.0	32.5
SG†	6B	62.8	65.2	69.7	58.1	37.2
GloVe	6B	65.8	72.7	77.8	53.9	38.1
SVD-L	42B	74.0	76.4	74.1	58.3	39.9
GloVe	42B	**75.9**	**83.6**	**82.9**	**59.6**	**47.8**
CBOW*	100B	68.4	79.6	75.4	59.4	45.5

NLP

Deep Learning

7. Other types of embeddings

Retrofitting Embeddings

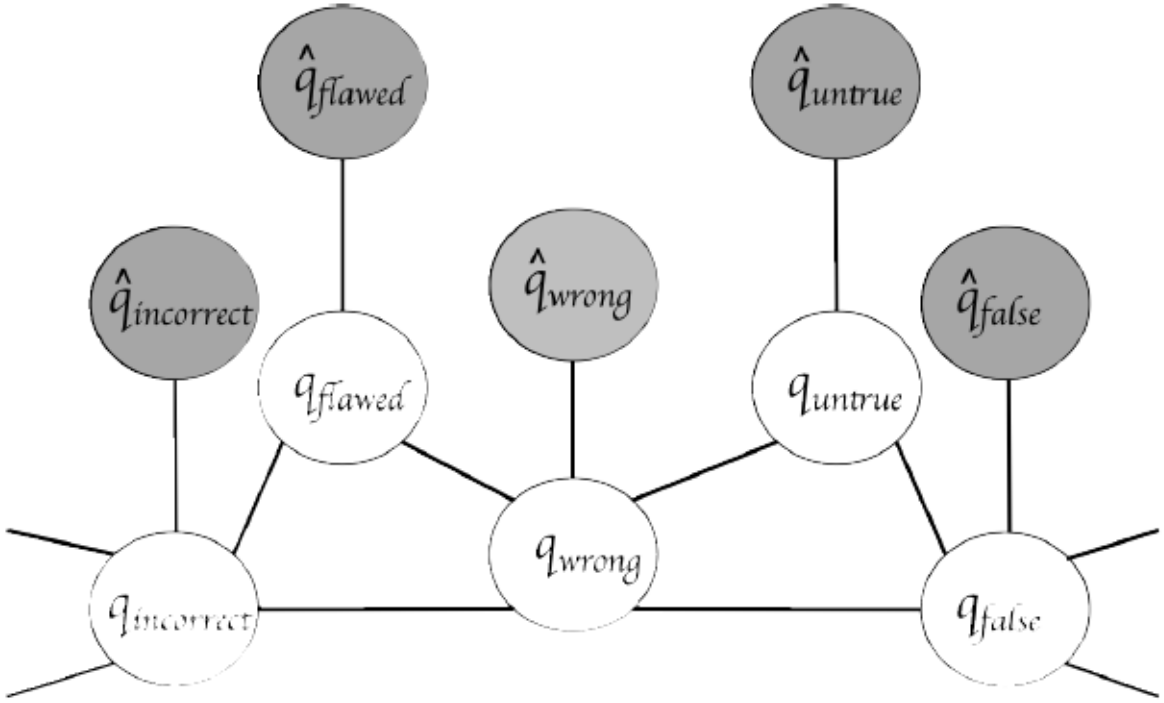


Figure 1: Word graph with edges between related words showing the observed (grey) and the inferred (white) word vector representations.

$$\Psi(Q) = \sum_{i=1}^{n} \left[\alpha_i \|q_i - \hat{q}_i\|^2 + \sum_{(i,j) \in E} \beta_{ij} \|q_i - q_j\|^2 \right]$$

[Faruqui et al. 2015]

Word Embeddings = Linear Combination of Sense Embeddings

$$v_{tie} \approx \alpha_1 v_{tie1} + \alpha_2 v_{tie2} + \alpha_3 v_{tie3} + \cdots \quad (1)$$

tie					spring				
trousers	season	scoreline	wires	operatic	beginning	dampers	flower	creek	humid
blouse	teams	goalless	cables	soprano	until	brakes	flowers	brook	winters
waistcoat	winning	equaliser	wiring	mezzo	months	suspension	flowering	river	summers
skirt	league	clinching	electrical	contralto	earlier	absorbers	fragrant	fork	ppen
sleeved	finished	scoreless	wire	baritone	year	wheels	lilies	piney	warm
pants	championship	replay	cable	coloratura	last	damper	flowered	elk	temperatures

Table 6: Five discourse atoms linked to the words *tie* and *spring*. Each atom is represented by its nearest 6 words. The algorithm often makes a mistake in the last atom (or two), as happened here.

[Arora et al. 2018]

Other types of embeddings

- Syntactic
- Mutilingual
- Multisense
- Contextual

Issues with embeddings

- Can overestimate similarity
  - Bar (1) vs. Bar (2)
- Can underestimate similariy
  - Cat vs. Cats
- Don’t incorporate world knowledge explicitly
- Can be biased
  - (later slides)
- Difficult to interpret

Multi-sense embeddings

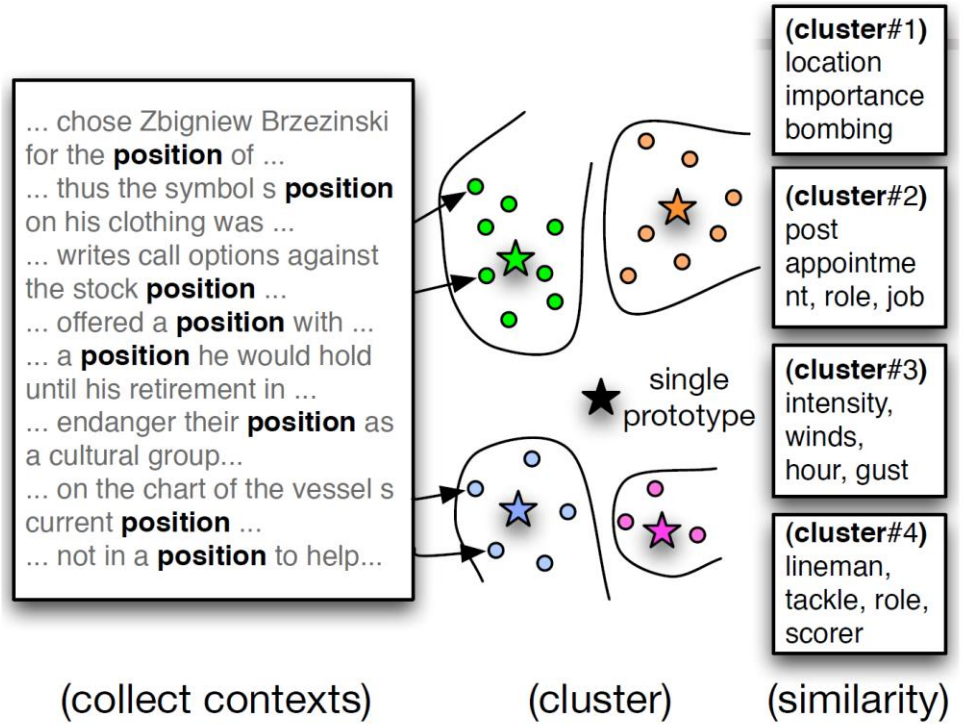


Figure 1: Overview of the multi-prototype approach to near-synonym discovery for a single target word independent of context. Occurrences are clustered and cluster centroids are used as prototype vectors. Note the "hurricane" sense of *position* (cluster 3) is not typically considered appropriate in WSD.

[Reisinger and Mooney 2010]

