Distributional Lexical Semantics (and word2vec)

Yoav Goldberg
yogo@cs.biu.ac.il

Bar Ilan University

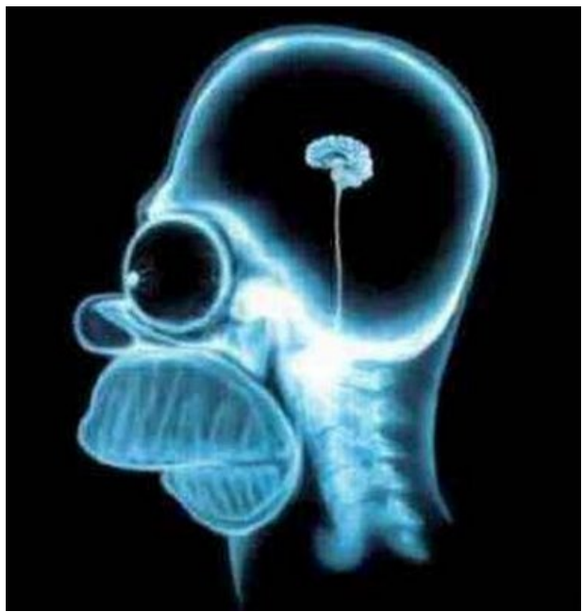
with some slides by
Ido Dagan, Omer Levy, Marco Baroni, Kathrin Erk

A lot of text.

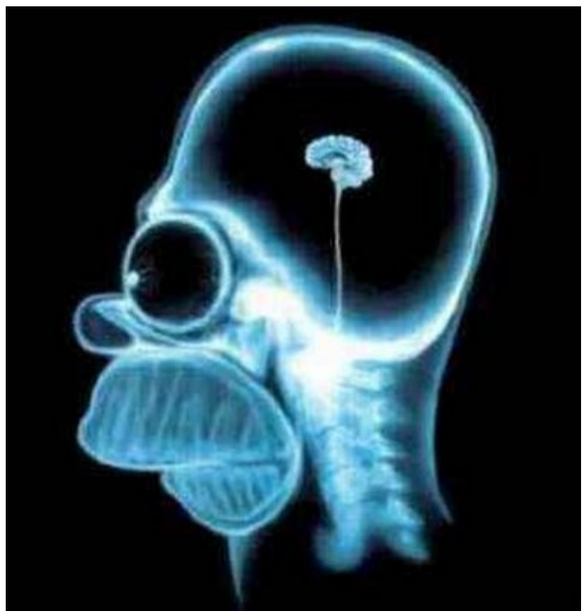
Need to understand what's being said.

this is where we come in.

NLP

text →  → meaning

NLP

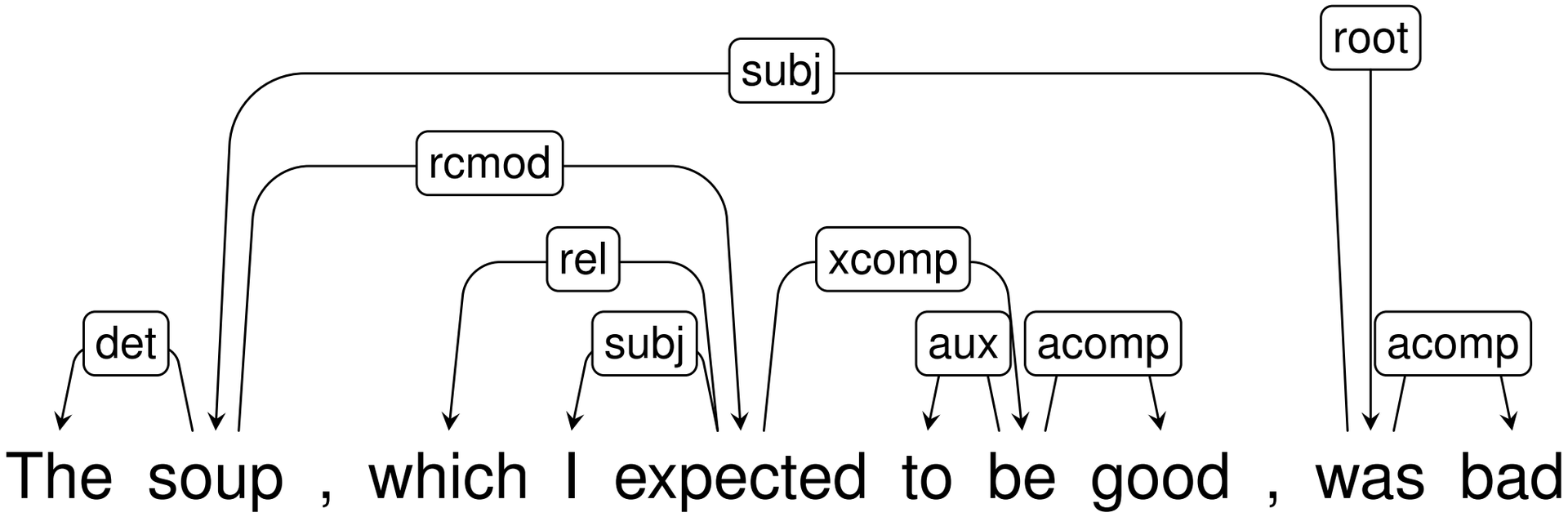
text →  → meaning

What does it mean to **understand**?

Understanding the Structure

The soup , which I expected to be good , was bad

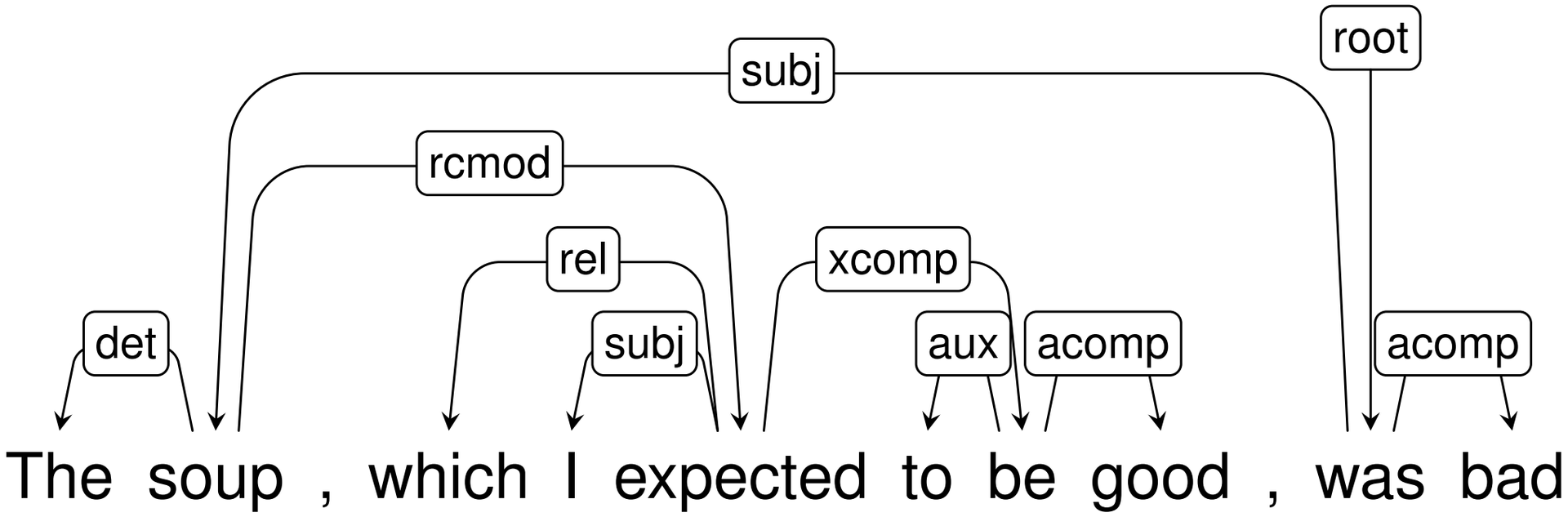
Understanding the Structure



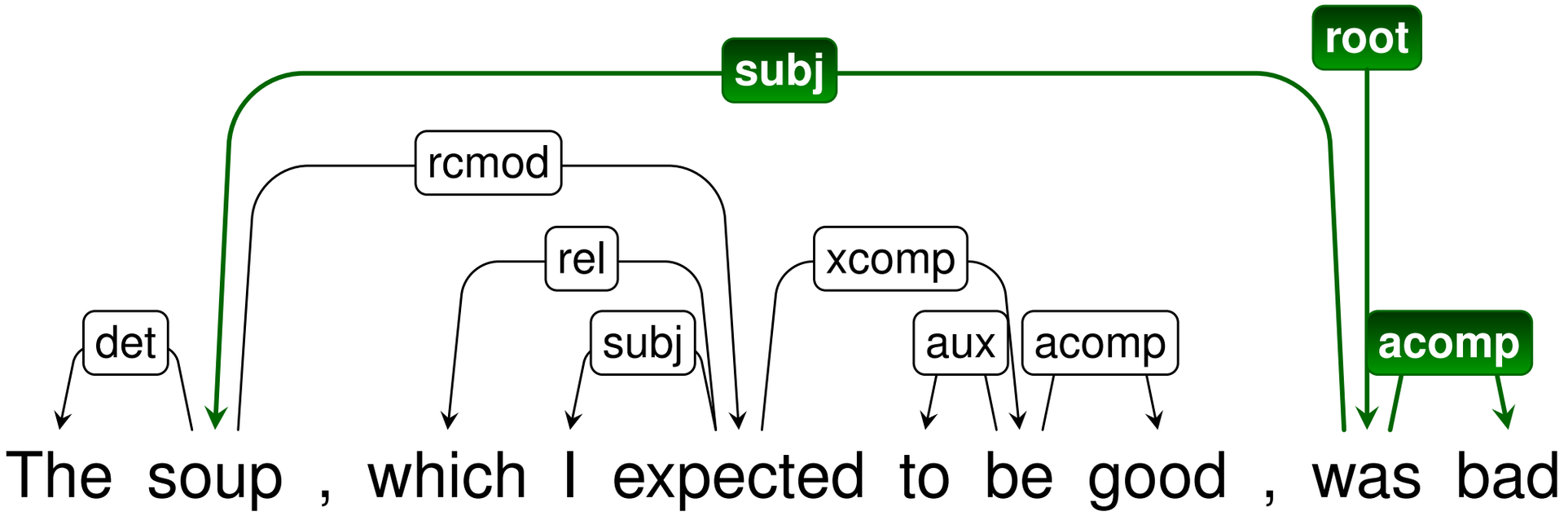
Understanding the Structure

This is called **Syntactic Parsing**.

