ML for Human Vision and Language

MSc Artificial Intelligence

Lecturer: Tejaswini Deoskar
t.deoskar@uu.nl

UiL-OTS, Utrecht University

Block 4, 2019

---

some slides by S. Goldwater, E. Shutova, C. Manning, and Jurafsky & Martin

Meaning (Semantics)

- Compositional semantics

They don't work very well yet with vectors

  - how meanings of phrases are constructed out of the meaning of individual words
    - Principle of compositionality: meaning of each whole phrase is derivable from the meaning of its parts
    - most commonly symbolic and logic based approaches used to build meaning representations

- Lexical semantics
  - how the meanings of individual words can be represented/learnt
  - vector-based approaches very successful recently
    - Open-class words : nouns, verbs, adjectives
    - "Logical words" : *not, many, every, a, few ...*

very good here, but not good at methodological words

What is lexical meaning?

What do we want to do when we want to model the meaning of words?

- we don't have the whole picture yet, but results in psychology and cognitive neuroscience as well as deep learning give us some clues
- different representations proposed, e.g.
  - formal semantic representations based on logic
  - Prototype theory
  - taxonomies relating words to each other
  - distributional (vector) based representations

FOCUS is on distributional representations

None of the representations gives us a complete account of lexical meaning as yet.

How to approach lexical meaning?

Formal semantics:

- set -theoretic approach, e.g. cat' : the set of all cats; dog' : the set of all dogs
- meaning postulates, e.g.

$$\forall x[bachelor'(x) \rightarrow male'(x) \wedge unmarried(x)]$$

- Defining concepts through enumeration of all of their features is highly problematic.

  * Very hard as one cannot define all features of a word (e.g. male, married unmarried)

How to approach lexical meaning?

Prototype Theory

- certain members of a category are more central or prototypical (instantiate the prototype)

  bird : sparrow is more prototypical than penguin

- introduced the notion of graded semantic categories
- no clear boundaries
- no requirement that a property or set of properties be shared by all members

  * It's a looser idea

Eleanor Rosch 1975. *Cognitive Representation of Semantic Categories* (J Experimental Psychology)

Thesaurus based meaning

WordNet : a thesaurus containing lists of **synonym sets** and **hypernyms** ("X is a Y" relations)

*Still used today

synonym set containing good

```
noun: good
noun: good, goodness
noun: good, goodness
noun: commodity, trade_good, good
adj: good
adj (sat): full, good
adj: good
adj (sat): estimable, good, honorable, respectable
adj (sat): beneficial, good
adj (sat): good
adj (sat): good, just, upright
…
adverb: well, good
adverb: thoroughly, soundly, good
```

hypernyms of panda

```
[Synset('procyonid.n.01'),
Synset('carnivore.n.01'),
Synset('placental.n.01'),
Synset('mammal.n.01'),
Synset('vertebrate.n.01'),
Synset('chordate.n.01'),
Synset('animal.n.01'),
Synset('organism.n.01'),
Synset('living_thing.n.01'),
Synset('whole.n.02'),
Synset('object.n.01'),
Synset('physical_entity.n.01'),
Synset('entity.n.01')]
```

Problems with resources like WordNet

- Great as a resource but not fine-grained enough
  - e.g. proficient is listed as a synomym of good, but this is correct only in certain contexts
- Missing new meanings
  - e.g. wicked, badass
- Subjective ; Require human labour
- Cannot compute word similarity effectively.
- Still widely used; **retrofitting** word-vectors using wordnet helps

Problem with representing words as discrete symbols

- Traditionally, we regard words as discrete symbols: hotel, motel
- Words can be represented by one hot vectors

motel = [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]
hotel = [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]

vector's dimension is equal to all the words in the vocabulary
[Vector dimension = number of words in vocabulary (e.g. 50,000)]

- There is no natural notion of similarity for one-hot vectors! These two vectors are orthogonal.
- **Instead**: learn to encode similarity in the vectors themselves!

Meaning from context(s)

What's the meaning of *'bardiwac'*?

- He handed her her glass of bardiwac.
- Beef dishes are made to complement the bardiwacs.
- Nigel staggered to his feet, face flushed from too much bardiwac.
- Malbec, one of the lesser-known bardiwac grapes, responds well to Australia's sunshine.
- I dined on bread and cheese and this excellent bardiwac.
- The drinks were delicious: blood-red bardiwac as well as light, sweet Rhenish.

$\Rightarrow$ *'bardiwac'* is a heavy red alcoholic beverage made from grapes

We are able to guess the meaning of *'bardiwac'* because the contexts in which it occurs (its distribution) are similar to the contexts where other known words appear:

similar distributions $\Leftrightarrow$ similar meaning

distribution = the context in which the word occurs

Distributional Hypothesis

"The degree of semantic similarity between two linguistic expressions A and B is a function of the similarity of the linguistic contexts in which A and B can appear." [Z. Harris (1954) *Distributional Structure*]

You shall know a word by the company it keeps (J. R. Firth, 1957)

Distributional Semantics

- Goal : find a representation that succinctly describes the meaning of a word (or phrase, sentence, document)
- Or at least : find a representation so as to determine if two words or texts have similar meanings

- Why is this a good thing?
  - Retrieve documents relevant to a query
  - Use representations as features for parsing, question answering, machine translation
  - Theory neutral, few assumptions re word meaning

Representing words by their contexts

- When a word w appears in a text, its context is the set of words that appear nearby

…government debt problems turning into ***banking*** *crises as happened in 2009…*
…saying that Europe needs unified ***banking*** *regulation to replace the hodgepodge…*
…India has just given its ***banking*** *system a shot in the arm…*

These context words will represent ***banking***

- Use the many contexts of w to build up a vector representation of w

	regulation	system	crises	debt	pet	bone	fur	fetch
banking =	1	1	1	1	0	0	0	0
⋮								
dog =	0	0	0	0	1	1	1	1

Each dimension of the vector is a context word (here, boolean valued)

Distributional semantics

Distributional semantics: family of techniques for representing word meaning based on (linguistic) contexts of use.