Multilingual embeddings

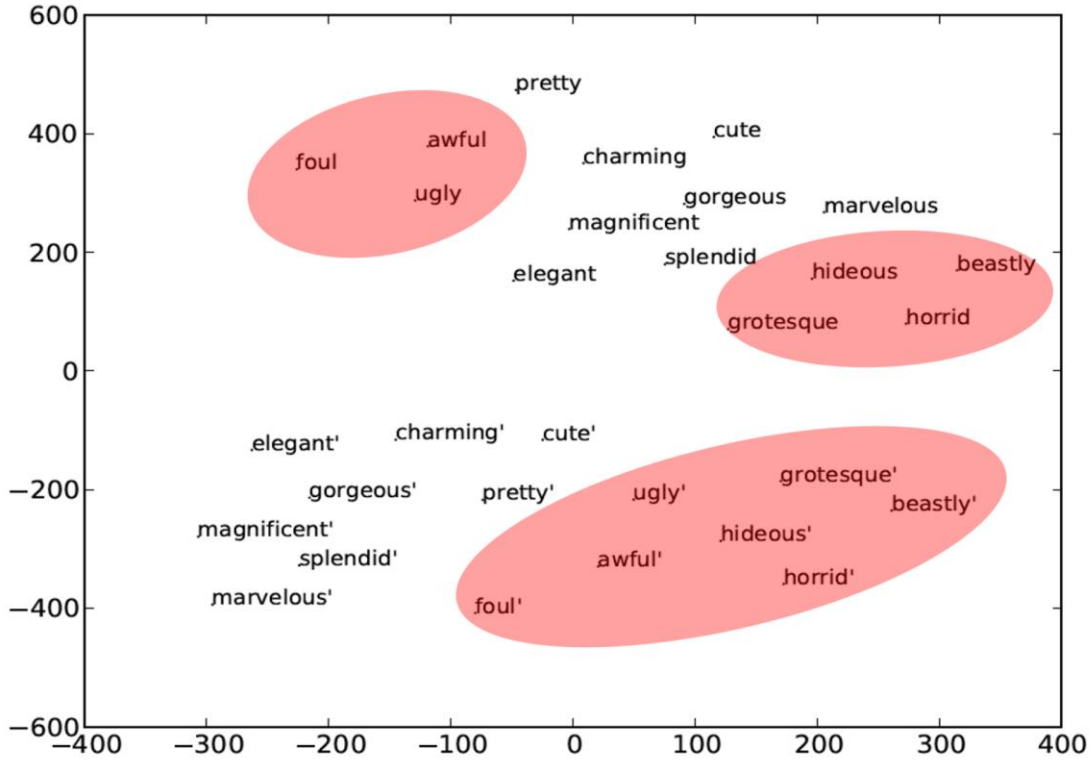
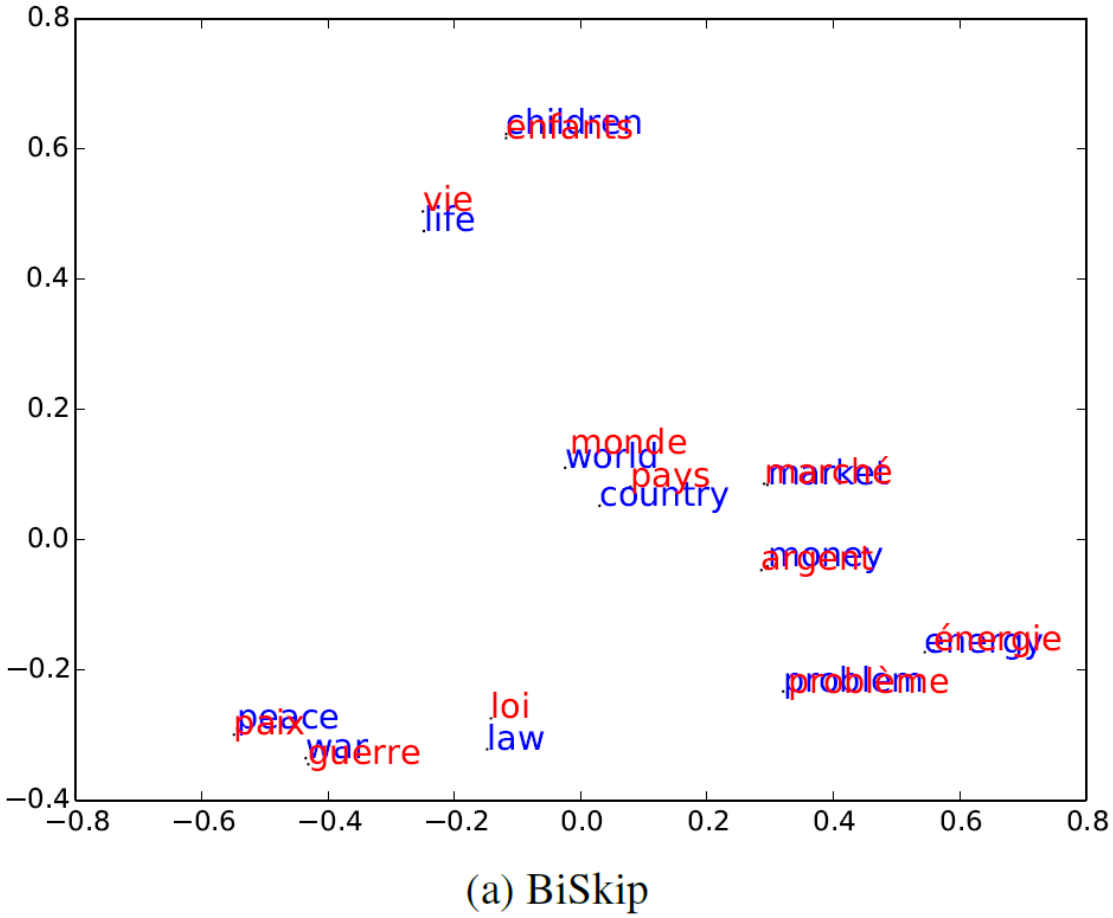


Figure 2: Monolingual (top) and multilingual (bottom; marked with apostrophe) word projections of the antonyms (shown in red) and synonyms of "beautiful".

[Faruqui and Dyer 2014]



(a) BiSkip

[Upadhyay et al. 2016]

Deep Learning

8. Uses of embeddings

Diachronic embeddings

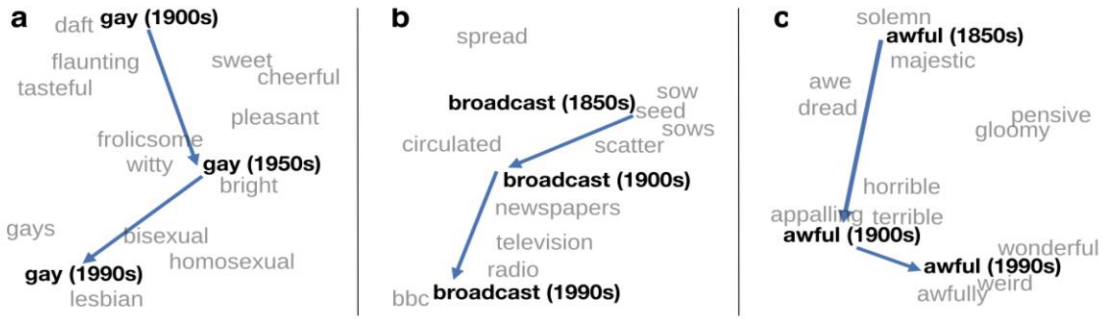


Figure 1: Two-dimensional visualization of semantic change in English using SGNS vectors.[2] **a**, The word *gay* shifted from meaning "cheerful" or "frolicsome" to referring to homosexuality. **b**, In the early 20th century *broadcast* referred to "casting out seeds"; with the rise of television and radio its meaning shifted to "transmitting signals". **c**, *Awful* underwent a process of pejoration, as it shifted from meaning "full of awe" to meaning "terrible or appalling" (Simpson et al., 1989).

Word	Moving towards	Moving away	Shift start	Source
gay	homosexual, lesbian	happy, showy	ca 1950	(Kulkarni et al., 2014)
fatal	illness, lethal	fate, inevitable	<1800	(Jatowt and Duh, 2014)
awful	disgusting, mess	impressive, majestic	<1800	(Simpson et al., 1989)
nice	pleasant, lovely	refined, dainty	ca 1890	(Wijaya and Yeniterzi, 2011)
broadcast	transmit, radio	scatter, seed	ca 1920	(Jeffers and Lehiste, 1979)
monitor	display, screen	—	ca 1930	(Simpson et al., 1989)
record	tape, album	—	ca 1920	(Kulkarni et al., 2014)
guy	fellow, man	—	ca 1850	(Wijaya and Yeniterzi, 2011)
call	phone, message	—	ca 1890	(Simpson et al., 1989)

Table 2: Set of attested historical shifts used to evaluate the methods. The examples are taken from previous works on semantic change and from the Oxford English Dictionary (OED), e.g. using 'obsolete' tags. The shift start points were estimated using attestation dates in the OED. The first six examples are words that shifted dramatically in meaning while the remaining four are words that acquired new meanings (while potentially also keeping their old ones).