Understanding the Structure



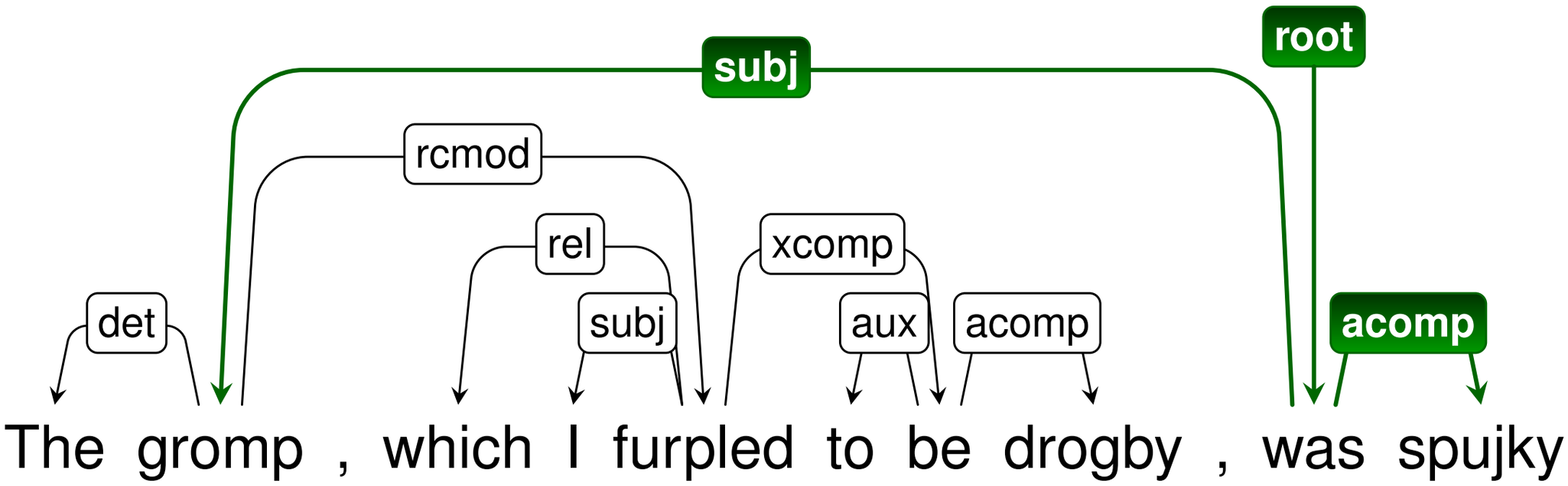
Understanding the Structure



Understanding the Structure

The gromp , which I furpled to be drogby , was spujky

Understanding the Structure



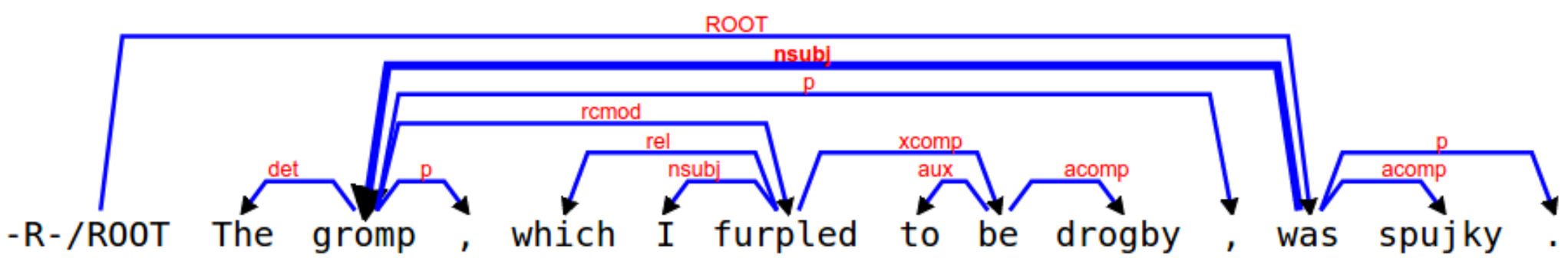
u.cs.biu.ac.il/~yogo/p ×

u.cs.biu.ac.il/~yogo/parse/index?text=The+gromp%2C+which+I+furpled+to+be+drogby%2C+was+spuj

Greedy parsing to Stanford Dependencies

The gromp, which I furpled to be drogby, was spujky.

Parse



Understanding the Structure

Can understand **structure** without understanding **words**.

But the words are also important.

This is not a lecture about parsing.

Today we will focus on the words.

Understanding the Words

soup was bad

Understanding the Words

soup was bad
soup was awful

Understanding the Words

soup was bad
soup was awful
soup was lousy

Understanding the Words

soup was bad
soup was awful
soup was lousy
soup was abysmal

Understanding the Words

soup was bad
soup was awful
soup was lousy
soup was abysmal
soup was icky

Understanding the Words

soup was bad
soup was awful
soup was lousy
soup was abysmal
soup was icky

chowder was nasty

Understanding the Words

soup was bad
soup was awful
soup was lousy
soup was abysmal
soup was icky

chowder was nasty
pudding was terrible

Understanding the Words

soup was bad
soup was awful
soup was lousy
soup was abysmal
soup was icky

chowder was nasty
pudding was terrible
cake was bad

Understanding the Words

soup was bad
soup was awful
soup was lousy
soup was abysmal
soup was icky

chowder was nasty
pudding was terrible
cake was bad
hamburger was lousy

Understanding the Words

soup was bad
soup was awful
soup was lousy
soup was abysmal
soup was icky

chowder was nasty
pudding was terrible
cake was bad
hamburger was lousy

service was poor

Understanding the Words

soup was bad
soup was awful
soup was lousy
soup was abysmal
soup was icky

chowder was nasty
pudding was terrible
cake was bad
hamburger was lousy

service was poor
atmosphere was shoddy

Understanding the Words

soup was bad
soup was awful
soup was lousy
soup was abysmal
soup was icky

chowder was nasty
pudding was terrible
cake was bad
hamburger was lousy

service was poor
atmosphere was shoddy
hammer was heavy

Understanding the Words

soup was bad
soup was awful
soup was lousy
soup was abysmal
soup was icky

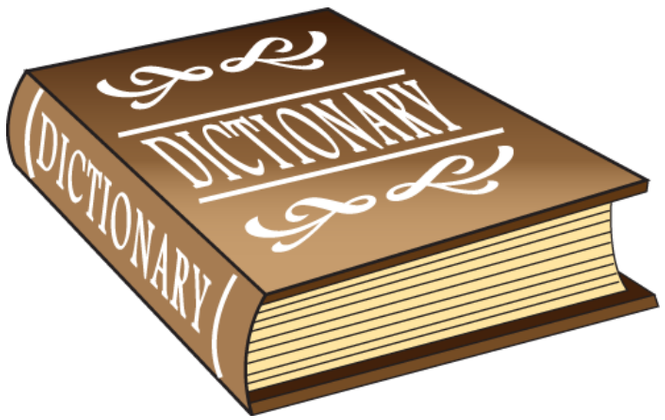
chowder was nasty
pudding was terrible
cake was bad
hamburger was lousy

service was poor
atmosphere was shoddy
hammer was heavy

- To the computer, each word is just a symbol, so these are all the same.
- But to us, some are more similar than others.
- We'd like a word representation that can capture that.

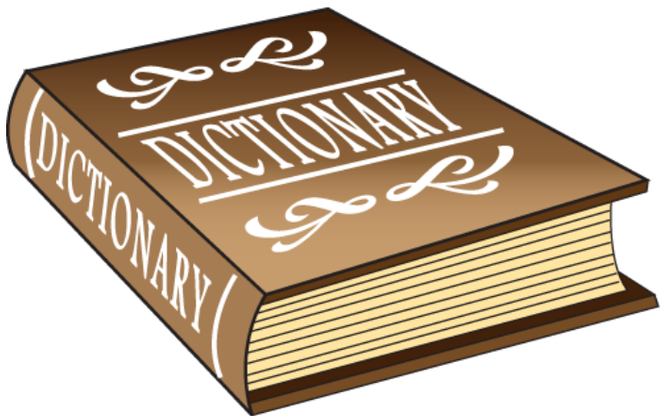
Representing Words

Use a dictionary?



Representing Words

Use a dictionary?



Doesn’t scale.

Representing Words

The distributional Hypothesis

Dr. Baroni saw a hairy little wampinuck sleeping behind a tree

Representing Words

The distributional Hypothesis

Dr. Baroni saw a hairy little wampinuck sleeping behind a tree

The Distributional Hypothesis – Haris, 1954

Words in similar contexts tend to have similar meanings

Firth, 1957

"*you should know a word by the company it keeps*"

Distributional Representation of Word Meaning and Similarity

The distributional hypothesis in real life

McDonald & Ramscar 2001

He filled the wampimuk, passed it around and we all drunk some

We found a little, hairy wampimuk sleeping behind the tree

What word can appear in the context of all these words?

Word 1: drown, bathroom, shower, fill, fall, lie, electrocute, toilet, whirlpool, iron, gin

Word 2: eat, fall, pick, slice, peel, tree, throw, fruit, pie, bite, crab, grate

Word 3: advocate, overthrow, establish, citizen, ideal, representative, dictatorship, campaign, bastion, freedom

Word 4: spend, enjoy, remember, last, pass, end, die, happen, brighten, relive

What word can appear in the context of all these words?

bathtub

Word 1: drown, bathroom, shower, fill, fall, lie, electrocute, toilet, whirlpool, iron, gin

apple

Word 2: eat, fall, pick, slice, peel, tree, throw, fruit, pie, bite, crab, grate

democracy

Word 3: advocate, overthrow, establish, citizen, ideal, representative, dictatorship, campaign, bastion, freedom

day

Word 4: spend, enjoy, remember, last, pass, end, die, happen, brighten, relive

What can you say about word number 5?
Distributional Similarity (2nd-order)

bathtub

Word 1: drown, bathroom, shower, fill, fall, lie, electrocute, toilet, whirlpool, iron, gin

apple

Word 2: eat, fall, ripe, slice, peel, tree, throw, fruit, pie, bite, crab, grate

day

Word 4: spend, enjoy, remember, last, pass, end, die, happen, brighten, relive

democracy

Word 3: advocate, overthrow, establish, citizen, ideal, representative, dictatorship, campaign, bastion, freedom

Word 5: eat, paint, peel, apple, fruit, juice, lemon, blue, grow

What can you say about word number 5?

Distributional Similarity (2nd-order)

bathtub

Word 1: drown, bathroom, shower, fill, fall, lie, electrocute, toilet, whirlpool, iron, gin

apple

Word 2: eat, fall, ripe, slice, peel, tree, throw, fruit, pie, bite, crab, grate

day

Word 4: spend, enjoy, remember, last, pass, end, die, happen, brighten, relive

democracy

Word 3: advocate, overthrow, establish, citizen, ideal, representative, dictatorship, campaign, bastion, freedom

orange

Word 5: eat, paint, peel, apple, fruit, juice, lemon, blue, grow

Counting context words

- They **picked up red** apples **that had fallen** to the ground
- **Eating** apples **is healthy**
- She **ate a red** apple
- **Pick an** apple

Word count, 3-word

a	be	eat	fall	have	healthy	pick	red	that	up
2	1	2	1	1	1	2	2	1	1

Distributional semantics

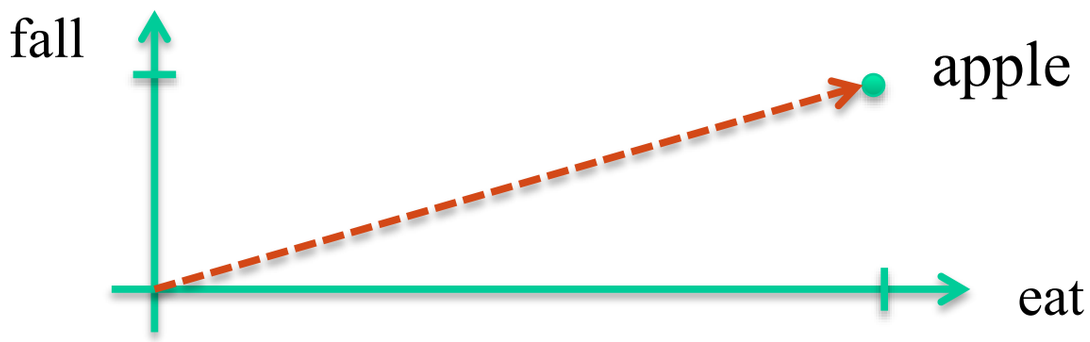
- Comparing two words:
  - Look at all context words for word1
  - Look at all context words for word2
    - How similar are those two context collections in their entirety?
- Compare distributional representations of two words

How can we compare two context collections in their entirety?

Count how often “apple” occurs close to other words in a large text collection (corpus):

eat	fall	ripe	slice	peel	tree	throw	fruit	pie	bite	crab
794	244	47	221	208	160	145	156	109	104	88

Interpret counts as coordinates:



Every context word becomes a dimension.

How can we compare two context collections in their entirety?

Count how often “apple” occurs close to other words in a large text collection (corpus):

eat	fall	ripe	slice	peel	tree	throw	fruit	pie	bite	crab
794	244	47	221	208	160	145	156	109	104	88

Do the same for “orange”:

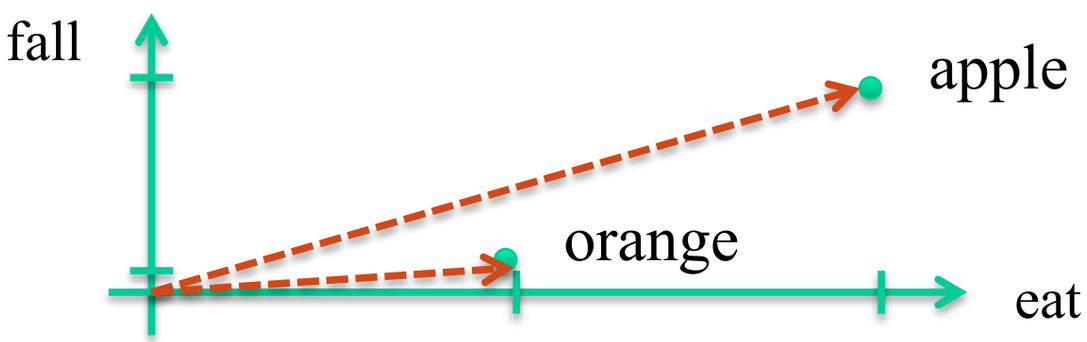
eat	fall	ripe	slice	peel	tree	throw	fruit	pie	bite	crab
265	22	25	62	220	64	74	111	4	4	8

How can we compare two context collections in their entirety?