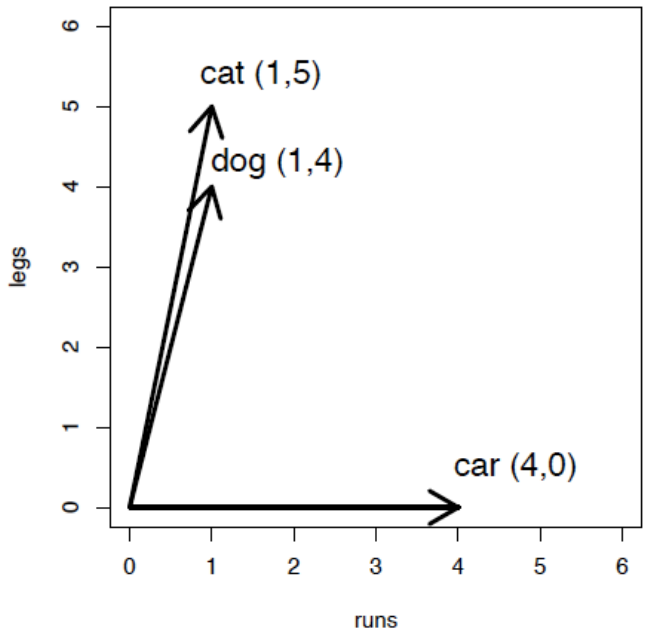
1. Count-based models: Booleans or counts/frequencies
   * count frequency of co-occurrence of a target and a context
   * words are represented as vectors (sparse)
   * dimensions correspond to elements in the context
2. Prediction models:
   * try to predict plausible contexts for a word
   * learn word representations in the process
   * generally implemented as a neural network

The semantic space

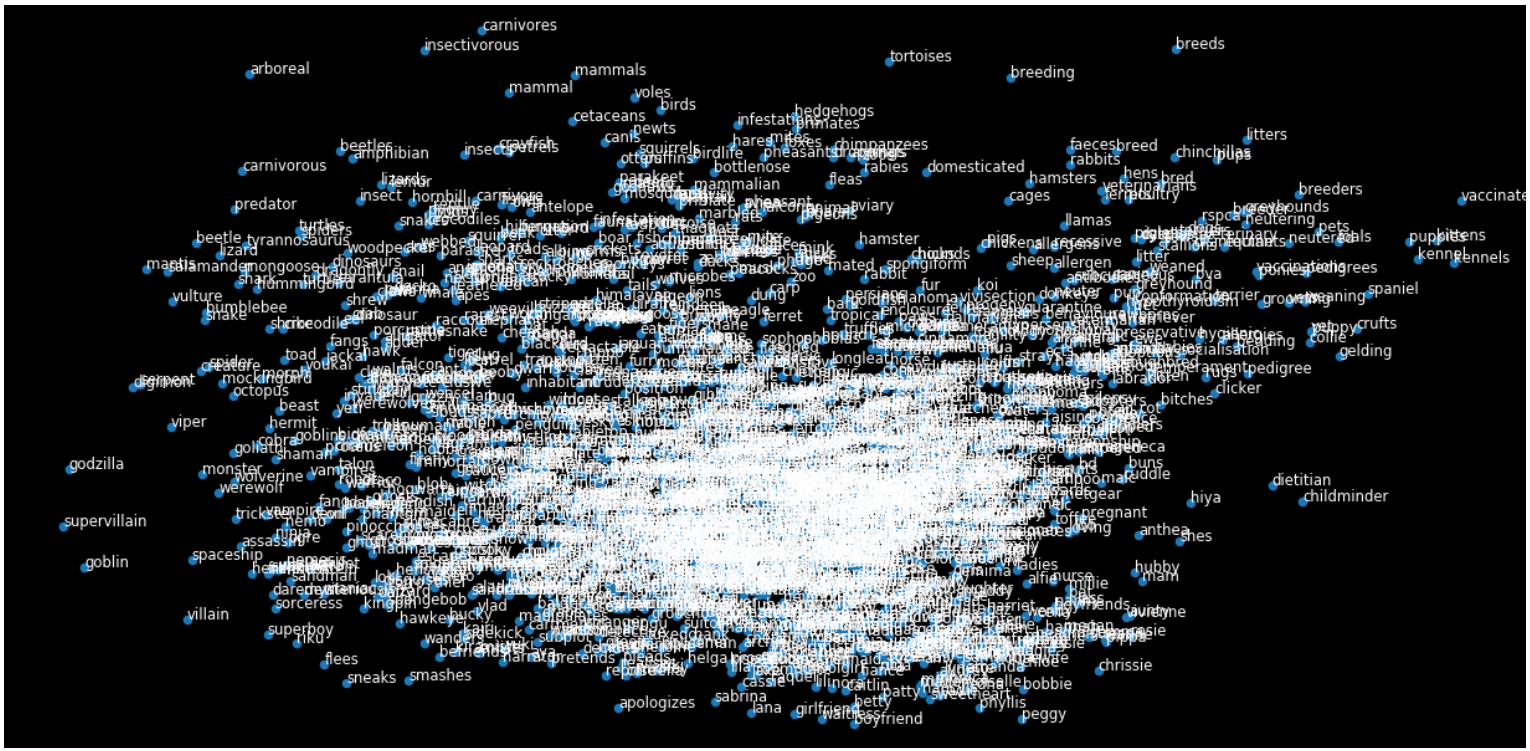
- Vectors exist in a vector space - the semantic space.
- The semantic space has dimensions that correspond to possible contexts - features of the semantic space.
- a distribution is a point in this space (the vector being defined with respect to the origin of that space)

Two dimensional example:

	run	legs
dog	1	4
cat	1	5
car	4	0



The semantic space



A space with 10,000 word vectors, flattened from 50,000 to 2 dimensions.

Questions to consider / modelling choices

Assignment 2B

- What defines "context"?
- How to weight the context words ?
  - Boolean?
  - Counts?
  - Other?
- How to measure similarity between vectors?

Modelling choices: context

- Word windows (unfiltered) : n words on either side of the lexical item.
- Word windows (filtered): n words on either side removing some words, e.g. function words, some very frequent content words **stopwords**
- Lexeme window: As above, but use *stems* .
- Dependencies: Context for a word might be the dependency structure it belongs to (Pado and Lapata, 2007)

*The prime **minister** acknowledged the question.*

minister [prime_a 1, acknowledge_v 1]

minister [prime_a_mod 1, acknowledge_v_subj 1]

minister [prime_a 1, acknowledge_v+question_n 1]

* These are methods to add more information than the context itself

Modelling choices: context

- For *semantic* similarity, use a large window (say 100 words); for *syntactic* similarity, use a small window (1-3 words) and track only frequent words
- Perceptual information: images, sounds, etc. to complement the conceptual data given by language alone

Modelling choices : Context weighting

- Binary model : if context c co-occurs with word w, value of vector $\vec{w}$ for dimension c is 1, 0 otherwise.

- Basic frequency model: value of vector $\vec{w}$ for dimension c is the number of times c co-occurs with w

- Is frequency good enough?
  - frequent words are expected to have high counts in the context vector, regardless of whether they occur more often with this word than with others
  - We want to know which words occur **unusually** often in the context of a target word w: more than we'd expect by chance.
    - **collocations**: words or terms that co-occur more often than would be expected by chance

Collocations/ PMI

- We want to know which words occur **unusually** often in the context of w_i: more than we'd expect by chance.

- Put another way, what **collocations** include w_i?
- collocations: words or terms that co-occur more often than would be expected by chance

Pointwise Mutual information

- One way: use pointwise mutual information:

$$PMI(x,y) = log_2 \frac{P(x,y)}{P(x)P(y)}$$

  * Numerator: Actual prob. of seeing words x and y together
  * Denominator: Predicted prob. of same, if x and y are independent.