Word	Language	Nearest-neighbors in 1900s	Nearest-neighbors in 1990s
wanting	English	lacking, deficient, lacked, lack, needed	wanted, something, wishing, anything, anybody
asile	French	refuge, asiles, hospice, vieillards, infirmerie	demandeurs, refuge, hospice, visas, admission
widerstand	German	scheiterte, volt, stromstärke, leisten, brechen	opposition, verfolgung, nationalsozialistische, nationalsozialismus, kollaboration

Table 5: Example words that changed dramatically in meaning in three languages, discovered using SGNS embeddings. The examples were selected from the top-10 most-changed lists between 1900s and 1990s as in Table 4. In English, *wanting* underwent subjectification and shifted from meaning "lacking" to referring to subjective "desire", as in "the education system is wanting" (1900s) vs. "I've been wanting to tell you" (1990s). In French *asile* ("asylum") shifted from primarily referring to "hospitals, or infirmaries" to also referring to "asylum seekers, or refugees". Finally, in German *Widerstand* ("resistance") gained a formal meaning as referring to the local German resistance to Nazism during World War II.

https://nlp.stanford.edu/projects/histwords/

[Hamilton et al. 2016]

Domain-Specific Sentiment Lexicons for Computational Social Science

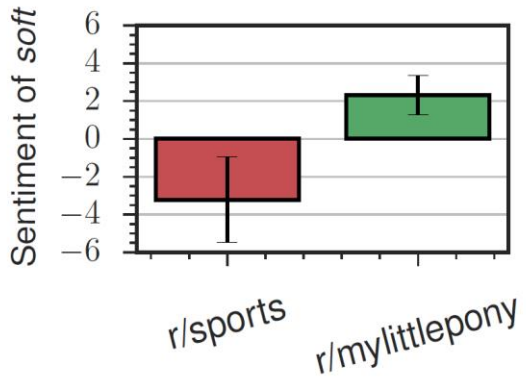
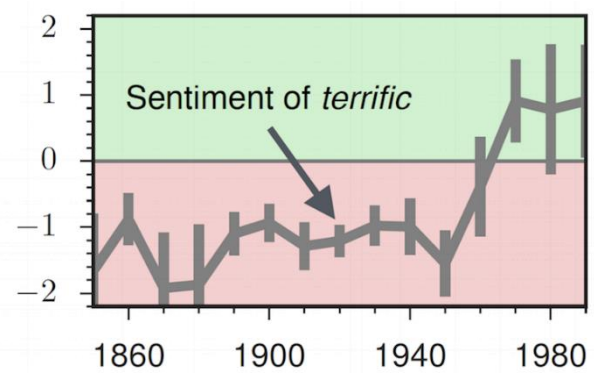
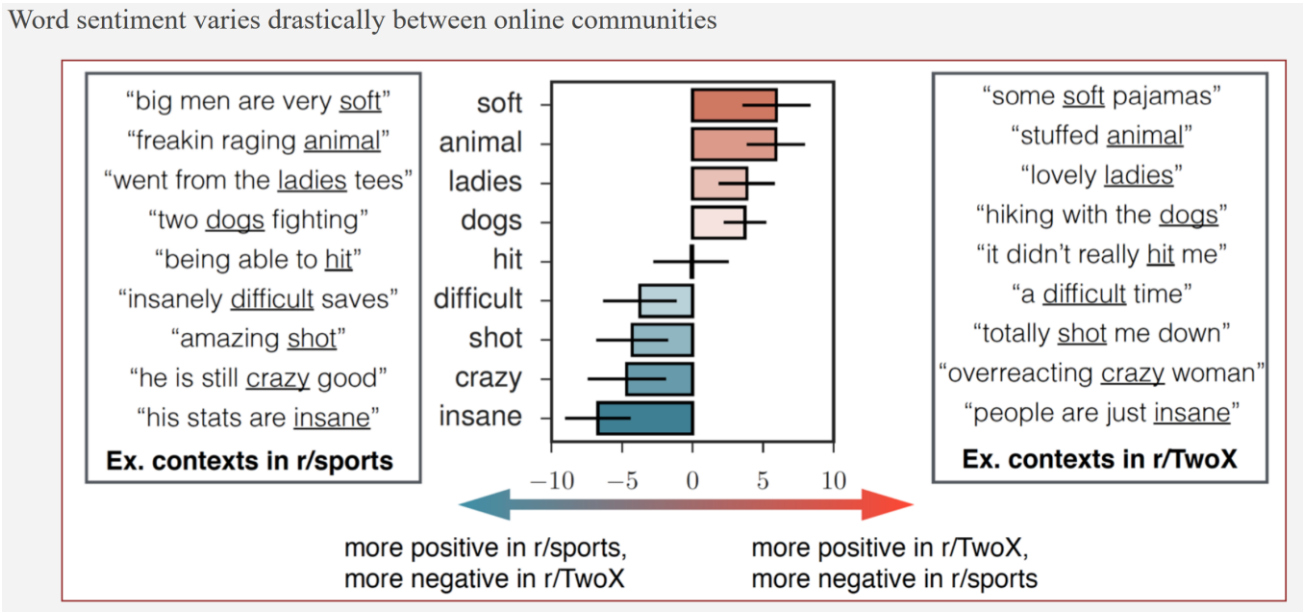
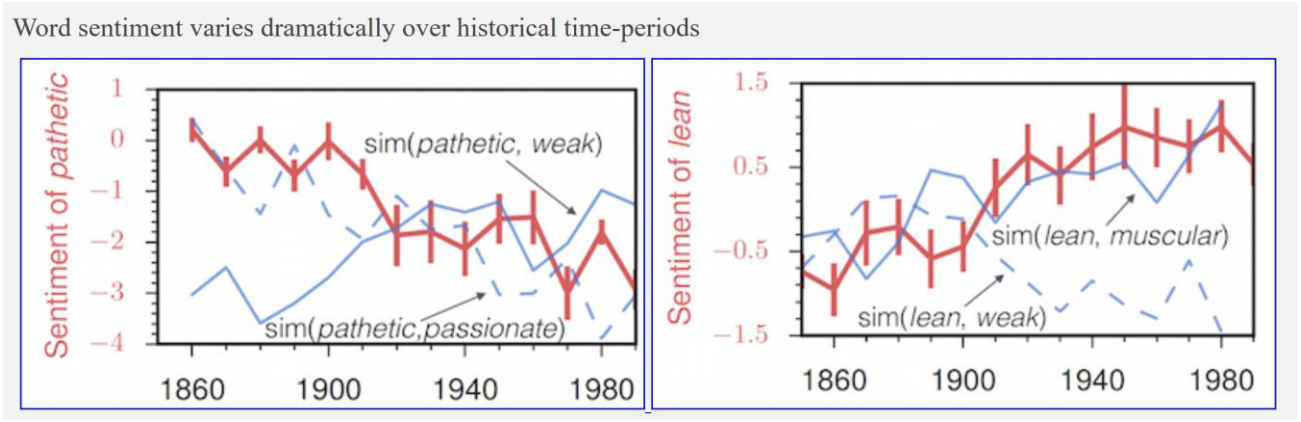


Figure 1: The sentiment of *soft* in different online communities. Sentiment values computed using SENTPROP (Section 3) on comments from Reddit communities illustrate how sentiment depends on social context. Bootstrap-sampled standard deviations provide a measure of confidence with the scores.





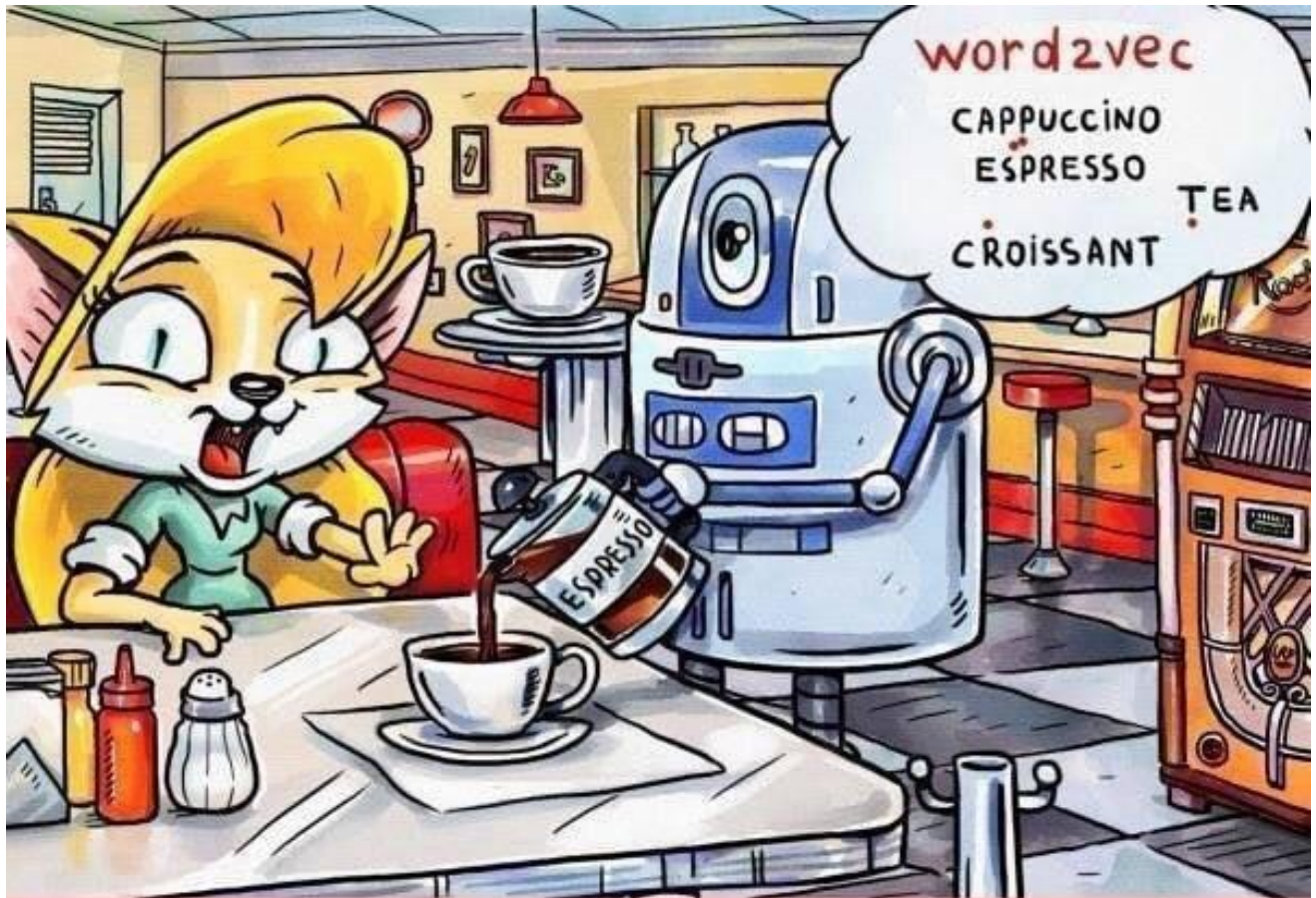


https://nlp.stanford.edu/projects/socialsent/

[Hamilton et a;l. 2016]