Then visualize both count tables as vectors in the same space:

eat	fall	ripe	slice	peel	tree	throw	fruit	pie	bite	crab
794	244	47	221	208	160	145	156	109	104	88

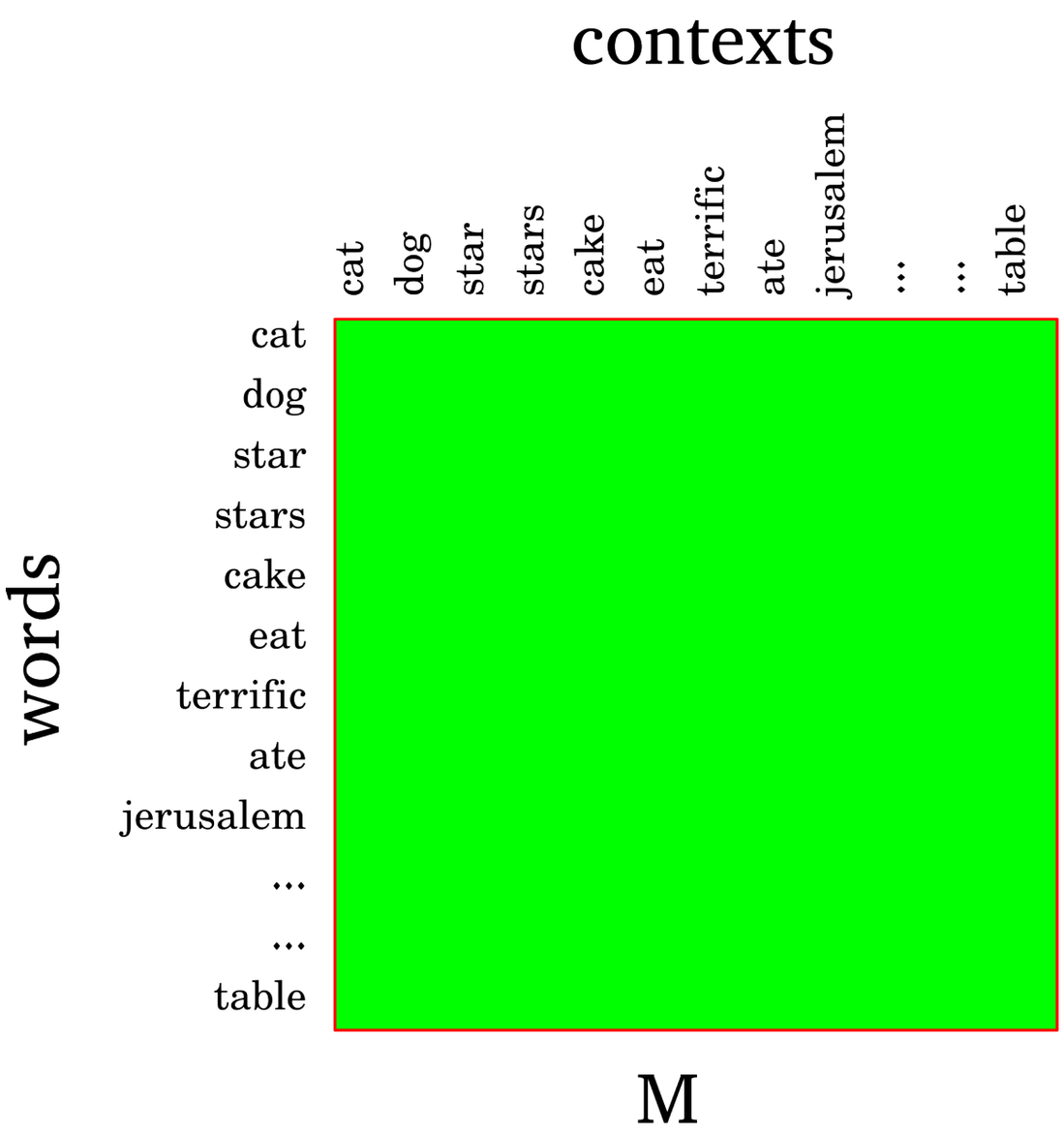
eat	fall	ripe	slice	peel	tree	throw	fruit	pie	bite	crab
265	22	25	62	220	64	74	111	4	4	8



Similarity between two words as proximity in space

Words as Vectors

We can arrange the words in a huge, sparse matrix, where each row is a word, and each column is a context.



now, each word is associated with a (sparse) vector of counts.

Using distributional models

- Finding (near-)synonyms: automatically building a thesaurus
- Related: use distributional similarity of documents (containing similar words) in Information Retrieval

Where can we find texts to use for making a distributional model?

- Text in electronic form!
- Newspaper articles
- Project Gutenberg: older books available for free
- Wikipedia
- Text collections prepared for language analysis:
  - Balanced corpora
  - WaC: Scrape all web pages in a particular domain
    - uk, fr, it, de (http://wacky.sslmit.unibo.it/doku.php?id=corpora)
  - ELRA, LDC hold corpus collections
    - For example, large amounts of newswire reports
  - Google n-grams, Google books

How much text do we need?

- At least:
  British National Corpus, 100 million words
- Better: add
  – UKWaC (2 billion words)
  – Wikipedia (2 billion words)

How much text do we need?

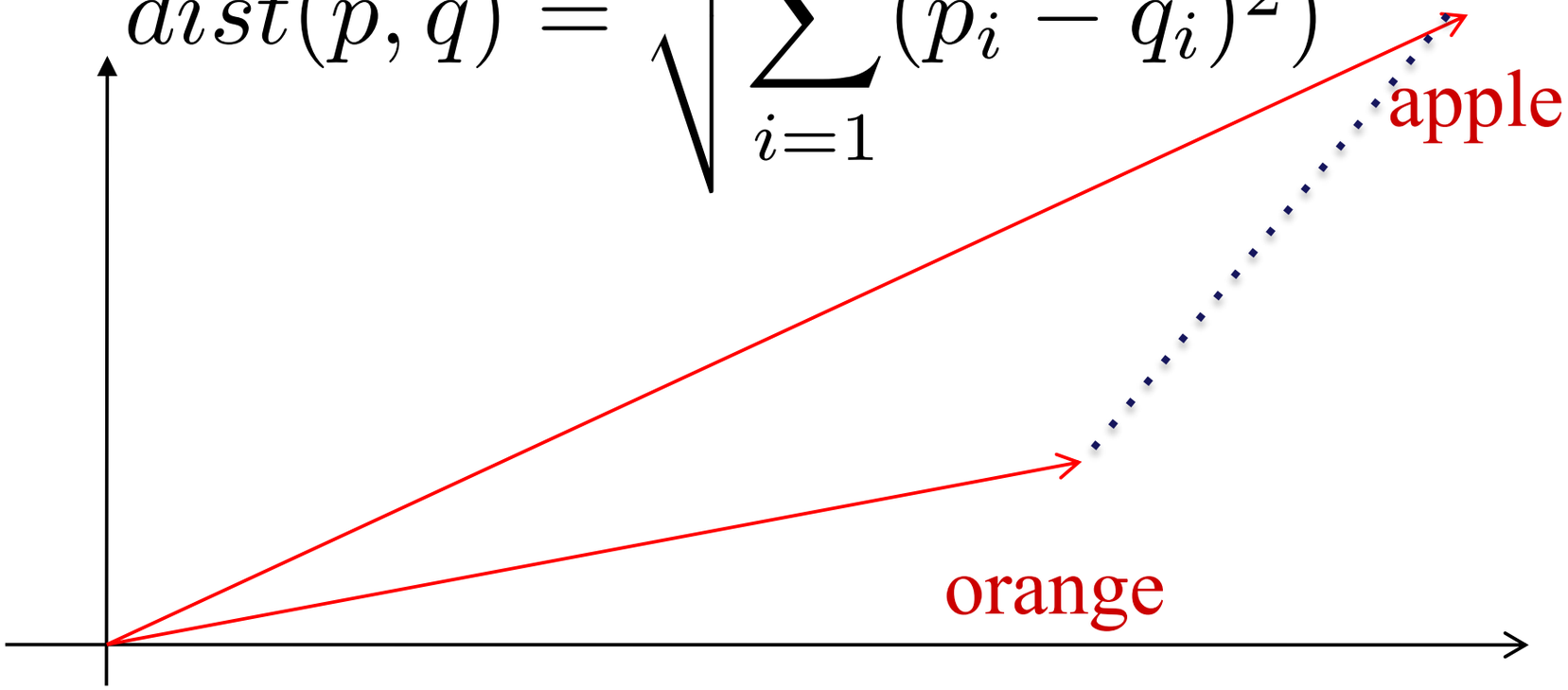
- At least:
  British National Corpus, 100 million words
- Better: add
  – UKWaC (2 billion words)
  – Wikipedia (2 billion words)

  Or go even larger with common-crawl (terrabytes)

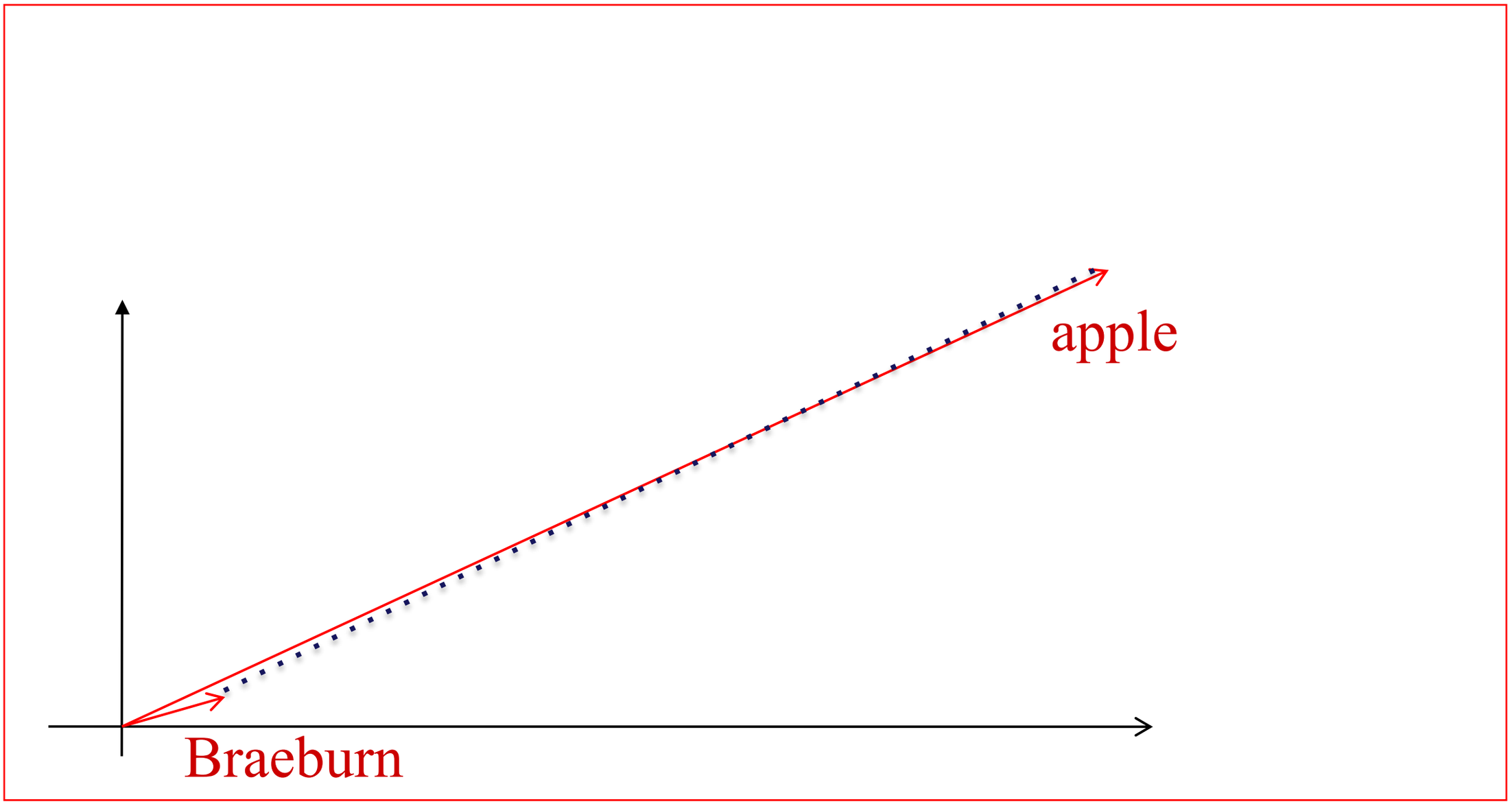
What do we mean by "similarity" of vectors?