- PMI tells us how much more or less likely the cooccurrence is, than if the words were independent : Good collocation pairs have high PMI

- Derivative (Positive PMI) used in practise were independent.

The more higher the collocations value the more informative this couple of words is

PPMI : improved PMI

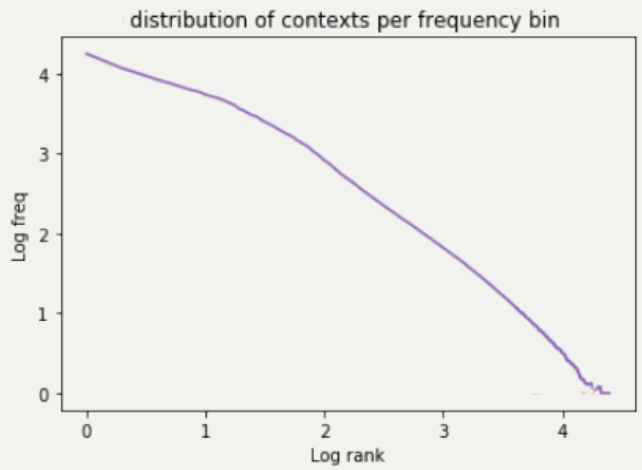
Modify PMI to better handle low frequencies.

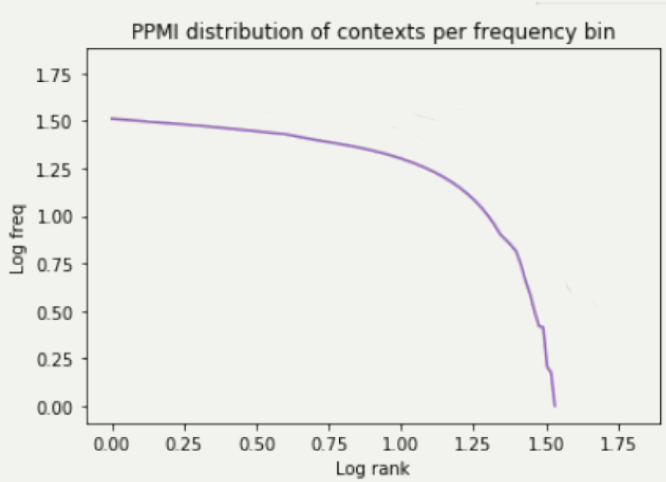
- Use positive PMI (PPMI): change all negative PMI values to 0.
  - Because for infrequent words, not enough data to accurately determine negative PMI values.

Alternatives to PMI for finding collocations

- There are a lot, all ways of measuring statistical (in)dependence.
  - Student t-test
  - Pearson's X^2 statistic
  - Dice co-efficient
  - Likelihood Ratio test (Dunning, 1993)
  - Lin association measure (Lin, 1998)
  - and many more ...
- Of those listed here, Dunning LR test probably most reliable for low counts.
- However, which works best may depend on particular application/evaluation.

De-Zipfianising with PPMI





PPMI drastically reduces the weight of very frequent words,flattening the original Zipfian distribution.

We pay *less* attention to frequent events, *more* to infrequent ones.

- PPMI reduces the weight of very frequent words, flattening the original distribution
- We pay less attention to frequent events, more to infrequent ones.
- Contrary to the 'pure' view of distributional semantics (*You shall*

How much data?

- As much data as possible?
  - British National Corpus (BNC): 100M words
  - Wikipedia: 900M words
  - Google News: 100bn words
- In general preferable, but
  - More data is not realistic from a psychological point of view.

Vectors vs human meaning

Machine exposed to:	**3-year old child exposed to:**
100M words (BNC)	25M words (US)
2B words (UKWaC)	20M words (Dutch)
100B words (Google News)	5M words (Mayan)
	(*Cristia et al 2017*)

What semantic space?

- Entire vocabulary
  - All information included - even rare contexts
  - Inefficient (100,000s dimensions).
  - Noisy and sparse (lots of zeroes)
- Top n words with highest frequency
  - More efficient (2000-10000 dimensions)
  - May miss out on rare but relevant contexts

What semantic space?

- Singular Value Decomposition (SVD): the number of dimensions is reduced by exploiting redundancies in the data.
  - Very efficient (200-500 dimensions). Captures generalisations in the data.
  - SVD matrices are not interpretable
- Principle Component Analysis (PCA), Non-negative matrix factorization (NMF) : uses a different factorization

Evaluation of DS models

- Default: Similarity and relatedness
- Linguistic evaluations : modelling lexical relations (synonymy, antonymy, ...)
- Cognitive evaluations : brain decoding/encoding

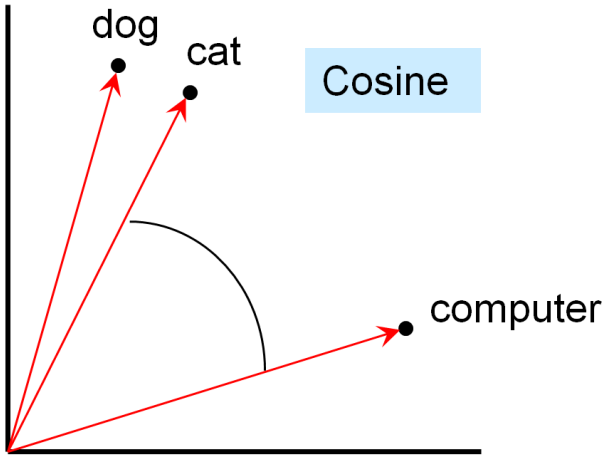
How to measure similarity

- So, let's assume we have context vectors for two words $\vec{v}$ and $\vec{w}$
- Each contains PMI (or PPMI) values for all context words
- One way to think of these vectors: as points in high-dimensional space
- How to measure similarity?

Normalized dot product =cosine

most common measure of similarity

- The normalized dot product is the cosine of the angle between vectors.



$$PMI(x,y) = log_2 \frac{P(x,y)}{P(x)P(y)}$$

$$sim_{DP}(\vec{v},\vec{w}) = \frac{\vec{v}\cdot\vec{w}}{|\vec{v}||\vec{w}|}$$

- Ranges from -1 (vectors pointing opposite directions) to 1 (same direction)

Other similarity measures

- Again, many alternatives
  - Jaccard measure
  - Dice measure
  - Jenson-Shannon divergence
- Again, may depend on particular application/evaluation

But what is similarity

- In DS, very broad notion: synonyms, near-synonyms, hyponyms, taxonomical siblings, antonyms, etc.
- Correlates with a psychological reality
- Evaluate via correlation with human judgments on a testset

—> calculate similarity with cosine and then correlate with reality

Relatedness

Reproduce human relatedness judgements (expressed as a score on a bounded scale)

- Participants are asked, e.g.: on a scale of 1-10, how related are the following concepts?