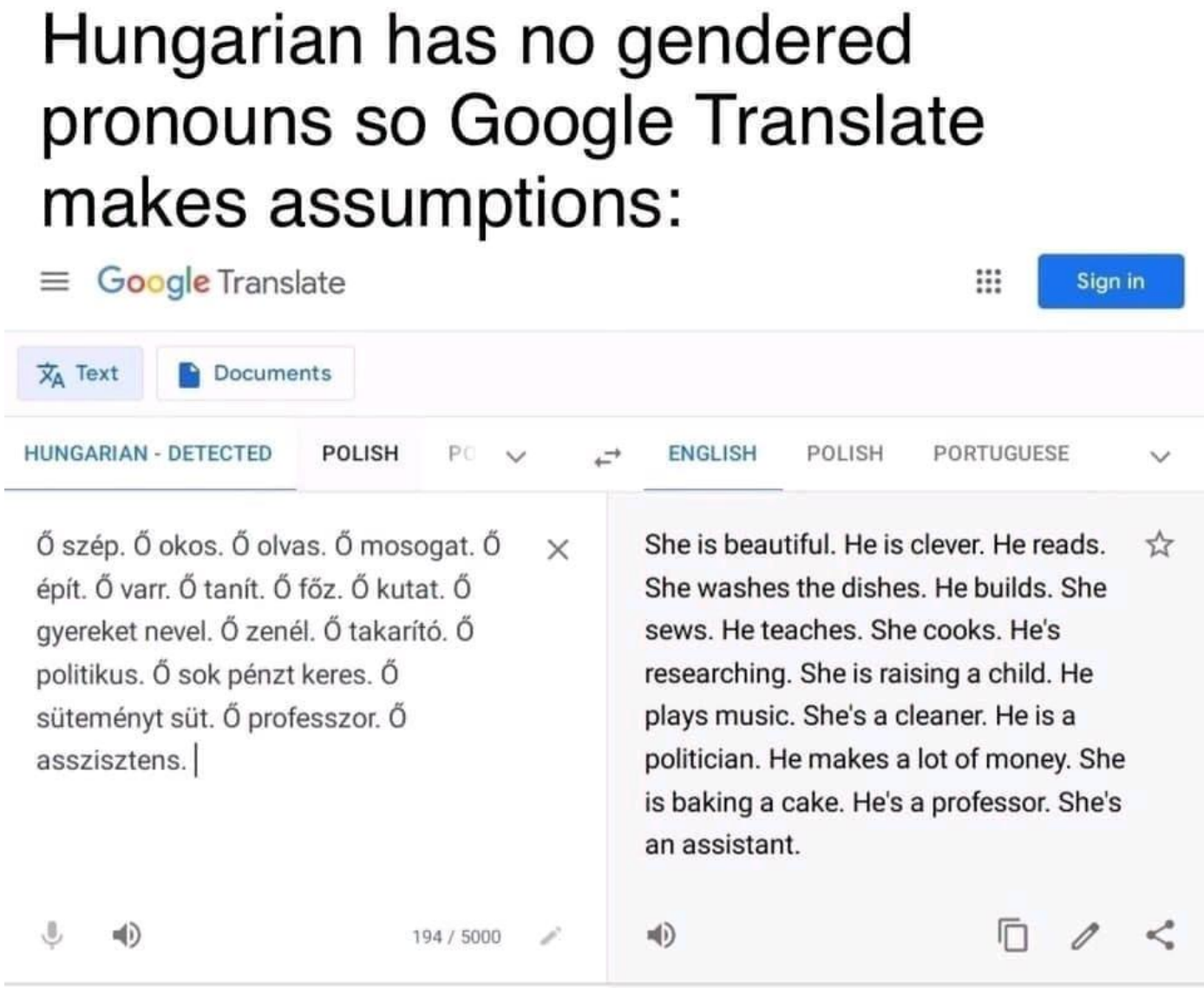
Deep Learning

9. Problems with embeddings



- Espresso? But I ordered a cappuccino!
- Don't worry, the cosine distance between them is so small that they are almost the same thing.

Gender bias (1)



Gender bias (2)

- Embeddings can amplify stereotypes

$$\overrightarrow{\text{man}} - \overrightarrow{\text{woman}} \approx \overrightarrow{\text{king}} - \overrightarrow{\text{queen}}$$

$$\overrightarrow{\text{man}} - \overrightarrow{\text{woman}} \approx \overrightarrow{\text{computer programmer}} - \overrightarrow{\text{homemaker}}$$

[Bolukbasi et al. 2016]

Extreme *she* occupations

1. homemaker	2. nurse	3. receptionist
4. librarian	5. socialite	6. hairdresser
7. nanny	8. bookkeeper	9. stylist
10. housekeeper	11. interior designer	12. guidance counselor

Extreme *he* occupations

1. maestro	2. skipper	3. protege
4. philosopher	5. captain	6. architect
7. financier	8. warrior	9. broadcaster
10. magician	11. figher pilot	12. boss

Figure 1: The most extreme occupations as projected on to the *she*−*he* gender direction on g2vNEWS. Occupations such as *businesswoman*, where gender is suggested by the orthography, were excluded.

Bolukbasi et al. 2016

Gender stereotype *she-he* analogies.

sewing-carpentry	register-nurse-physician	housewife-shopkeeper
nurse-surgeon	interior designer-architect	softball-baseball
blond-burly	feminism-conservatism	cosmetics-pharmaceuticals
giggle-chuckle	vocalist-guitarist	petite-lanky
sassy-snappy	diva-superstar	charming-affable
volleyball-football	cupcakes-pizzas	hairdresser-barber

Gender appropriate *she-he* analogies.

queen-king	sister-brother	mother-father
waitress-waiter	ovarian cancer-prostate cancer	convent-monastery

Figure 2: **Analogy examples**. Examples of automatically generated analogies for the pair *she-he* using the procedure described in text. For example, the first analogy is interpreted as *she*:*sewing* :: *he*:*carpentry* in the original w2vNEWS embedding. Each automatically generated analogy is evaluated by 10 crowd-workers are to whether or not it reflects gender stereotype. Top: illustrative gender stereotypic analogies automatically generated from w2vNEWS, as rated by at least 5 of the 10 crowd-workers. Bottom: illustrative generated gender-appropriate analogies.

Bolukbasi et al. 2016

Sources of Bias

- Bias in the labeling
- Sample selection bias
- Bias in choice of labels
- Bias in features or model structure
- Bias in loss function
- Deployed systems create feedback loops

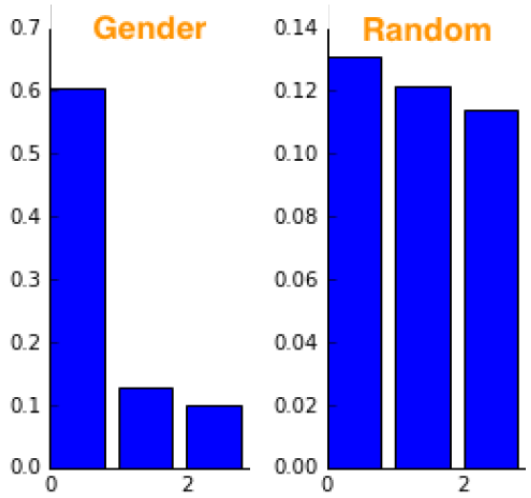
[slide from Marine Carpuat]

Bias laundering

“Instead of relying on algorithms, which we can be accused of manipulating for our benefit, we have turned to machine learning, an ingenious way of disclaiming responsibility for anything. Machine learning is like money laundering for bias. It's a clean, mathematical apparatus that gives the status quo the aura of logical inevitability. The numbers don't lie.”