Euclidean distance (a dissimilarity measure!):

$$dist(\vec{p}, \vec{q}) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2)}$$



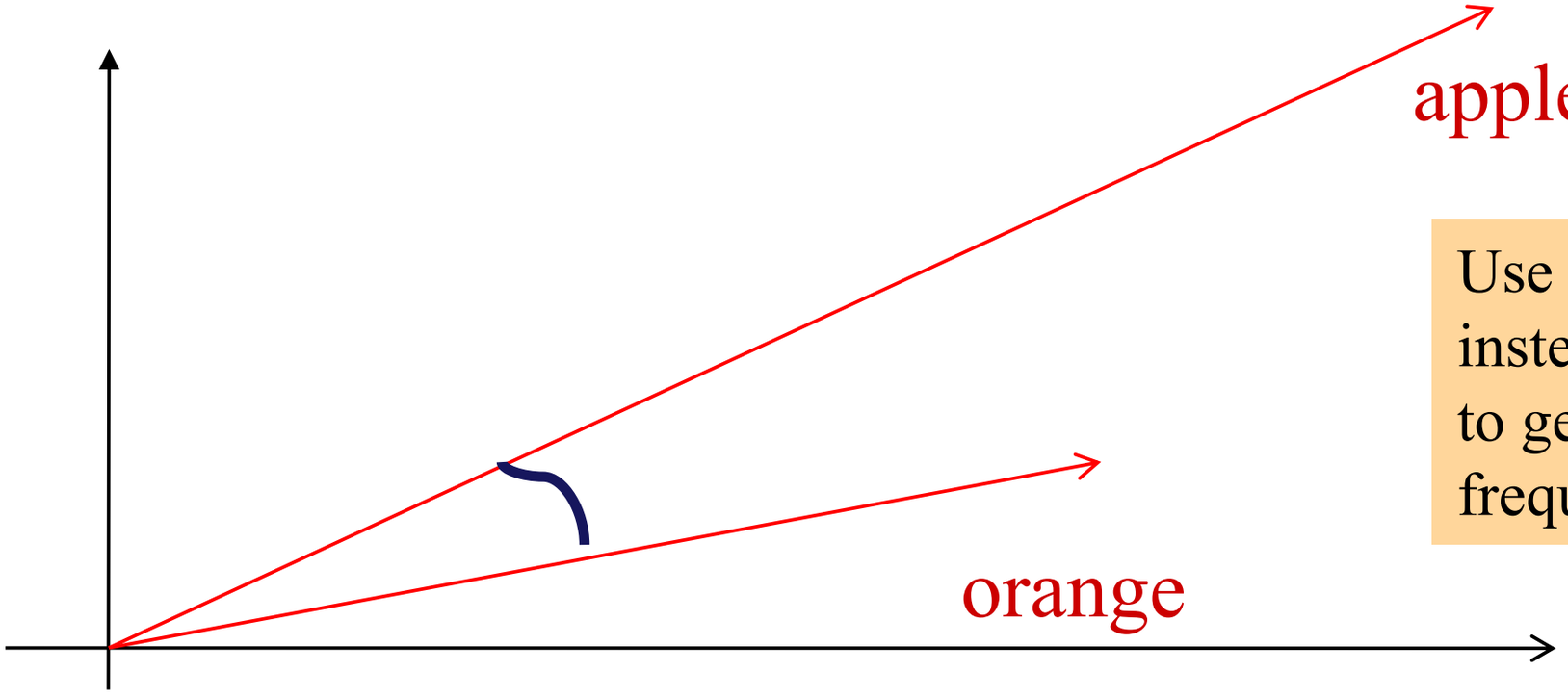
Problem with Euclidean distance: very sensitive to word frequency!



What do we mean by "similarity" of vectors?

Cosine similarity:

$$cos(\vec{p}, \vec{q}) = \frac{\sum_{i=1}^{n} p_i \cdot q_i}{\sqrt{\sum_{i=1}^{n} p_i^2} \cdot \sqrt{\sum_{i=1}^{n} q_i^2}}$$



Use angle between vectors instead of point distance to get around word frequency issues

Some counts for “letter” in “Pride and Prejudice”. What do you notice?

the	to	of	and	a	her	she	his	is	was	in	that
102	75	72	56	52	50	41	36	35	34	34	33

had	i	from	you	as	this	mr	for	not	on	be	he
32	28	28	25	23	23	22	21	21	20	18	17

but	elizabeth	with	him	which	by	when	jane
17	17	16	16	16	15	14	12

Some counts for “letter” in “Pride and Prejudice”. What do you notice?

the	to	of	and	a	her	she	his	is	was	in	that
102	75	72	56	52	50	41	36	35	34	34	33

had	i	from	you	as	this	mr	for	not	on	be	he
32	28	28	25	23	23	22	21	21	20	18	17

but	elizabeth	with	him	which	by	when	jane
17	17	16	16	16	15	14	12

Some counts for “letter” in “Pride and Prejudice”. What do you notice?

the	to	of	and	a	her	she	his	is	was	in	that
102	75	72	56	52	50	41	36	35	34	34	33

had	i	from	you	as	this	mr	for	not	on	be	he
32	28	28	25	23	23	22	21	21	20	18	17

but	elizabeth	with	him	which	by	when	jane
17	17	16	16	16	15	14	12

All the most frequent co-occurring words are function words.

Some words are more informative than others

- Function words co-occur frequently with <u>all</u> words
  - That makes them less informative
- They have much higher co-occurrence counts than content words
  - They can "drown out" more informative contexts

Using association rather than co-occurrence counts

- Degree of association between target and context:
  - High association: high co-occurrence with "letter", lower with everything else
  - Low association: lots of co-occurrence with all words
- Many ways of implementing this
- For example Pointwise Mutual Information between target a and context b:

$$PMI(a,b) = \log \frac{P(a,b)}{P(a) \cdot P(b)}$$

PMI

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

$$P(w,c) = \frac{\#(w,c)}{\sum_{w,c \in D} \#(w,c)}$$

$$P(c) = \frac{\#(\square,c)}{\sum_{w,c \in D} \#(w,c)}$$

$$P(w) = \frac{\#(w,\square)}{\sum_{w,c \in D} \#(w,c)}$$

PMI

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

matrix entry

$$P(w,c) = \frac{\#(w,c)}{\sum_{w,c \in D} \#(w,c)}$$

$$P(c) = \frac{\#(\square,c)}{\sum_{w,c \in D} \#(w,c)}$$

$$P(w) = \frac{\#(w,\square)}{\sum_{w,c \in D} \#(w,c)}$$

PMI

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

matrix entry

sum of matrix entries

$$P(w,c) = \frac{\#(w,c)}{\sum_{w,c \in D} \#(w,c)}$$

$$P(c) = \frac{\#(\square,c)}{\sum_{w,c \in D} \#(w,c)}$$

$$P(w) = \frac{\#(w,\square)}{\sum_{w,c \in D} \#(w,c)}$$

PMI

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

matrix entry

sum of matrix entries

$$P(w,c) = \frac{\#(w,c)}{\sum_{w,c \in D} \#(w,c)}$$

sum of matrix column

$$P(c) = \frac{\#(\square,c)}{\sum_{w,c \in D} \#(w,c)}$$

$$P(w) = \frac{\#(w,\square)}{\sum_{w,c \in D} \#(w,c)}$$

PMI

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

matrix entry

sum of matrix entries

$$P(w,c) = \frac{\#(w,c)}{\sum_{w,c \in D} \#(w,c)}$$

sum of matrix column

$$P(c) = \frac{\#(\square,c)}{\sum_{w,c \in D} \#(w,c)}$$

sum of matrix row

$$P(w) = \frac{\#(w,\square)}{\sum_{w,c \in D} \#(w,c)}$$

PMI

We often move from PMI to PPMI (Positive PMI)

$$PPMI(a,b) = max(PMI(a,b), 0)$$

Weighting

Adjusting raw co-occurrence counts:

	bright	in		
stars	385	10788	...	← Counts

Weighting

Adjusting raw co-occurrence counts:

	bright	in		
stars	385	10788	...	← Counts
stars	43.6	5.3	...	← PPMI

Weighting

Adjusting raw co-occurrence counts:

	bright	in		
stars	385	10788	...	← Counts
stars	43.6	5.3	...	← PPMI

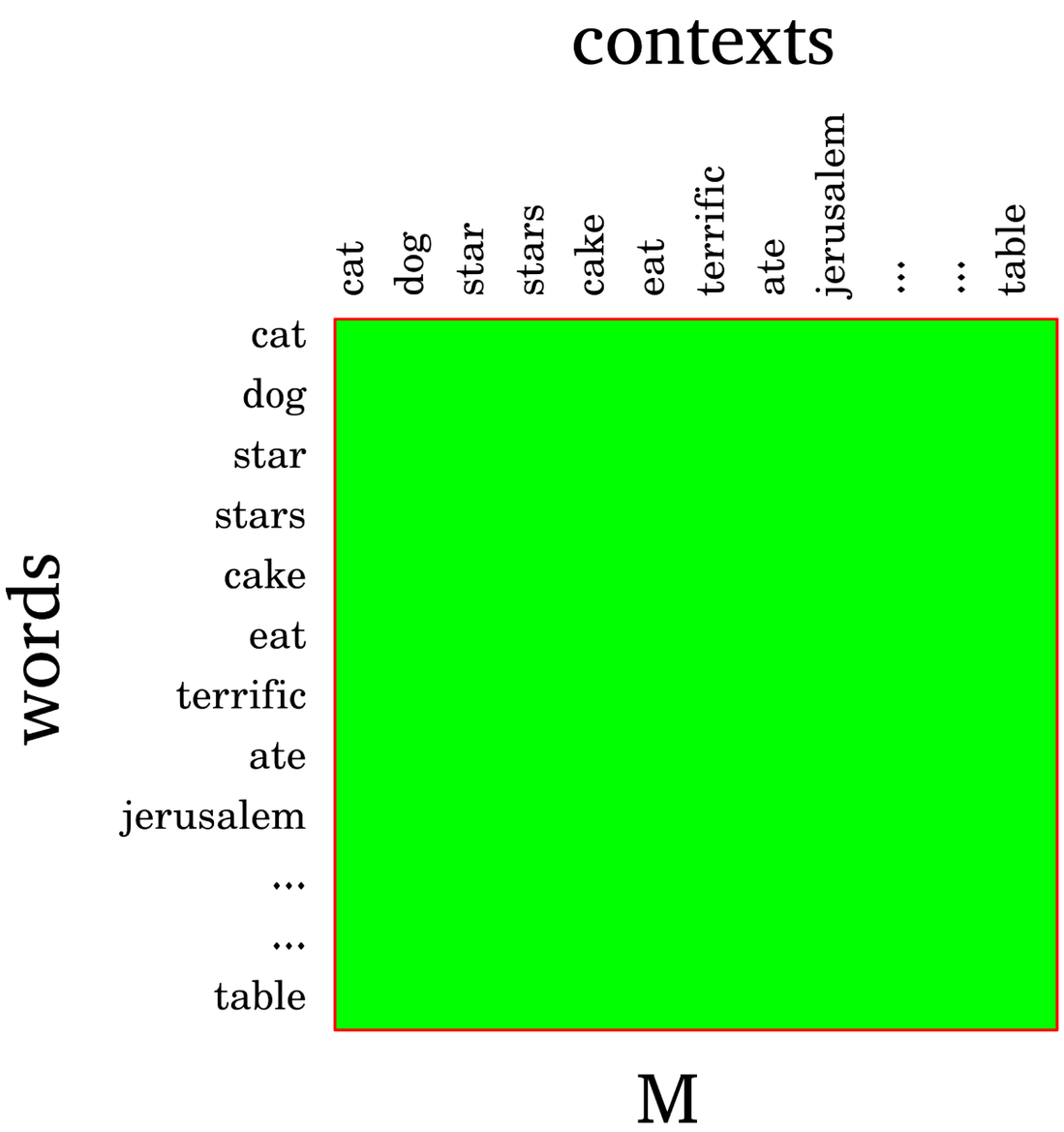
Other weighting schemes:

- TF-IDF
- Local Mutual Information
- Dice

See Ch4 of J.R. Curran's thesis (2004) and S. Evert's thesis (2007) for surveys of weighting methods

Words as Vectors

We can arrange the words in a huge, sparse matrix, where each row is a word, and each column is a context.