LEMON FLOWER

- Usually given some examples initially to set the scale

LEMON-TRUTH	1
LEMON-ORANGE	10

- Answers depend a lot on how the question is asked ('related'? vs. 'similar'? vs. other terms)

Word association

- Participants see/hear a word, say the first word that comes to mind
- Data collected from lots of people provides probabilities of each answer:

Using free association

		ORANGE	0.16
		SOUR	0.11
		TREE	0.09
LEMON	$\Rightarrow$	YELLOW	0.08
		TEA	0.07
		JUICE	0.05
		...	

Example data from the Edinburgh Associative Thesaurus: `http://www.eat.rl.ac.uk/`

Comparing to human data

- Human judgments provide a ranked list of related words (or associations) for each word w
- Computer system provides a ranked list of most "similar" words to w
- Compute the Spearman rank correlation between the lists (how well do the rankings match?)
- Often report on several data sets, as their details differ
  - Rubenstein & Goodenough (1965): 65 noun pairs
  - Finkelstein et al (2002): WordSim353
  - Bruni et al (2014): MEN (3000 pairs)

  Current human correlation on the MEN dataset is over 0.7

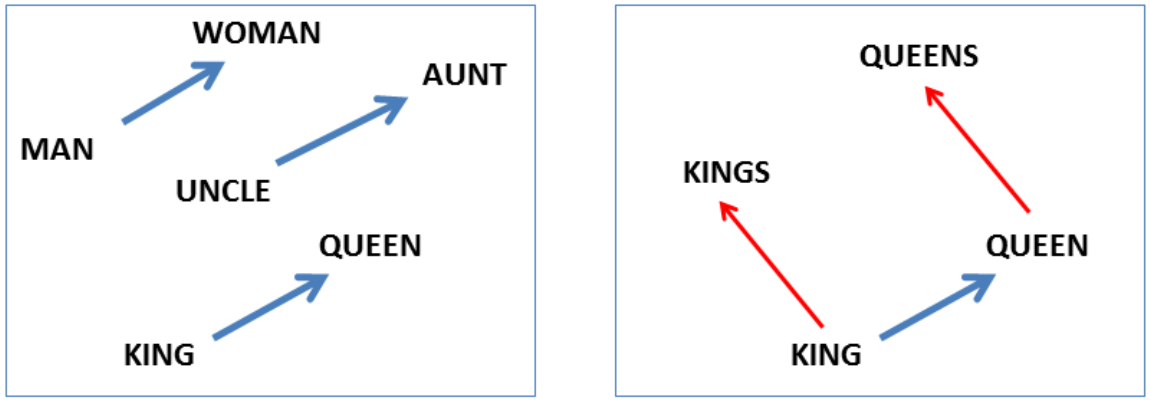
Similarity vs relatedness

- SimLex-999 (Hill et al.) encodes similarity rather than relatedness
  cat - purr are related (occur in the same contexts)
  cat - dog are similar (substitutable in many contexts, share properties)
- Similarity is much harder to get out of distributional spaces and requires additional processing, such as 'retrofitting' (Faruqui et al, 2014), which injects information from lexical resources into vectors

Linguistic relations : Analogy

Capture analogy via vector offsets

$$man - woman \approx king - queen$$



Mikolov et al. 2013. *Linguistic Regularities in Continuous Space Word Representations*

Analogy

They capture analogy

Analogy task: ***a*** *is to* ***b*** *as* ***c*** *is to* ***d***
The system is given words a, b, c, and it needs to find d.

"apple" is to "apples" as "car"' is to ?
"man" is to "woman" as "king" is to ?

Solution: capture analogy via vector offsets

Subtract vectors from each other

$$a - b \approx c - d$$

$$man - woman \approx king - queen$$

$$d_w = \underset{d'_w \in V}{\text{argmax}}\, cos(a - b, c - d')$$

Linguistic relations : Hyponymy/Antonymy

- Hyponyms: a word of more specific meaning than a general term (X is a type-of Y)
  spoon - cutlery
  - best learnt in a supervised manner
- Antonyms: Usually occur in the same contexts
  - best learnt in a supervised manner

Learning a more compact space

- So far, our vectors have length V , the size of the vocabulary
  - dimensions corresponding to the words in the vocabulary (with vocabularies of 20,000 to 50,000)
- Do we really need this many dimensions?
- Can we represent words in a smaller dimensional space that preserves the similarity relationships of the larger space?
  - short (length 50 -1000)
  - dense (most elements are non-zero)
- Why short vectors?
  - easier to include as features in machine learning systems
  - contain fewer parameters, hence less prone to overfitting, generalise better.

Short Dense Vectors

- Vectors that are short (length 50 -1000)
- Dense (most elements are non-zero)
- Why short vectors?
  - easier to include as features in machine learning systems
  - contain fewer parameters, hence less prone to overfitting, generalise better.

Distributional semantic models

- Count-based models
  - Explicit vectors: dimensions are elements in the context
  - **long sparse** vectors with **interpretable** dimensions
  - Fast training
- Prediction-based models
  - Train a model to predict plausible contexts for a word
  - learn word representations in the process
  - **short dense** vectors with **latent** dimensions
  - Improved performance on other tasks

Neural Network inspired dense embeddings

- Neural network language models are trained to predict contexts from words or vice-versa - this process amounts to learning word embeddings
- The process for learning these embeddings has a strong relationship with dot-product similarity metrics
- We'll look at one particular version of this : Word2Vec

Prediction-based distributional models: Word2Vec

Word2Vec (Mikolov et al., 2013, Efficient Estimation of Word Representations in Vector Space)

- train a neural network to predict neighbouring words
- learn dense embeddings for the words in the training corpus in the process
- inspired by work on neural language models

- Consists of 2 models :
  - CBOW (continuous bag of words): predict a center word from the surrounding context
  - skipgram : predict the distribution (probability) of context words from a center word

Context window

If context size $= 2$:

center word

$$w_{t-2} \quad w_{t-1} \quad w_t \quad w_{t+1} \quad w_{t+2}$$

More generally:

center word

$$w_{t-m} \quad \ldots \quad w_{t-1} \quad w_t \quad w_{t+1} \quad \ldots \quad w_{t+m}$$

Word2vec: Overview

- We have a large corpus; every word in a fixed vocabulary is represented by a vector
- Go through each position t in the text, which has a center word w and context words c
- Use the similarity of the word vectors for w and c to calculate the probability of c given w (or vice versa), i.e $p(w|c)$ or $p(c|w)$
- Keep adjusting the word vectors to maximize this probability

Language Model

A Language Model is a model that assigns

- a probability to a sequence of words.

$$P(w_1, w_2, w_3, \ldots, w_n)$$

$$= P(w_1)\, P(w_2 \,|\, w_1)\, P(w_3 \,|\, w_1, w_2) \ldots P(w_n | w_1, \ldots w_{n-1})$$

- probability of an upcoming word given the previous words (context)

Language Model

- Bigram model: Assume that the probability of the sequence depends on the pairwise probability of a word in the sequence and the word next to it (here, before it).
  $P(w_1, w_2, w_3, \ldots, w_n) = \prod_{i=2}^{n} P(w_i|w_{i-1})$