- Maciej Cegłowski

	def.	stereo.
$\overrightarrow{\text{she}} - \overrightarrow{\text{he}}$	92%	89%
$\overrightarrow{\text{her}} - \overrightarrow{\text{his}}$	84%	87%
$\overrightarrow{\text{woman}} - \overrightarrow{\text{man}}$	90%	83%
$\overrightarrow{\text{Mary}} - \overrightarrow{\text{John}}$	75%	87%
$\overrightarrow{\text{herself}} - \overrightarrow{\text{himself}}$	93%	89%
$\overrightarrow{\text{daughter}} - \overrightarrow{\text{son}}$	93%	91%
$\overrightarrow{\text{mother}} - \overrightarrow{\text{father}}$	91%	85%
$\overrightarrow{\text{gal}} - \overrightarrow{\text{guy}}$	85%	85%
$\overrightarrow{\text{girl}} - \overrightarrow{\text{boy}}$	90%	86%
$\overrightarrow{\text{female}} - \overrightarrow{\text{male}}$	84%	75%



	RG	WS	analogy
Before	62.3	54.5	57.0
Hard-debiased	62.4	54.1	57.0
Soft-debiased	62.4	54.2	56.8

Figure 2: **Left:** Ten word pairs to define gender, along with agreement with sets of definitional and stereotypical words solicited from the crowd. The accuracy is shown for the corresponding gender classifier based on which word is closer to a target word, e.g., the *she-he* classifier predicts a word is female if it is closer to *she* than *he*. **Middle:** The bar plot shows the percentage of variance explained in the PCA of the 10 pairs of gender words. The top component explains significantly more variance than any other; the corresponding percentages for random words shows a more gradual decay (Figure created by averaging over 1,000 draws of ten random unit vectors in 300 dimensions). **Right:** The table shows performance of the original w2vNEWS embedding ("before") and the debiased w2vNEWS on standard evaluation metrics measuring coherence and analogy-solving abilities: RG [27], WS [10], MSR-analogy [21]. Higher is better. The results show that the performance does not degrade after debiasing. Note that we use a subset of vocabulary in the experiments. Therefore, the performances are lower than the previously published results. See Appendix for full results.

Figure 3: Selected words projected along two axes: x is a projection onto the difference between the embeddings of the words *he* and *she*, and y is a direction learned in the embedding that captures gender neutrality, with gender neutral words above the line and gender specific words below the line. Our hard debiasing algorithm removes the gender pair associations for gender neutral words. In this figure, the words above the horizontal line would all be collapsed to the vertical line.

Three steps (Bolukbasi et al. 2016)

- Find gender neutral words with biases in the original embeddings
- Identify the gender-specific space V and its orthogonal complement $V^{\perp}$.
- Project embeddings of the gender neutral words to the subspace $V^{\perp}$.

The first step, called **Identify gender subspace**, is to identify a direction (or, more generally, a subspace) of the embedding that captures the bias. For the second step, we define two options: **Neutralize and Equalize** or **Soften**. **Neutralize** ensures that gender neutral words are zero in the gender subspace. **Equalize** perfectly equalizes sets of words outside the subspace and thereby enforces the property that any neutral word is equidistant to all words in each equality set. For instance, if {grandmother, grandfather} and {guy, gal} were two equality sets, then after equalization *babysit* would be equidistant to *grandmother* and *grandfather* and also equidistant to *gal* and *guy*, but presumably closer to the grandparents and further from the *gal* and *guy*. This is suitable for applications where one does not want any such pair to display any bias with respect to neutral words.

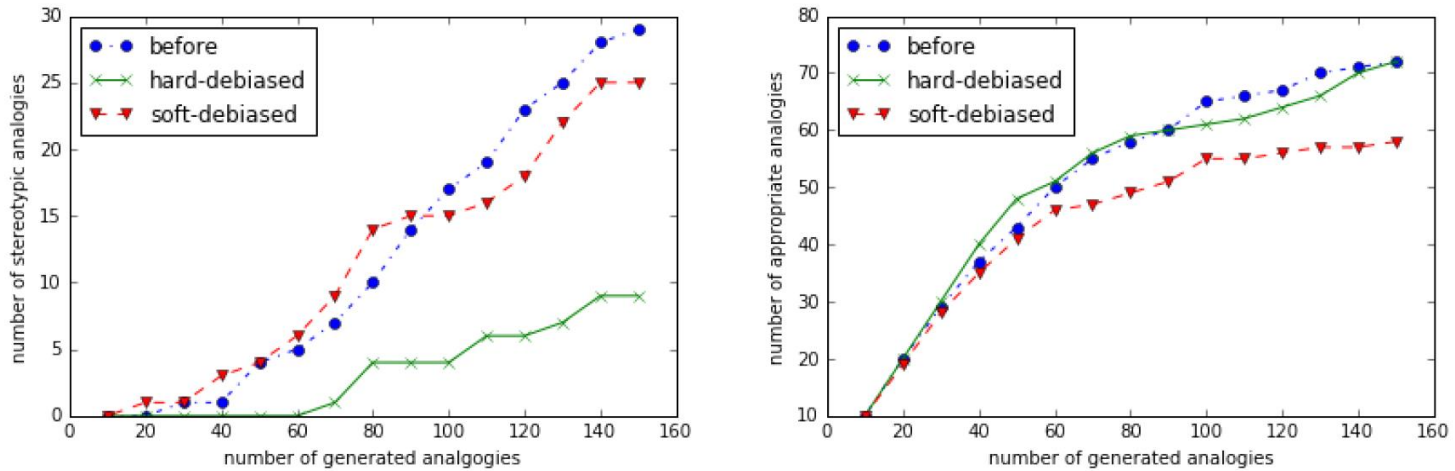


Figure 4: Number of stereotypical (Left) and appropriate (Right) analogies generated by word embeddings before and after debiasing.

Links

- https://ruder.io/word-embeddings-1/index.html
- https://ruder.io/word-embeddings-softmax/index.html
- https://ruder.io/secret-word2vec/index.html
- https://ruder.io/cross-lingual-embeddings/index.html

Deep Learning

10. Subword Embeddings

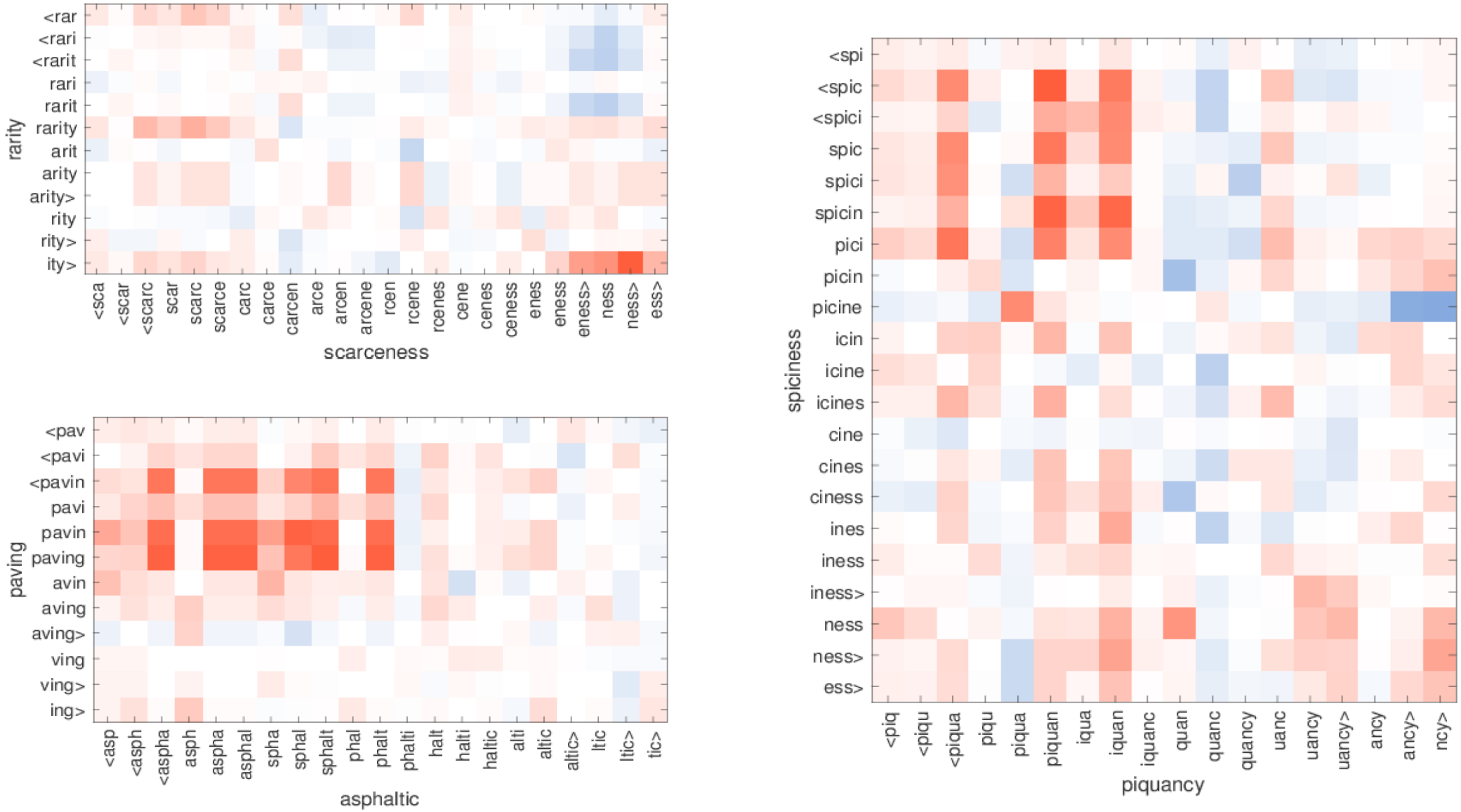


Figure 2: Illustration of the similarity between character n-grams in out-of-vocabulary words. For each pair, only one word is OOV, and is shown on the x axis. Red indicates positive cosine, while blue negative.