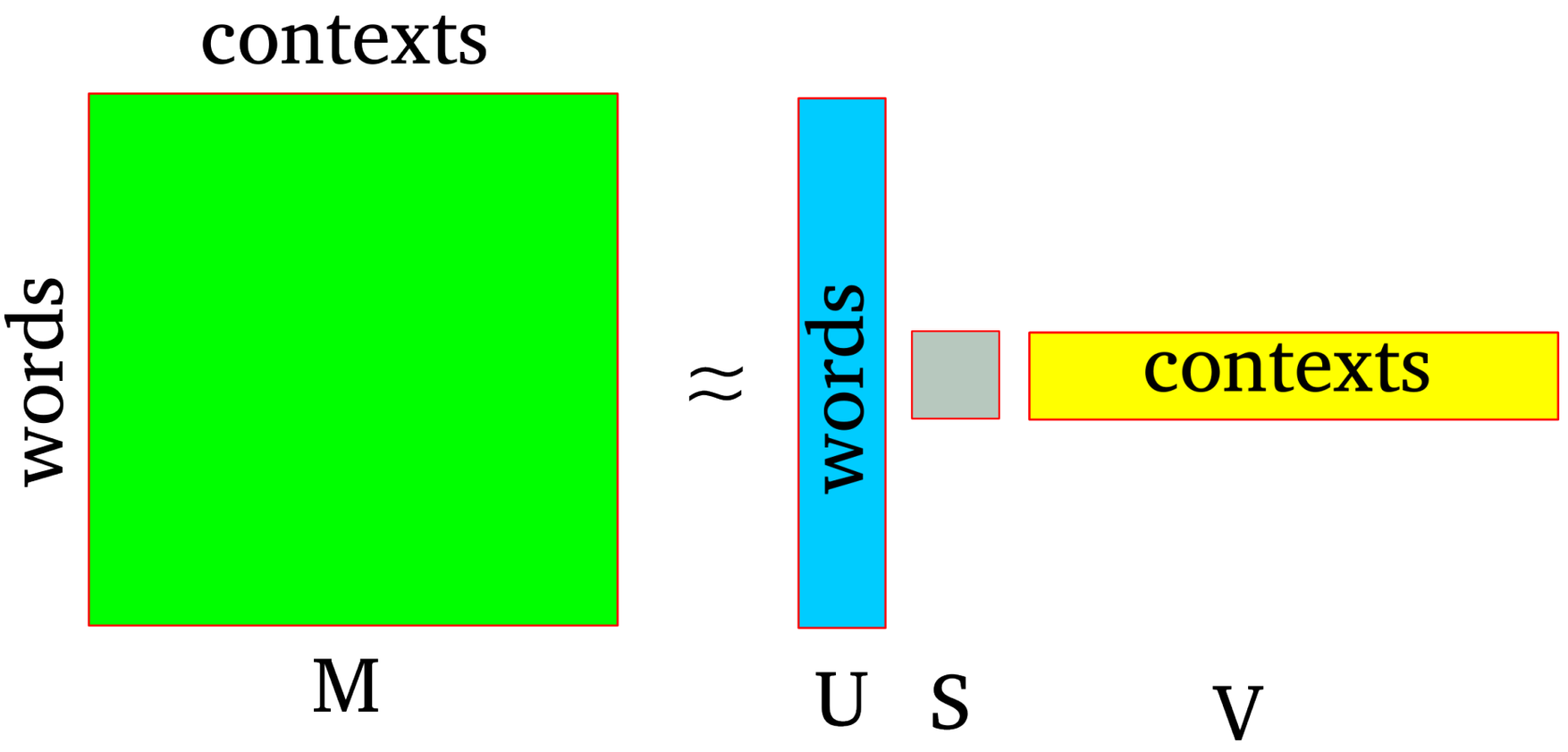


Dimensionality reduction

- Vector spaces often range from tens of thousands to millions of context dimensions
- Some of the methods to reduce dimensionality:
  - Select context features based on various relevance criteria
  - Random indexing
  - Following claimed to also have a beneficial *smoothing* effect:
    - Singular Value Decomposition
    - Non-negative matrix factorization
    - Probabilistic Latent Semantic Analysis
    - Latent Dirichlet Allocation

Words as Vectors

We often apply SVD or similar technique of dimensionality reduction.



Words as Vectors – It works

Nearest neighbours to dog

- cat
- horse
- fox
- pet
- rabbit
- pig
- animal
- mongrel
- sheep
- pigeon

Words as Vectors – It works

Nearest neighbours to dog

2-word window

- cat
- horse
- fox
- pet
- rabbit
- pig
- animal
- mongrel
- sheep
- pigeon

Words as Vectors – It works

Nearest neighbours to dog

2-word window

- cat
- horse
- fox
- pet
- rabbit
- pig
- animal
- mongrel
- sheep
- pigeon

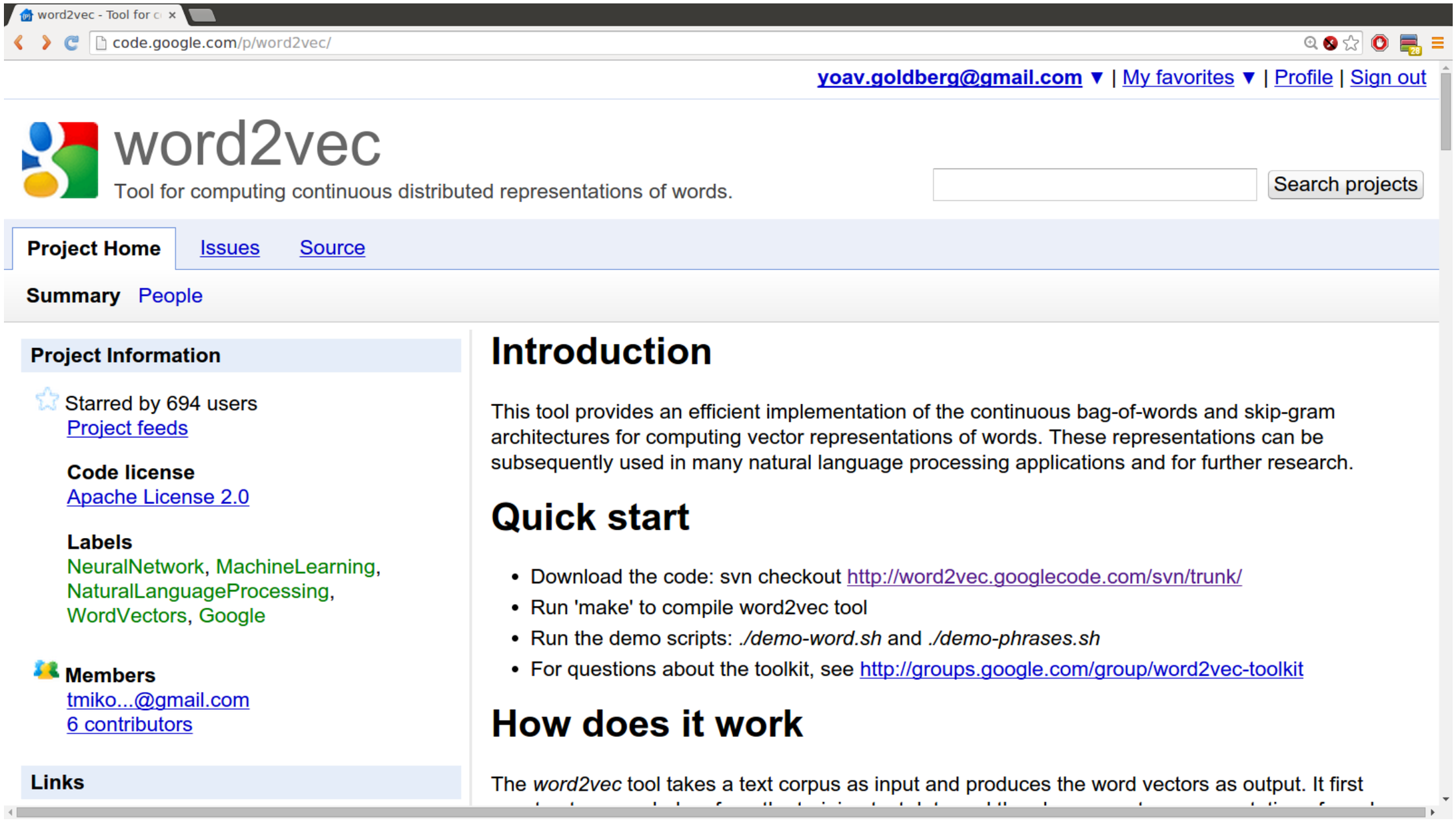
30-word window

- kennel
- puppy
- pet
- bitch
- terrier
- rottweiler
- canine
- cat
- to bark
- Alastian

note that the choice of context has a very important effect.

so does the choice of underlying corpus.

word2vec


word2vec - Tool for c
code.google.com/p/word2vec/
yoav.goldberg@gmail.com ▼ | My favorites ▼ | Profile | Sign out
word2vec
Tool for computing continuous distributed representations of words.
Search projects
Project Home
Issues
Source
Summary
People
Project Information
Starred by 694 users
Project feeds
Code license
Apache License 2.0
Labels
NeuralNetwork, MachineLearning, NaturalLanguageProcessing, WordVectors, Google
Members
tmiko...@gmail.com
6 contributors
Links
Introduction
This tool provides an efficient implementation of the continuous bag-of-words and skip-gram architectures for computing vector representations of words. These representations can be subsequently used in many natural language processing applications and for further research.
Quick start
Download the code: svn checkout http://word2vec.googlecode.com/svn/trunk/
Run 'make' to compile word2vec tool
Run the demo scripts: ./demo-word.sh and ./demo-phrases.sh
For questions about the toolkit, see http://groups.google.com/group/word2vec-toolkit
How does it work
The word2vec tool takes a text corpus as input and produces the word vectors as output. It first

Google

word2vec

Web Images **Videos** News Maps More Search tools

About 384 results (0.56 seconds)

MLMU.cz - Radim Řehůřek - Word2vec & friends (7.1.2015 ...

www.youtube.com/watch?v=wTp3P2UnTfQ

Jan 14, 2015 - Uploaded by Marek Modrý

I'll go over a particular model published by Google, called **word2vec**, its optimizations, applications and ...

Word2Vec convergence on Vimeo

https://vimeo.com/112168934

Nov 18, 2014

This is "**Word2Vec** convergence" by MaciejLyst on Vimeo, the home for high quality videos and the people who ...

Statistical Semantic入門~分布仮説からword2vecまで #1 ...

www.ustream.tv/recorded/43497190

Statistical Semantic入門~分布仮説から**word2vec**まで #1. February 5, 2014 at 7:16pm ...

Statistical Semantic入門~分布仮説からword2vecまで #2, PFI ...

www.ustream.tv/recorded/43497424

Feb 5, 2014

非常に説明がわかりやすいです！「ゲーミフィケーション入門」と「マーケティングとスタートアップの話」を見ましたが、どちらも 非常に理解しやすかった ...

GigaOM Show: Samsung watch secrets spilled! B&N's Nook ...

https://www.nicovico.com/.../528a406e7f094232b7b7

O'REILLY®

VIEW CART

Book

- Theme
  - Perl
  - Java
  - Web & Internet
  - XML
  - データベース
  - Security
  - Linux
  - Unix
  - Macintosh/OS X
  - Windows
  - ネットワーク/システム管理
  - プログラミング
  - ハードウェア
  - Math
- Series
  - Hacks
  - クックブック
  - クイックリファレンス
  - デスクトップリファレンス
  - The Missing Manual
  - Make
  - Head First

Feature

- 新刊情報
- カタログ（PDF）
- Makezine

word2vecによる自然言語処理

西尾泰和
2014年05月 発行
ISBN978-4-87311-683-9
フォーマット ePub mobi

Ebook Storeで電子版を購入:
価格1,512円

カートに入れる

内容 目次

Tomas Mikolovらによって提案されたニューラルネットワーク（CBOW, Skip-gram）のオープンソース実装word2vecについて、基本的な使い方を体験し、さらにその仕組みを学ぶ書籍です。
基本的な使い方から、自分の好きなコーパスの作り方、登場の背景、仕組み、さらには応用例や弱点についてもコンパクトなボリュームで概観できます。付録にはword2vecの出力結果を主成分分析を使って可視化する方法について解説しています。
なお本書はEbook版のみの販売となります。

著者の西尾さんによる本書の解題[リンク]

Catch O'Reilly

- New Release
- Ebook Store
- オラの村
- Make: Japan
- ORJ on Twitter
- Bookclub News
- ORJ for Mobile

Feedback

皆さんのご意見をお聞かせください。ご購入いただいた書籍やオライリー・ジャパンへのご感想やご意見、ご提案などをお聞かせください。より良い書籍づくりやサービス改良のための参考にさせていただきます。
[feedbackページへ]