- In a Word-Word Matrix of context size 1, we can learn these pair-wise probabilities (CBOW and Skipgram are models that learn these probabilities)

Skip-gram

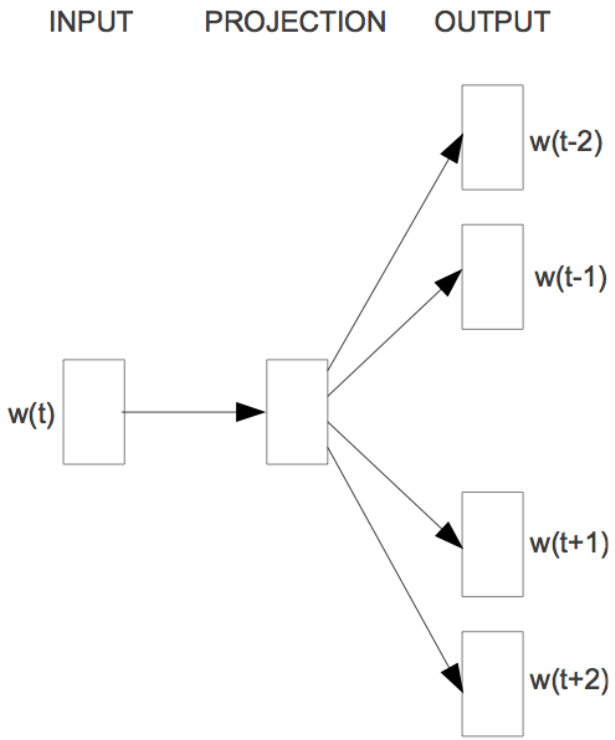
Given a word w_t :

- Predict each neighbouring word
  - $*$ in a context window of $2L$ words
- For $L = 2$, we predict 4 neighbouring words :

w_t

$$[w_{t-2},\ w_{t-1},\ w_{t+1},\ w_{t+2}]$$

Skip-gram



Slide credit: Tomas Mikolov

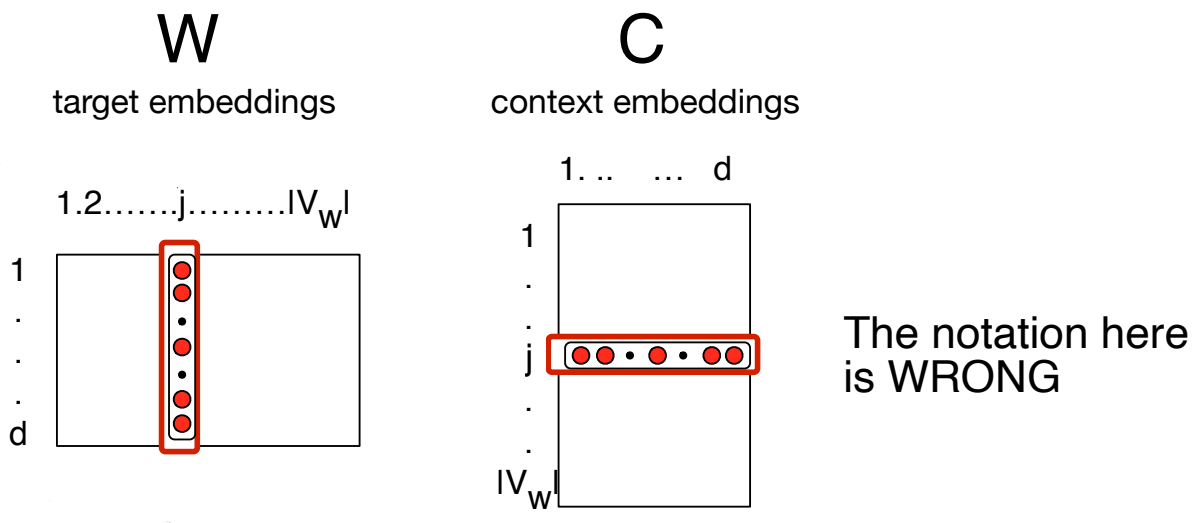
Two vectors (embeddings) per word

- We want to learn two vectors, for each word
  - when the word is in the context : **context** vector c (or output vector u)
  - when the word is in the center : **word** vector or **target** vector v (or input vector v)

Skip-gram: Parameter matrices

For each word $w_j \in V_w$:

- word vector v, in word matrix W :
- context vector c, in context matrix C



Slide credit: Dan Jurafsky

Notation (J&M): Skip gram

- w_j : word j from vocabulary V_w
- $\mathcal{W} \in \mathbb{R}^{n \times |V_w|}$: Input word matrix
- $\mathcal{C} \in \mathbb{R}^{|V_w| \times n}$: Output word matrix
- n is dimension of embedding
- v_j : j^{th} column of $\mathcal{W}$, i.e. target embedding of word w_j (input word representation)
- c_j : j^{th} row of $\mathcal{C}$, i.e. context embedding of word w_j (output word representation)

Skipgram- setup

- Walk through the corpus pointing at word $w(t)$, whose index in the vocabulary is j — we will call it w_j
- We want to predict $w(t+1)$, whose index in the vocabulary is k — we will call it w_k
- to do this, we need to compute $p(w_k \mid w_j)$
- Intuition behind skip gram : to compute this probability, we need to compute similarity between w_j and w_k

Use previous Count-based model to compete prediction on word

Skip-gram: Computing similarity

Similar vectors have a high dot product : $sim(c_k, v_j) \propto c_k \cdot v_j$

Similarity as dot-product between the target vector and the context vector

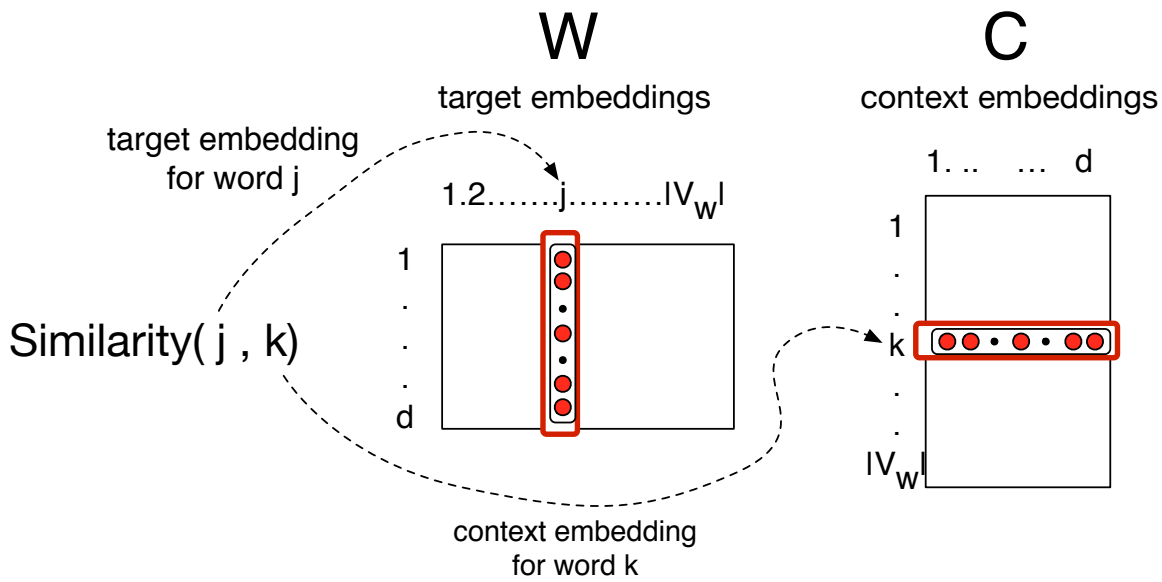


diagram: Dan Jurafsky

Skip-gram: Computing probability

- Dot product $c_k \cdot v_j$ is a number ranging from -inf. to +inf
- Use a **softmax** function to normalise the dot product into probabilities