More about FastText

- https://fasttext.cc/docs/en/unsupervised-tutorial.html
- https://towardsdatascience.com/fasttext-ea9009dba0e8
- https://stackabuse.com/python-for-nlp-working-with-facebook-fasttext-library/
- https://pypi.org/project/fasttext/
- https://www.analyticsvidhya.com/blog/2017/07/word-representations-text-classification-using-fasttext-nlp-facebook/
- https://hackernoon.com/text-classification-simplified-with-facebooks-fasttext-b9d3022ac9cb
- https://www.geeksforgeeks.org/fasttext-working-and-implementation/
- https://amitness.com/2020/06/fasttext-embeddings/

Deep Learning

Conclusion

Using Embeddings

- Learning the parameters from the corpus
- Use existing embeddings
- Start with existing embeddings, then update

Summary

- Word embeddings perform matrix factorization of the co-occurrence matrix
- Word2vec is a simple feed-forward neural network
- Training is done using backpropagation using SGD
- Negative sampling for training

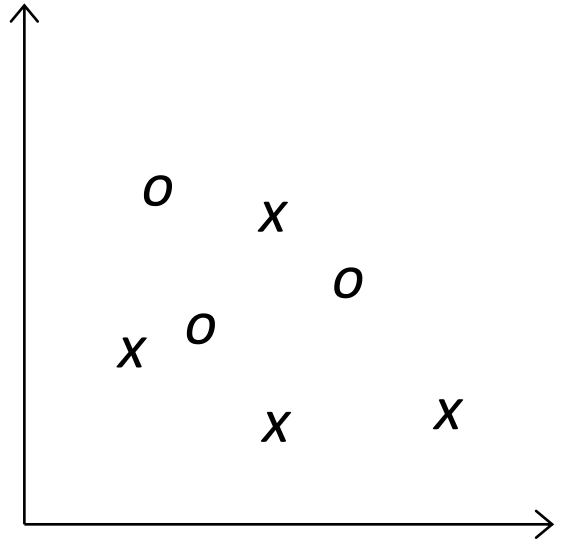
Bonus Topic

342. Text Kernels

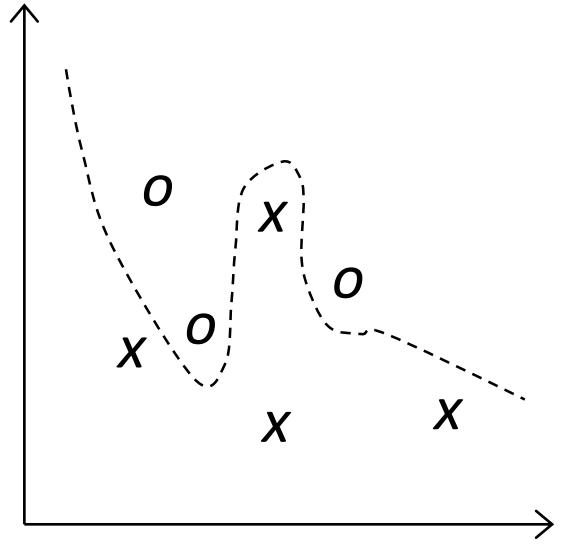
Unigram Kernels

- Same as a bag of words
- For example, we can split the sentence “Google bought Kaggle” into the three unigrams:
  - “Google”
  - “bought”
  - “Kaggle”
- Similarly, we can split “Kaggle bought Google” into:
  - “Kaggle”
  - “bought”
  - “Google”
- However, the vector representations of the two sentences will be identical (<1,1,1>) and therefore their cosine similarity will be computed as 1.