Compare Search terms

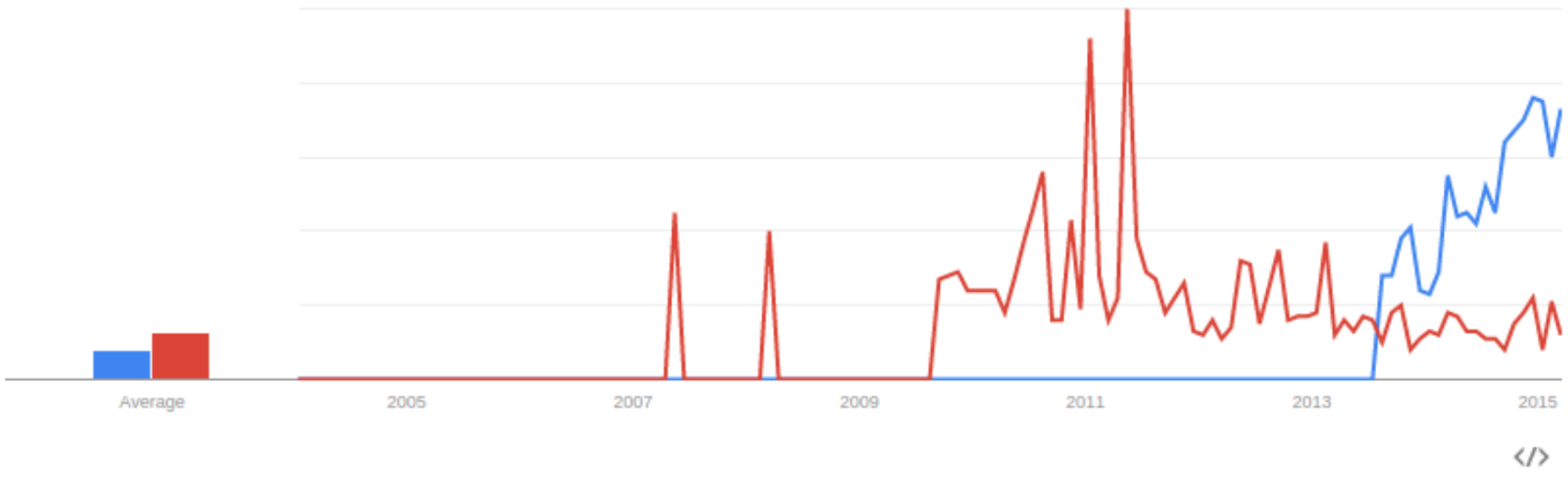
word2vec
Search term

dependency parsing
Search term

+ Add term

Interest over time

News headlines Forecast

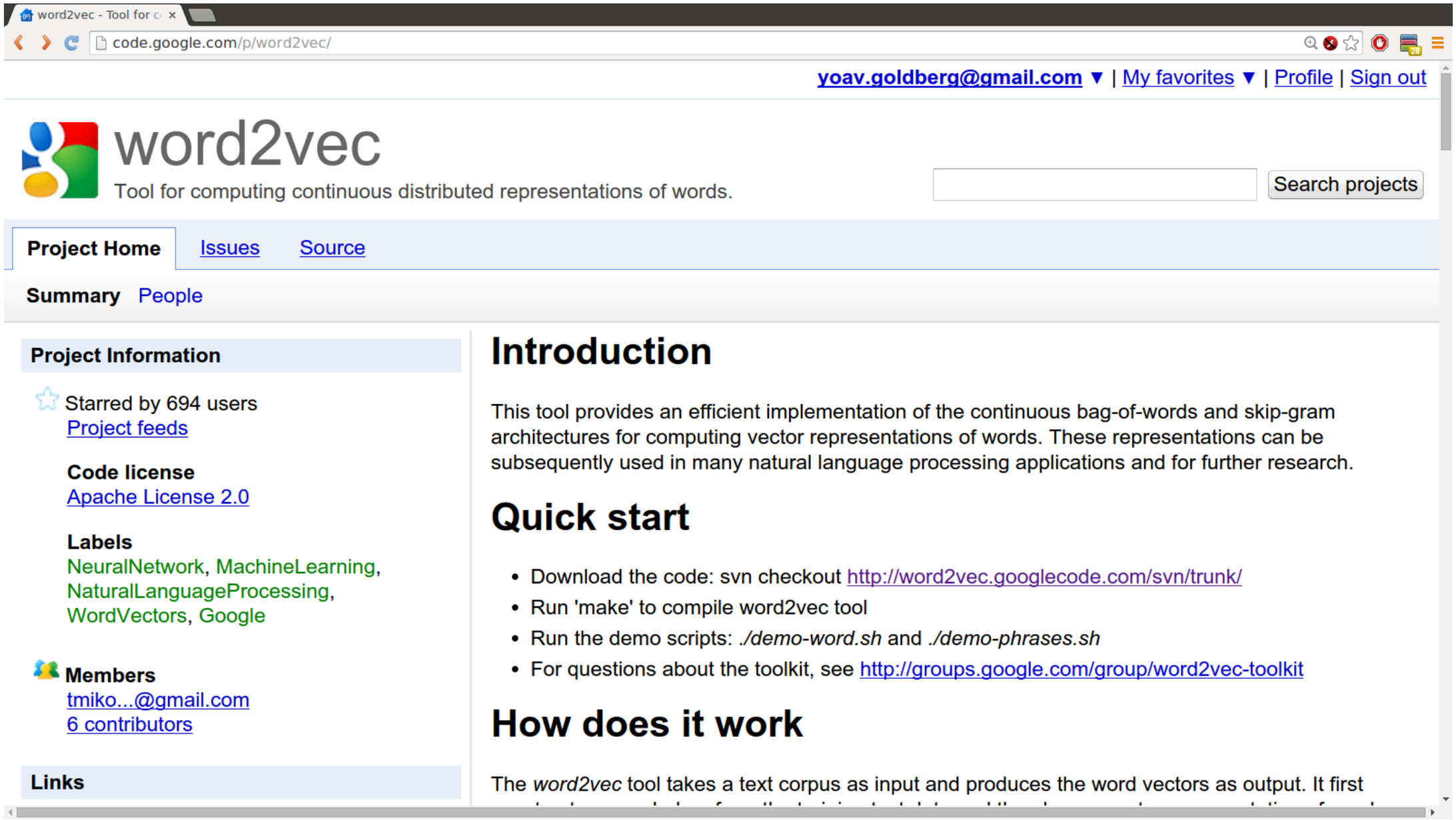


From Distributional to Distributed Semantics

This part of the talk

- `word2vec` as a black box
- a peek inside the black box
- relation between word-embeddings and the distributional representation
- tailoring word embeddings to your needs using `word2vecf`

word2vec

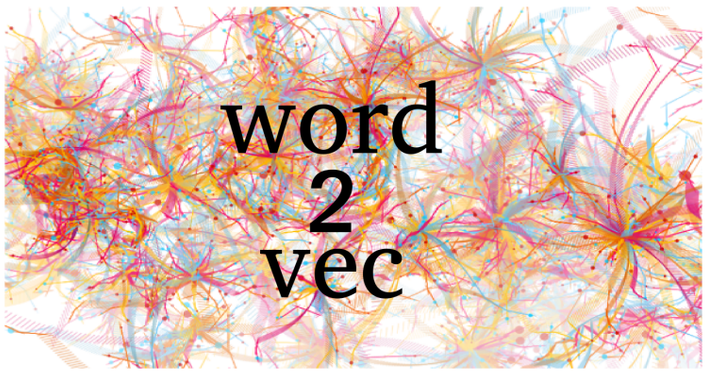

word2vec - Tool for c
code.google.com/p/word2vec/
yoav.goldberg@gmail.com ▼ | My favorites ▼ | Profile | Sign out
word2vec
Tool for computing continuous distributed representations of words.
Search projects
Project Home
Issues
Source
Summary
People
Project Information
Starred by 694 users
Project feeds
Code license
Apache License 2.0
Labels
NeuralNetwork, MachineLearning, NaturalLanguageProcessing, WordVectors, Google
Members
tmiko...@gmail.com
6 contributors
Links
Introduction
This tool provides an efficient implementation of the continuous bag-of-words and skip-gram architectures for computing vector representations of words. These representations can be subsequently used in many natural language processing applications and for further research.
Quick start
Download the code: svn checkout http://word2vec.googlecode.com/svn/trunk/
Run 'make' to compile word2vec tool
Run the demo scripts: ./demo-word.sh and ./demo-phrases.sh
For questions about the toolkit, see http://groups.google.com/group/word2vec-toolkit
How does it work
The word2vec tool takes a text corpus as input and produces the word vectors as output. It first

word2vec

feed in text

Text

WIKIPEDIA



wait a few hours

d

|V| words

dog = (0.12,-0.32,0.92,0.43,-0.3 ...)
cat = (0.15,-0.29,0.90,0.39,-0.32 ...)

chair = (0.8,0.9,-0.76,0.29,0.52 ...)

get a |V|xd matrix W where each row is a vector for a word

word2vec

- dog
  - cat, dogs, dachshund, rabbit, puppy, poodle, rottweiler, mixed-breed, doberman, pig
- sheep
  - cattle, goats, cows, chickens, sheeps, hogs, donkeys, herds, shorthorn, livestock
- november
  - october, december, april, june, february, july, september, january, august, march
- jerusalem
  - tiberias, jaffa, haifa, israel, palestine, nablus, damascus katamon, ramla, safed
- teva
  - pfizer, schering-plough, novartis, astrazeneca, glaxosmithkline, sanofi-aventis, mylan, sanofi, genzyme, pharmacia

Working with Dense Vectors

Word Similarity

- Similarity is calculated using *cosine similarity*:

$$sim(\vec{dog}, \vec{cat}) = \frac{\vec{dog} \cdot \vec{cat}}{||\vec{dog}|| \; ||\vec{cat}||}$$

- For normalized vectors ($||x|| = 1$), this is equivalent to a dot product:

$$sim(\vec{dog}, \vec{cat}) = \vec{dog} \cdot \vec{cat}$$

- **Normalize the vectors when loading them.**

Working with Dense Vectors

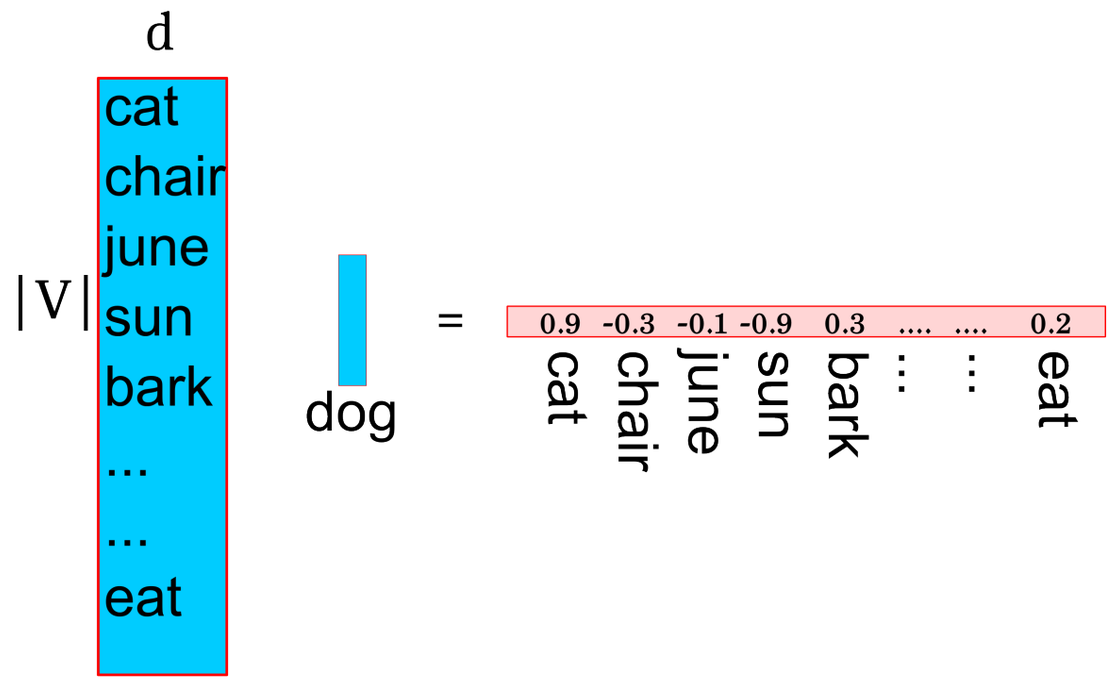
Finding the most similar words to $\vec{dog}$

- Compute the similarity from word $\vec{v}$ to all other words.