$$p(w_k \mid w_j) = \frac{e^{c_k \cdot v_j}}{\sum_{c_i \in V} e^{c_i \cdot v_j}}$$

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

Skip-gram: Objective

Maximising the probability of the corpus (data likelihood)

- For each position $t = 1, \ldots T$, predict context words within a window

$$\text{Likelihood} = L(\theta) = \prod_{t=1}^{T} \prod_{m \leq j \leq m} P(w_{t+j} \mid w_t; \theta)$$

- Objective function $J(\theta)$ is the average negative log likelihood

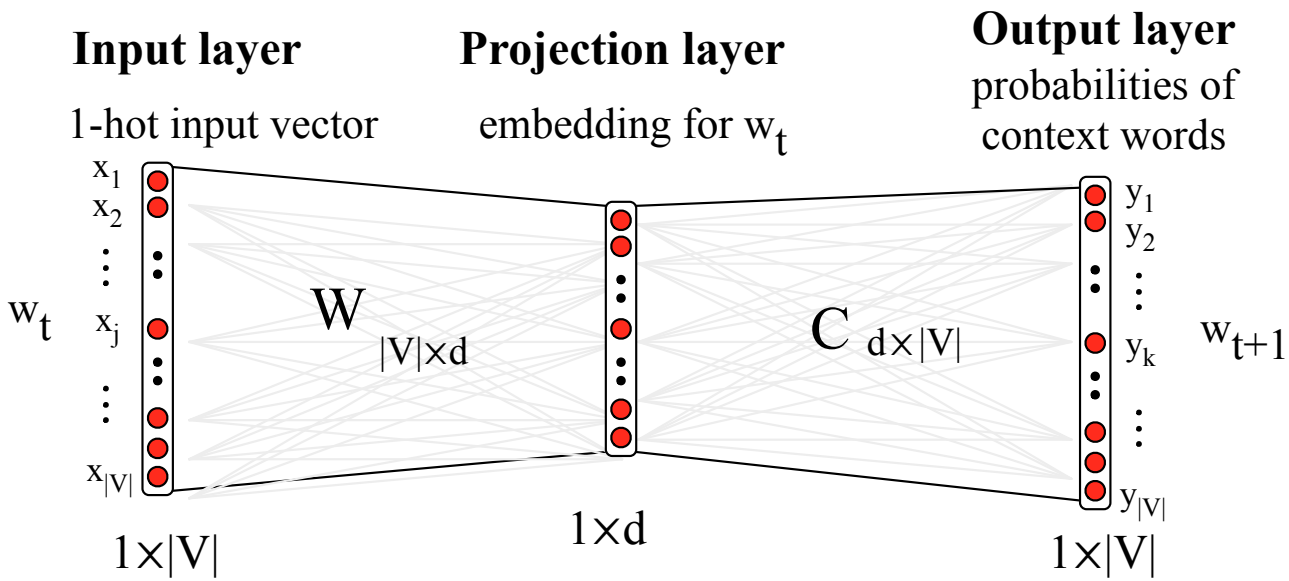
$$J(\theta) = -\frac{1}{T} \log L(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{m \leq j \leq m} \log P(w_{t+j} \mid w_t; \theta)$$

How to calculate $P(w_{t+j} \mid w_t; \theta)$

- Use two vectors per word: v_j is the word vector; c_k is the context vector

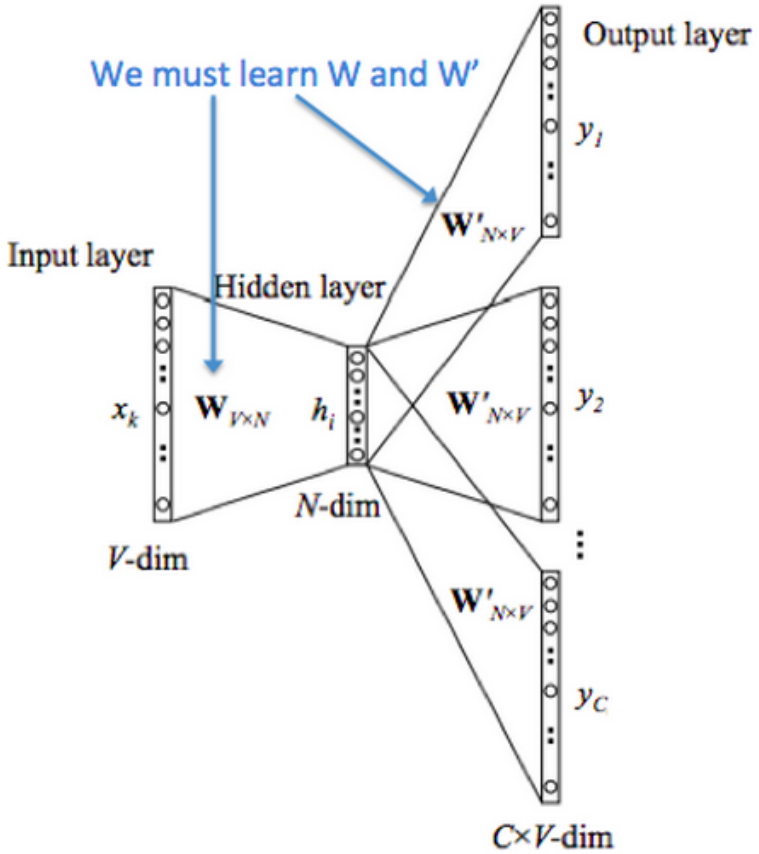
$$p(w_k \mid w_j) = \frac{e^{c_k \cdot v_j}}{\sum_{c_i \in V} e^{c_i \cdot v_j}}$$