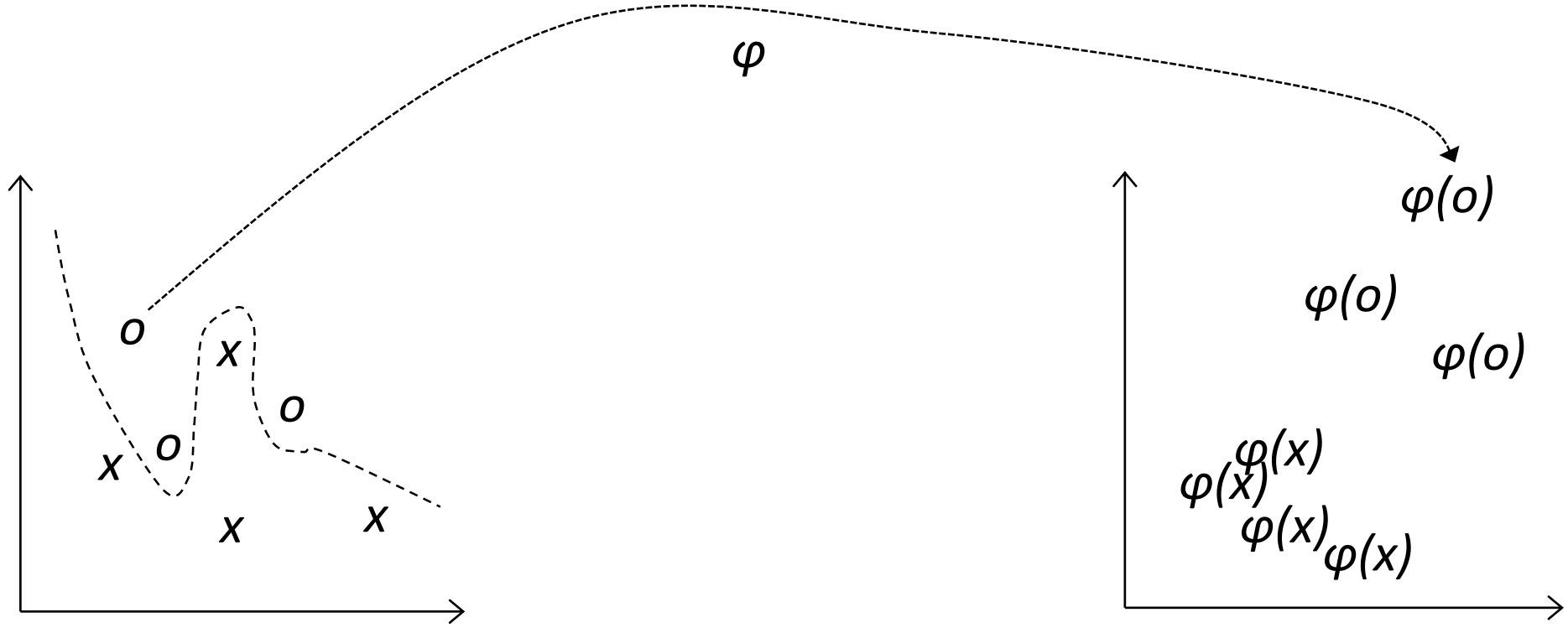
The Basic Idea behind Kernels



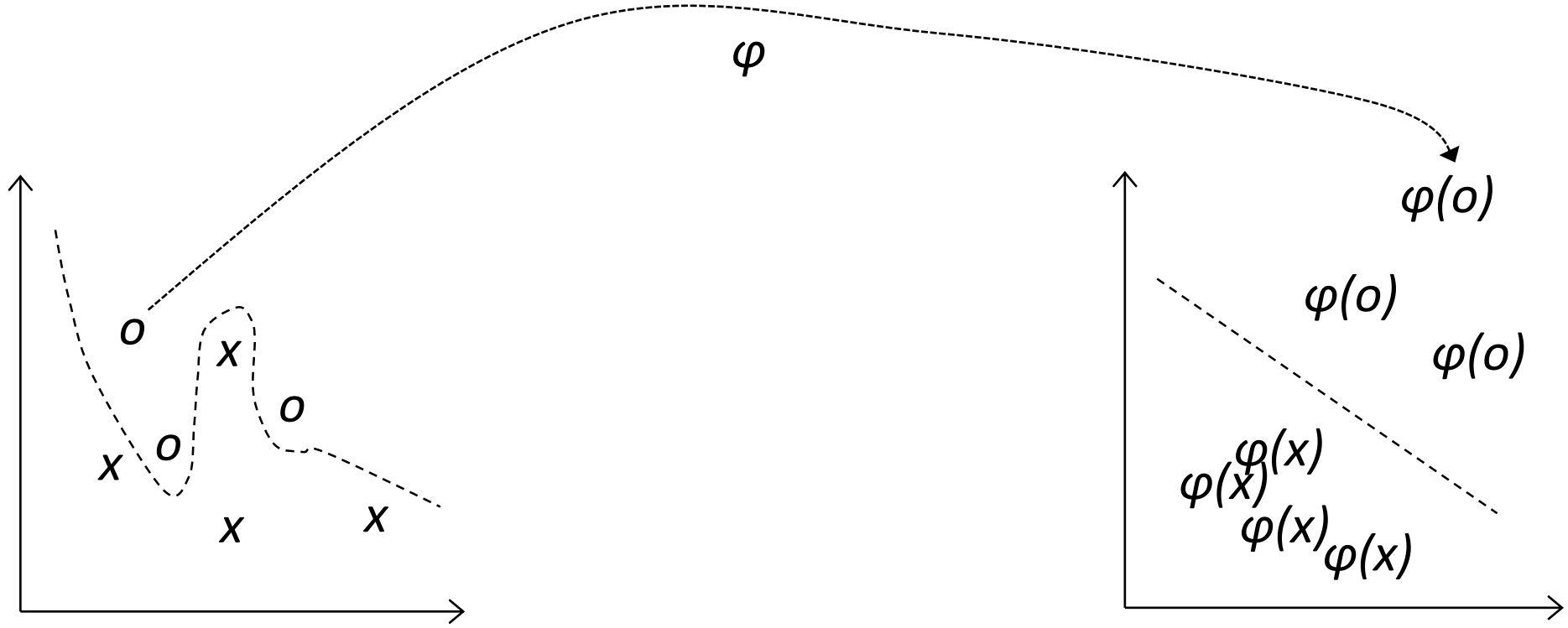
The Basic Idea behind Kernels



The Basic Idea behind Kernels



The Basic Idea behind Kernels



Bigram Kernels

- Using a bigram kernel, we can split the sentence “Google bought Kaggle” into the two bigrams:
  - “Google bought”
  - “bought Kaggle”
- Similarly, we can split “Kaggle bought Google” into:
  - “Kaggle bought”
  - “bought Google”
- Using this representation, the vectors corresponding to the two sentences are <1,1,0,0> and <0,0,1,1>, respectively, and their cosine similarity is 0.

BLEU Example

One hundred artists from 16 countries will exhibit 270 pieces of **artistic work made on porcelain** in the Palace of Arts in Cairo in a an exhibition inaugurated earlier this week and that will last for two weeks.

One hundred artists from 61 states are exhibiting 270 pieces of **porcelain artwork** at the Arts Palace in Cairo, as part of a two-week exhibition that opened at the beginning of this week.

100 artists from 16 countries are exhibiting 270 pieces of work **on porcelain** in Cairo's Arts Palace as part of a 2-week exhibition, which opened at the beginning of the week.

One hundred artists from 16 countries will display 270 pieces of art works **undertaken on porcelain** in the Palace of Arts in Cairo in an exhibition opened at the beginning of this week and which will continue for two weeks.

A hundred artists from 16 countries will exhibit 270 pieces of artistic works **that are made on Porcelain** in the arts palace in Cairo in a an exhibition inaugurated earlier this week, which will last for two weeks.

One hundred artists from 61 countries exhibited 270 types of art pieces **done on porcelain** in the Cultural Palace of Cairo, in an exhibition that was inaugurated at the beginning of the current week. The exhibition would last two weeks.

One hundred artists from 16 countries will exhibit 270 works of art **made with porcelain** in the Arts Castle in Cairo in an exhibition that opened earlier this week and will last for two weeks.

A hundred artists from 16 countries are displaying 270 **porcelain** art pieces in an exhibition that opened early in the week, and will continue over a span of two weeks.

A hundred artists from 16 countries display 270 artistic **porcelain** pieces at the Palace of Arts in Cairo, in an exhibition that was opened earlier this week and continuing for two weeks.

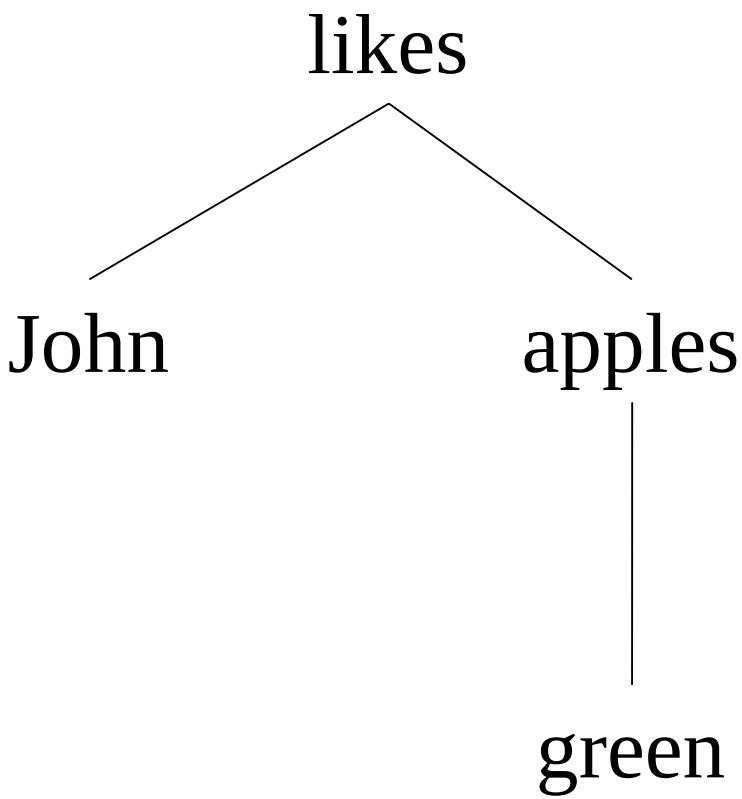
Letter and Substring Kernels

- Letter n-grams can be used for various applications such as spelling correction, language recognition, and named entity recognition.
- For example, the word *stop* can be represented as the set of all of its substrings: s, t, o, p, st, to, op, sto, top, and stop.
- In this representation, sim(stop,stops) > sim(stop,plot), even though all three words are different.

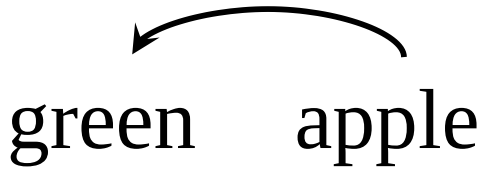
Subsequences

- Unlike a substring, a subsequence doesn't need to consist of contiguous words (or letters).
- *comp*, *cotr*, *opter*, *cpute* – all of these are letter-based subsequences of *computer*
- Subsequence kernels (of words, not letters) are most useful for measuring similarity between sentences.