Working with Dense Vectors

Finding the most similar words to $\vec{dog}$

- Compute the similarity from word $\vec{v}$ to all other words.
- This is a **single matrix-vector product**: $W \cdot \vec{v}^\top$

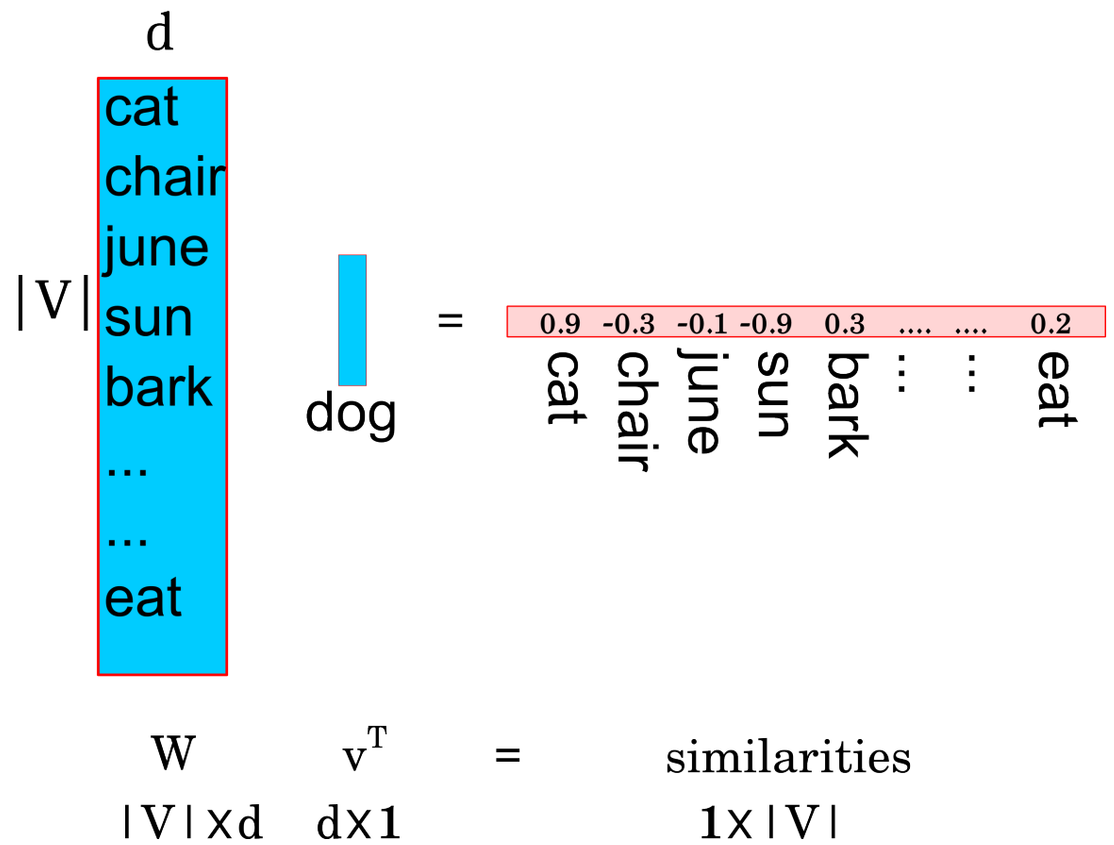


W |V|xd v^T dx1 = similarities 1x|V|

Working with Dense Vectors

Finding the most similar words to $\vec{dog}$

- Compute the similarity from word $\vec{v}$ to all other words.
- This is a **single matrix-vector product**: $W \cdot \vec{v}^{\top}$

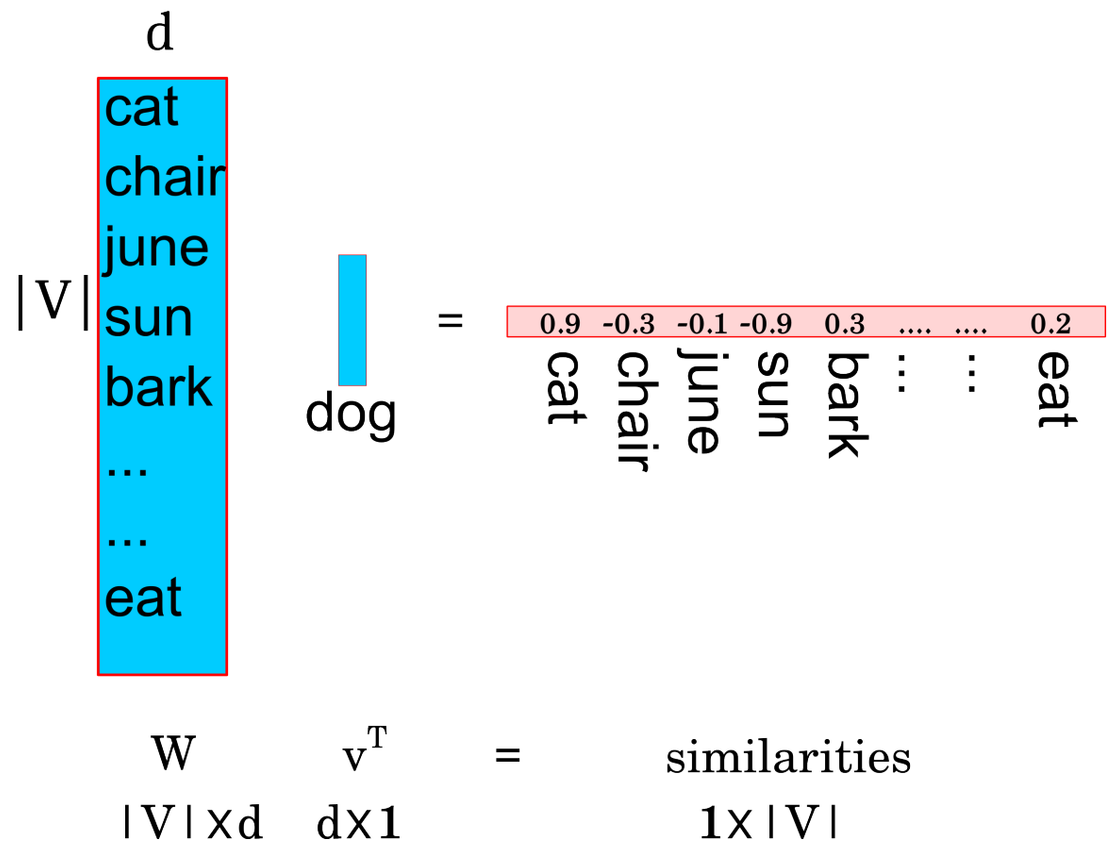


- Result is a $|V|$ sized vector of similarities.
- Take the indices of the k-highest values.

Working with Dense Vectors

Finding the most similar words to $\vec{dog}$

- Compute the similarity from word $\vec{v}$ to all other words.
- This is a **single matrix-vector product**: $W \cdot \vec{v}^\top$



- Result is a $|V|$ sized vector of similarities.
- Take the indices of the k-highest values.
- **FAST! for 180k words, d=300: $\sim$30ms**

Working with Dense Vectors

Most Similar Words, in python+numpy code

```
W,words = load_and_normalize_vectors("vecs.txt")
# W and words are numpy arrays.
w2i = {w:i for i,w in enumerate(words)}

dog = W[w2i['dog']] # get the dog vector

sims = W.dot(dog)    # compute similarities

most_similar_ids = sims.argsort()[-1:-10:-1]
sim_words = words[most_similar_ids]
```

Working with Dense Vectors

Similarity to a group of words

- “Find me words most similar to cat, dog and cow”.
- Calculate the pairwise similarities and sum them:

$$W \cdot \vec{cat} + W \cdot \vec{dog} + W \cdot \vec{cow}$$

- Now find the indices of the highest values as before.

Working with Dense Vectors

Similarity to a group of words

- "Find me words most similar to cat, dog and cow".
- Calculate the pairwise similarities and sum them:

$$W \cdot \vec{cat} + W \cdot \vec{dog} + W \cdot \vec{cow}$$

- Now find the indices of the highest values as before.
- Matrix-vector products are wasteful. **Better option:**

$$W \cdot (\vec{cat} + \vec{dog} + \vec{cow})$$

Working with dense word vectors can be very efficient.

Working with dense word vectors can be very efficient.

But where do these vectors come from?

How does word2vec work?

word2vec implements several different algorithms:

Two training methods

- Negative Sampling
- Hierarchical Softmax

Two context representations

- Continuous Bag of Words (CBOW)
- Skip-grams

How does word2vec work?

word2vec implements several different algorithms:

Two training methods

- **Negative Sampling**
- Hierarchical Softmax

Two context representations

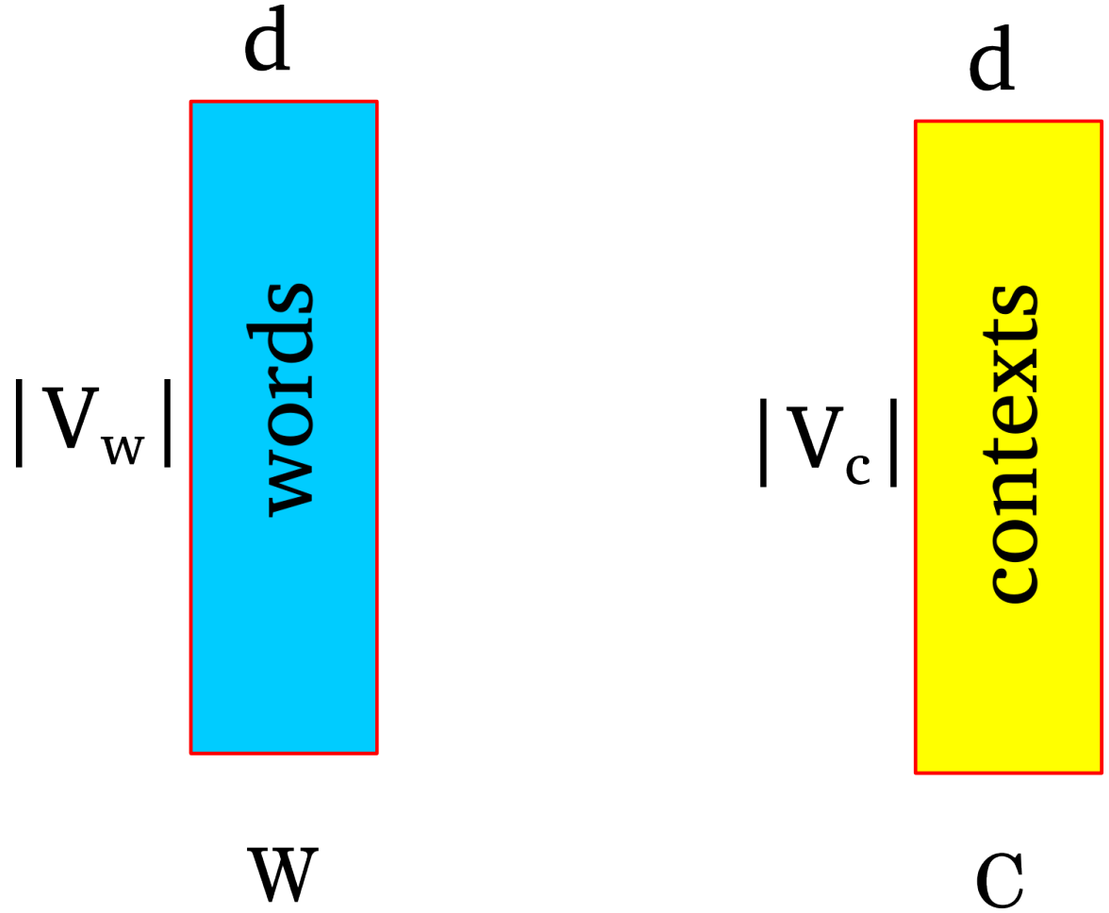
- Continuous Bag of Words (CBOW)
- **Skip-grams**

We'll focus on skip-grams with negative sampling.

intuitions apply for other models as well.

How does word2vec work?

- Represent each word as a *d* dimensional vector.
- Represent each context as a *d* dimensional vector.
- Initalize all vectors to random weights.
- Arrange vectors in two matrices, *W* and *C*.



How does word2vec work?

While more text:

- Extract a word window:

A springer is [a cow or **heifer** close to calving].

c_1 c_2 c_3 w c_4 c_5 c_6

- w is the focus word vector (row in W).
- c_i are the context word vectors (rows in C).

How does word2vec work?

While more text:

- Extract a word window:

```
A springer is[ a   cow  or  heifer  close  to  calving ].
              c1   c2   c3    w      c4     c5     c6
```

- Try setting the vector values such that:

$$\sigma(w \cdot c_1) + \sigma(w \cdot c_2) + \sigma(w \cdot c_3) + \sigma(w \cdot c_4) + \sigma(w \cdot c_5) + \sigma(w \cdot c_6)$$

is **high**

How does word2vec work?

While more text:

- Extract a word window:

A springer is [a cow or **heifer** close to calving].

c_1 c_2 c_3 w c_4 c_5 c_6

- Try setting the vector values such that:

$$\sigma(w \cdot c_1)+\sigma(w \cdot c_2)+\sigma(w \cdot c_3)+\sigma(w \cdot c_4)+\sigma(w \cdot c_5)+\sigma(w \cdot c_6)$$

is **high**

- Create a corrupt example by choosing a random word w'

[a cow or **comet** close to calving]

c_1 c_2 c_3 w' c_4 c_5 c_6

- Try setting the vector values such that:

$$\sigma(w' \cdot c_1)+\sigma(w' \cdot c_2)+\sigma(w' \cdot c_3)+\sigma(w' \cdot c_4)+\sigma(w' \cdot c_5)+\sigma(w' \cdot c_6)$$

is **low**

Word2vec Formulation

D : a set of original (correct) word-context pairs

$\overline{D}$: a set of corrupt (incorrect) word-context pairs

- *k negative samples* are generated for each correct sample

The model needs to estimate:

$P(D=1|w,c)$: the probability that (w,c) is from D

– Should be high for pairs from D, low if

$P(D=0|w,c) = 1 - P(D=1|w,c)$: $\overline{D}$

the probability that (w,c) is from

Word2vec Formulation (cont.)