Visualising skip-gram as a network

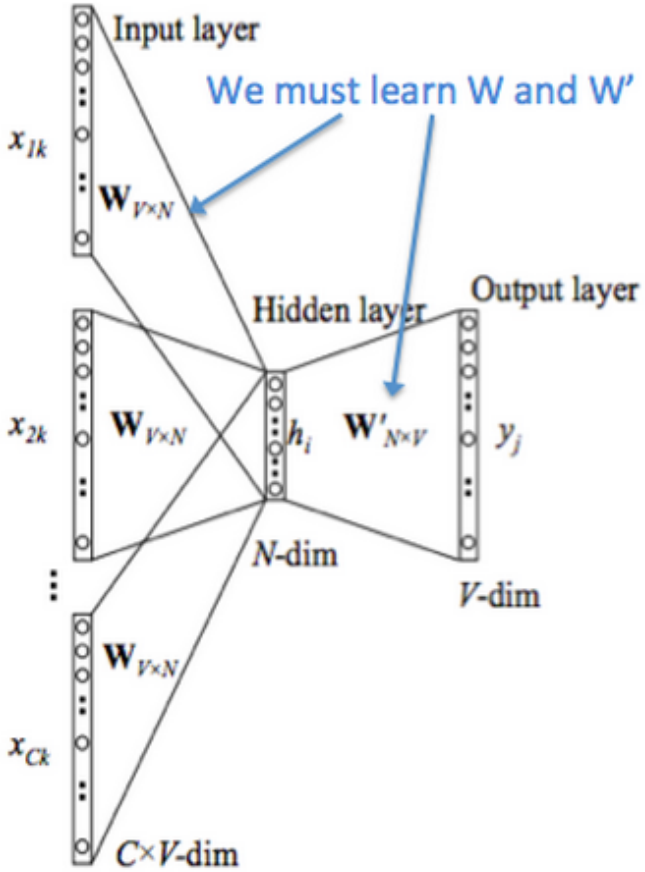


Slide credit: Dan Jurafsky

Network schematic: Skip-gram



Network schematic: CBOW



Skip-gram with negative sampling

- Problem with softmax : expensive to compute the denominator for the whole vocabulary

$$p(w_k \mid w_j) = \frac{e^{c_k \cdot v_j}}{\sum_{c_i \in V} e^{c_i \cdot v_j}}$$

- Solution: approximate the denominator by **negative sampling**
  - at training time, walk through the corpus
  - for each target word and positive context
  - sample k noise samples (negative samples), i.e. other words
  - Goal: to move the embeddings towards the positive context words, and away from the noise words

Skip-gram with negative sampling

Objective in training:

- Make the word like the context words
  lemon, a [tablespoon of apricot preserves or] jam.

  c_1 c_2 w c_3 c_4
- And not like the k negative examples

[cement idle dear coaxial apricot attendant whence forever puddle]

n_1 n_2 n_3 n_4 w n_5 n_6 n_7 n_8

Skip-gram with negative sampling: training data

Convert the dataset into word pairs:

- **Positive (+)**

  (apricot, tablespoon)
  (apricot, of)
  (apricot, jam)
  (apricot, or)

- **Negative (-)**

  (apricot, cement)
  (apricot, idle)
  (apricot, attendant)
  (apricot, dear)
  ...

Skip-gram with negative sampling

- instead of treating it as a **multi-class problem** (and returning a probability distribution over the whole vocabulary)
- **return a probability** that word w_k is a valid context for word w_j

$$P(+|w_j, w_k)$$

$$P(-|w_j, w_k) = 1 - P(+|w_j, w_k)$$

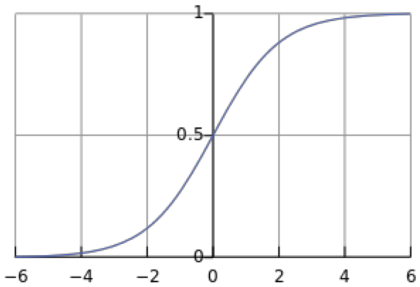
Skip-gram with negative sampling

- model similarity as a dot product

$$sim(c_k, v_j) \propto c_k \cdot v_j$$

- turn this into a probability using the **sigmoid function**

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



$$P(+|w_j, w_k) = \frac{1}{1 + e^{-c_k \cdot v_j}}$$

$$P(-|w_j, w_k) = 1 - P(+|w_j, w_k) = 1 - \frac{1}{1 + e^{-c_k \cdot v_j}} = \frac{1}{1 + e^{c_k \cdot v_j}}$$

Skip-gram with negative sampling

New objective function:

- maximise the probability of a word and context being in the corpus data if it indeed is
- maximise the probability of a word and context *not* being in the corpus data if it indeed is not

$$\theta = \underset{\theta}{\text{argmax}} \prod_{(w,c)\in D} P(D=1|w,c,\theta) \prod_{(w,c)\in \tilde{D}} P(D=0|w,c,\theta)$$

- Maximising the likelihood $=$ minimising the negative log likelihood

$$J = -\sum_{(w,c)\in D} \log \frac{1}{1+\exp(-u_w^T v_c)} - \sum_{(w,c)\in \tilde{D}} \log\left(\frac{1}{1+\exp(u_w^T v_c)}\right)$$

Skip-gram with negative sampling: new objective function

$$
\begin{aligned}
\theta &= \underset{\theta}{\text{argmax}} \prod_{(w,c)\in D} P(D=1|w,c,\theta) \prod_{(w,c)\in \tilde{D}} P(D=0|w,c,\theta) \\
&= \underset{\theta}{\text{argmax}} \prod_{(w,c)\in D} P(D=1|w,c,\theta) \prod_{(w,c)\in \tilde{D}} (1-P(D=1|w,c,\theta)) \\
&= \underset{\theta}{\text{argmax}} \sum_{(w,c)\in D} \log P(D=1|w,c,\theta) + \sum_{(w,c)\in \tilde{D}} \log(1-P(D=1|w,c,\theta)) \\
&= \underset{\theta}{\text{argmax}} \sum_{(w,c)\in D} \log \frac{1}{1+\exp(-u_w^T v_c)} + \sum_{(w,c)\in \tilde{D}} \log(1-\frac{1}{1+\exp(-u_w^T v_c)}) \\
&= \underset{\theta}{\text{argmax}} \sum_{(w,c)\in D} \log \frac{1}{1+\exp(-u_w^T v_c)} + \sum_{(w,c)\in \tilde{D}} \log(\frac{1}{1+\exp(u_w^T v_c)})
\end{aligned}
$$

Word embeddings in practise

Word2vec is often used for pretraining in other tasks.

- It will help your models start from an **informed** position
- Requires only **plain text** - which we have a lot of
- Is very **fast** and easy to use
- Already **pretrained** vectors also available (trained on 100B words)

However, for best performance it is important to continue training, fine-tuning the embeddings for a specific task.