Dependencies

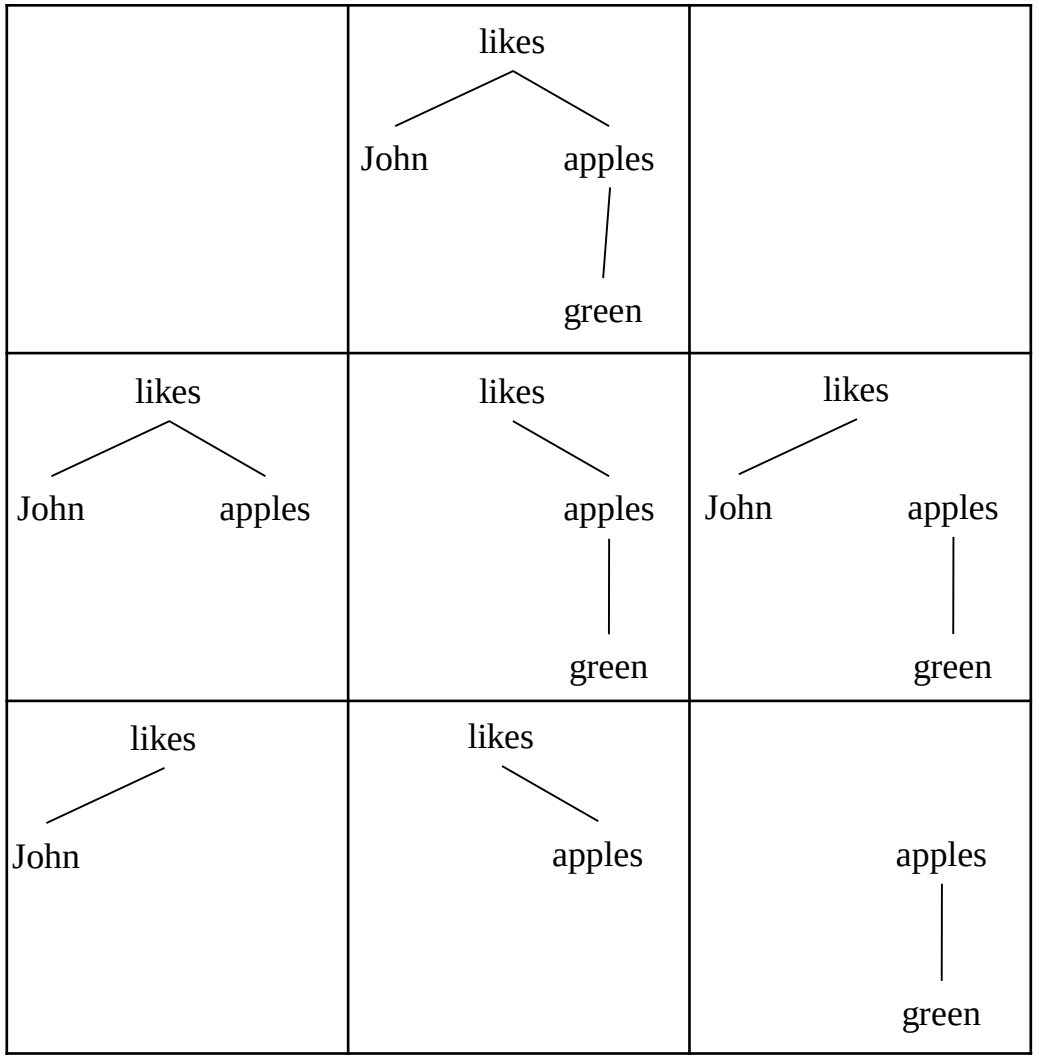


Dependencies



- *green*: modifier, child
- *apple*: head, parent

The Dependency Tree Kernel



Dependency Tree Kernel Example

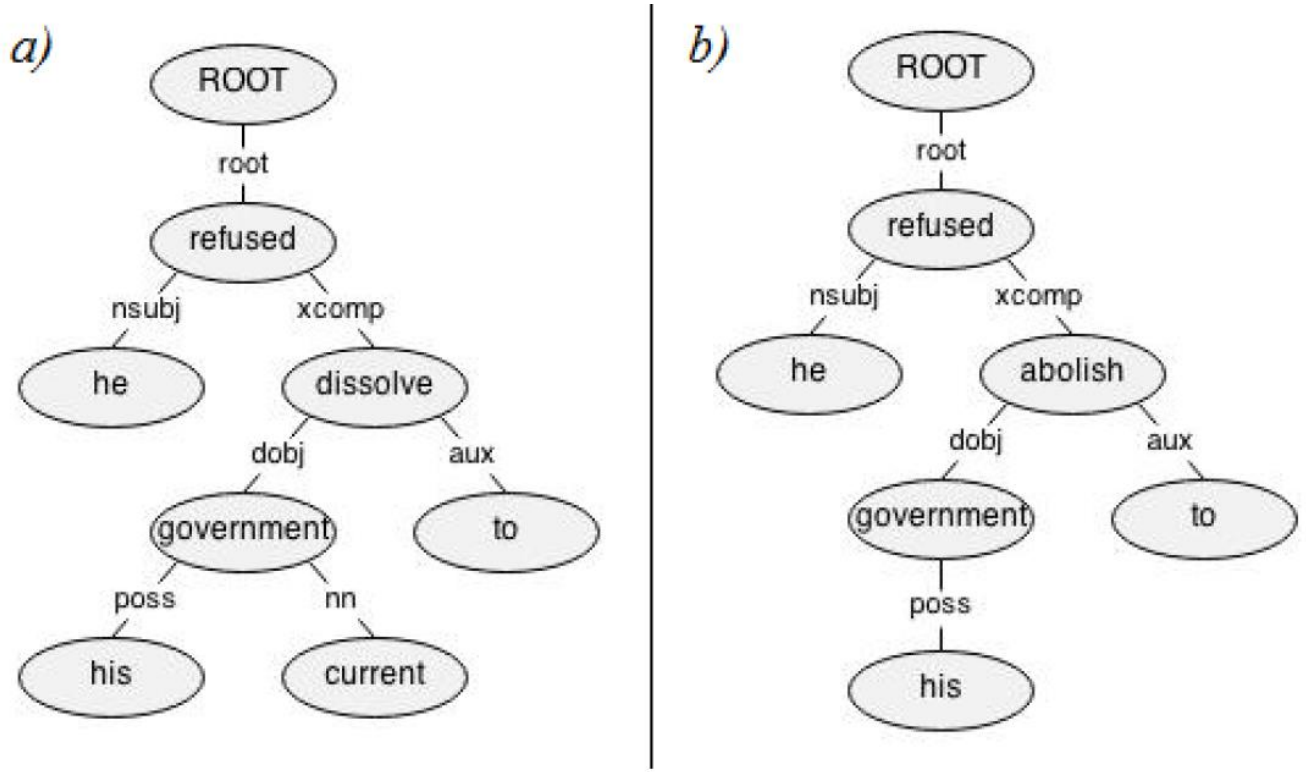
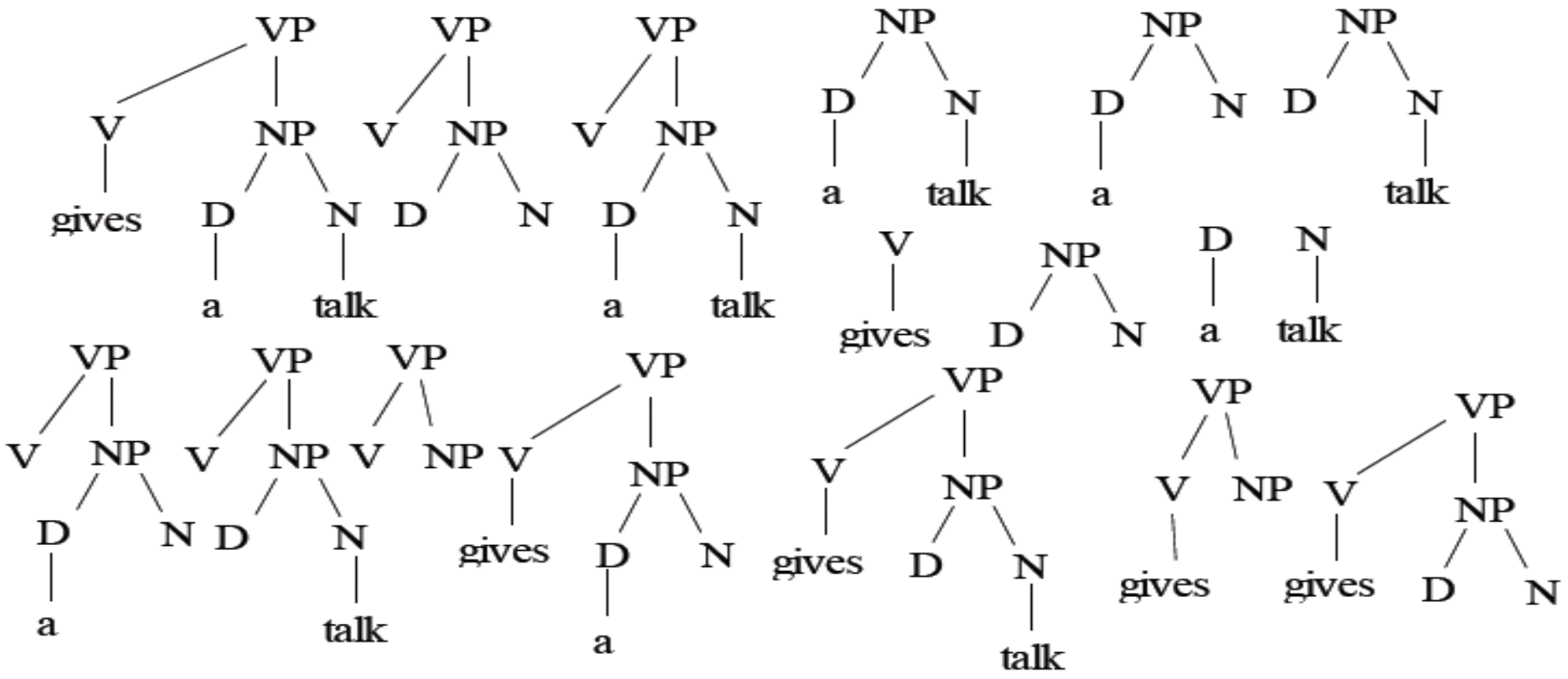


Figure 1: Typed dependency trees of the sentences *"He refused to dissolve his current government."* and *"He refused to abolish his government."*, respectively.

[Özateş, Özgür, and Radev, LREC 2016]

The Syntactic Tree Kernel



Note that siblings are not split up (Collins and Duffy, 2002)

NLP