GloVe: Global Vectors for Word Representation

[Pennington, Socher, and Manning, 2014]

Two main classes of methods for word embeddings

- Count-based + matrix factorization (e.g. SVD) to get dense embeddings
  - capture global statistical information
  - capture word similarity, but poor on tasks like word-analogy
- Shallow window-based, learn by making predictions in local context windows
  - capture complex linguistic patterns beyond similarity
  - fail to make use of global statistics

GloVe: Trains on global word co-occurrence counts

Encoding meaning in vector differences [Pennington, Socher, and Manning, 2014]

- Very fast training
- Scalable to huge corpora, but good performance even with small corpus and small vectors
- State-of-the-art results on many semantic tasks (esp. analogy)

Encoding meaning in vector differences [Pennington, Socher, and Manning, 2014]

Crucial insight : Ratios of co-occurrence probabilities can encode meaning components

	x = solid	x = gas	x = water	x = random
$P(x\|\text{ice})$	large	small	large	small
$P(x\|\text{steam})$	small	large	large	small
$\frac{P(x\|\text{ice})}{P(x\|\text{steam})}$	large	small	~1	~1

Encoding meaning in vector differences [Pennington, Socher, and Manning, 2014]

Crucial insight : Ratios of co-occurrence probabilities can encode meaning components

	x = solid	x = gas	x = water	x = random
$P(x\|\text{ice})$	large	small	large	small
$P(x\|\text{steam})$	small	large	large	small
$\frac{P(x\|\text{ice})}{P(x\|\text{steam})}$	large	small	~1	~1

Ratio is better able to distinguish relevant words from irrelevant ones, as compared to raw probabilities!

Encoding meaning in vector differences [Pennington, Socher, and Manning, 2014]

Crucial insight : Ratios of co-occurrence probabilities can encode meaning components

	x = solid	x = gas	x = water	x = fashion
$P(x\|\text{ice})$	1.9 x 10^{-4}	6.6 x 10^{-5}	3.0 x 10^{-3}	1.7 x 10^{-5}
$P(x\|\text{steam})$	2.2 x 10^{-5}	7.8 x 10^{-4}	2.2 x 10^{-3}	1.8 x 10^{-5}
$\frac{P(x\|\text{ice})}{P(x\|\text{steam})}$	8.9	8.5 x 10^{-2}	1.36	0.96

Co-occurrence matrix

- X : Co-occurrence matrix
- X_{ij} : the number of times word j occurs in the context of word i

Co-occurrence matrix

- X : Co-occurrence matrix
- X_{ij} : the number of times word j occurs in the context of word i
- X_i: number of times any word k appears in the context of word i

$$X_i = \sum_k X_{ik}$$

Co-occurrence matrix

- X : Co-occurrence matrix
- X_{ij} : the number of times word j occurs in the context of word i
- X_i: number of times any word k appears in the context of word i

$$X_i = \sum_k X_{ik}$$

- Probability of j appearing in the context of i

$$P_{ij} = P(w_j \mid w_i) = \frac{X_{ij}}{X_i}$$

- Populating this matrix needs a single pass through the corpus to collect the statistics: can be large, but is a one-time cost.

Encoding meaning in vector differences

How can we capture ratios of co-occurrence probabilities as linear meaning components in a word vector space?

A: Log-bilinear model: $w_i \cdot w_j = \log P(i|j)$

with vector differences $w_x \cdot (w_a - w_b) = \log \frac{P(x|a)}{P(x|b)}$

GloVe

- Fast training
- Scalable to huge corpora
- Good performance even with small corpus and small vectors
- State-of-the-art results on similarity, analogy and other relations.

Glove Results

Nearest words to frog:

1. frogs
2. toad
3. litoria
4. leptodactylidae
5. rana
6. lizard
7. eleutherodactylus

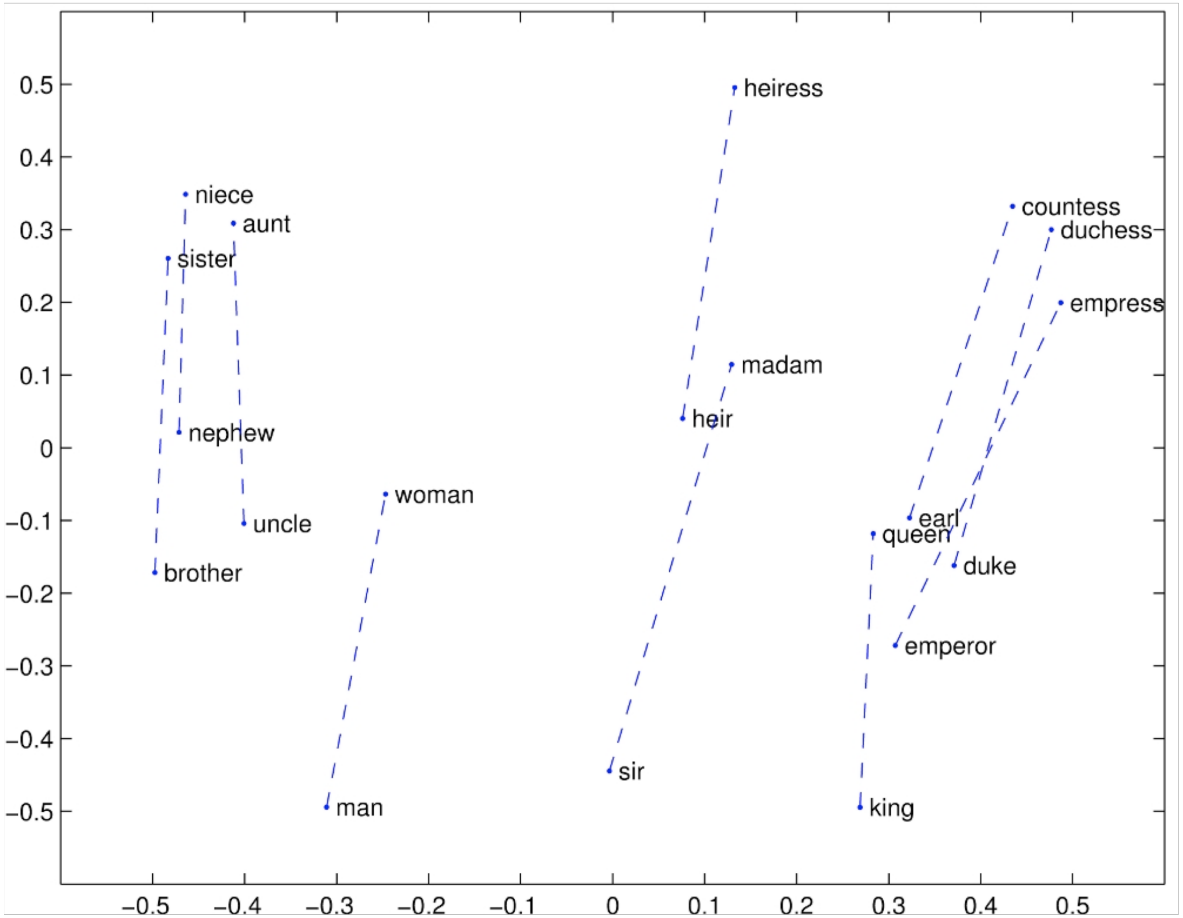
litoria

leptodactylidae

rana

eleutherodactylus

Glove Visualizations



Glove Visualizations : Superlatives