Modeling probability as sigmoid of dot product score $s(w,c)$:

$$P(D=1|w,c) = \frac{1}{1+e^{-s(w,c)}}$$

Learning goal: find vectors w,c for all words and contexts that maximize log-likelihood of data $D \cup \bar{D}$:

$$\mathcal{L}(\Theta; D, \bar{D}) = \sum_{(w,c)\in D} \log P(D=1|w,c) + \sum_{(w,c)\in \bar{D}} \log P(D=0|w,c)$$

Negative samples generated by original frequency, or smoothed - deemphasizing frequent words, better in practice:

$$\frac{\#(w)}{\sum_{w'} \#(w')} \quad \text{or} \quad \frac{\#(w)^{0.75}}{\sum_{w'} \#(w')^{0.75}}$$

Word2vec Formulation (cont.)

The *Skip-gram* model assumes independence between the context elements.

Denote $c_{1:k}$ a context of k elements:

$$P(D=1|w,c_i) = \frac{1}{1+e^{-\mathbf{w}\cdot\mathbf{c_i}}}$$

$$P(D=1|w,c_{1:k}) = \prod_{i=1}^{k} P(D=1|w,c_i) = \prod_{1=i}^{k} \frac{1}{1+e^{-\mathbf{w}\cdot\mathbf{c_i}}}$$

$$\log P(D=1|w,c_{1:k}) = \log \sum_{i=1}^{k} \frac{1}{1+e^{-\mathbf{w}\cdot\mathbf{c_i}}}$$

Very effective in practice, commonly used.

How does word2vec work?

The training procedure results in:

- $w \cdot c$ for **good** word-context pairs is **high**.
- $w \cdot c$ for **bad** word-context pairs is **low**.
- $w \cdot c$ for **ok-ish** word-context pairs is **neither high nor low**.

As a result:

- Words that share many contexts get close to each other.
- Contexts that share many words get close to each other.

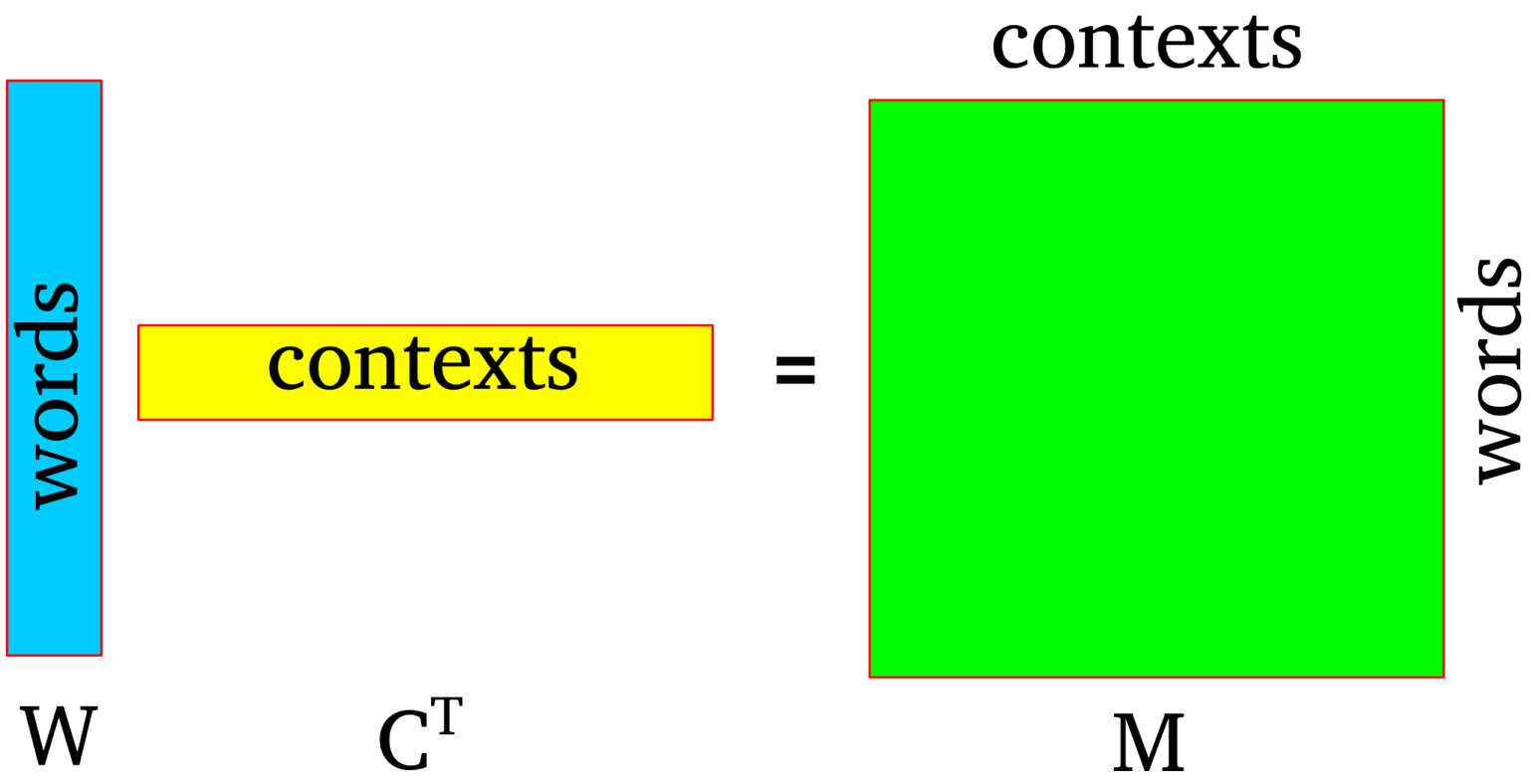
At the end, word2vec throws away C and returns W.

Reinterpretation

Imagine we didn't throw away C. Consider the product $WC^\top$

Reinterpretation

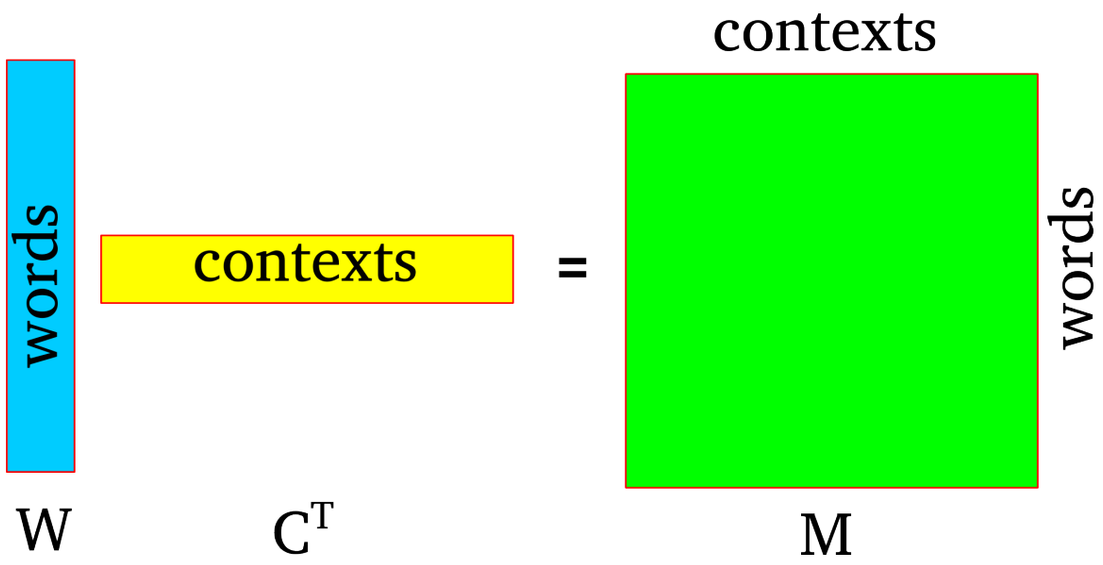
Imagine we didn't throw away *C*. Consider the product $WC^\top$



The result is a matrix *M* in which:

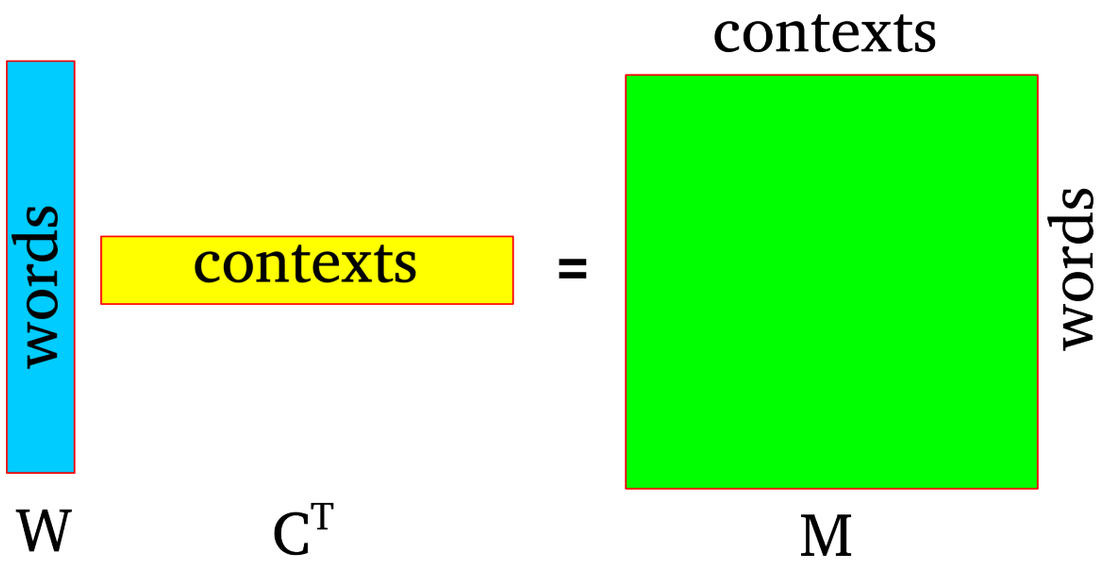
- Each row corresponds to a word.
- Each column corresponds to a context.
- Each cell correspond to $w \cdot c$, an association measure between a word and a context.

Reinterpretation



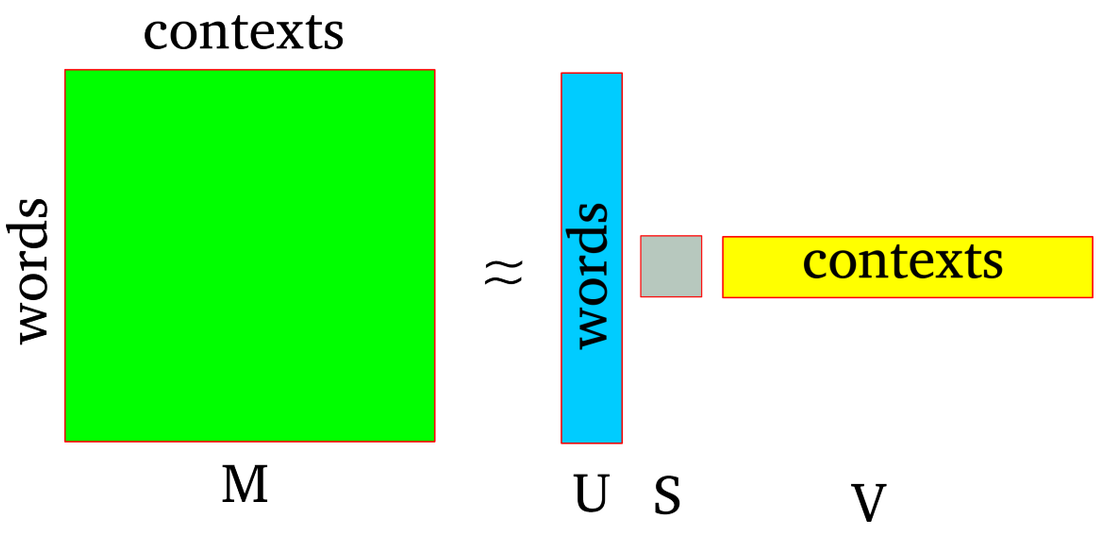
Does this remind you of something?

Reinterpretation



Does this remind you of something?

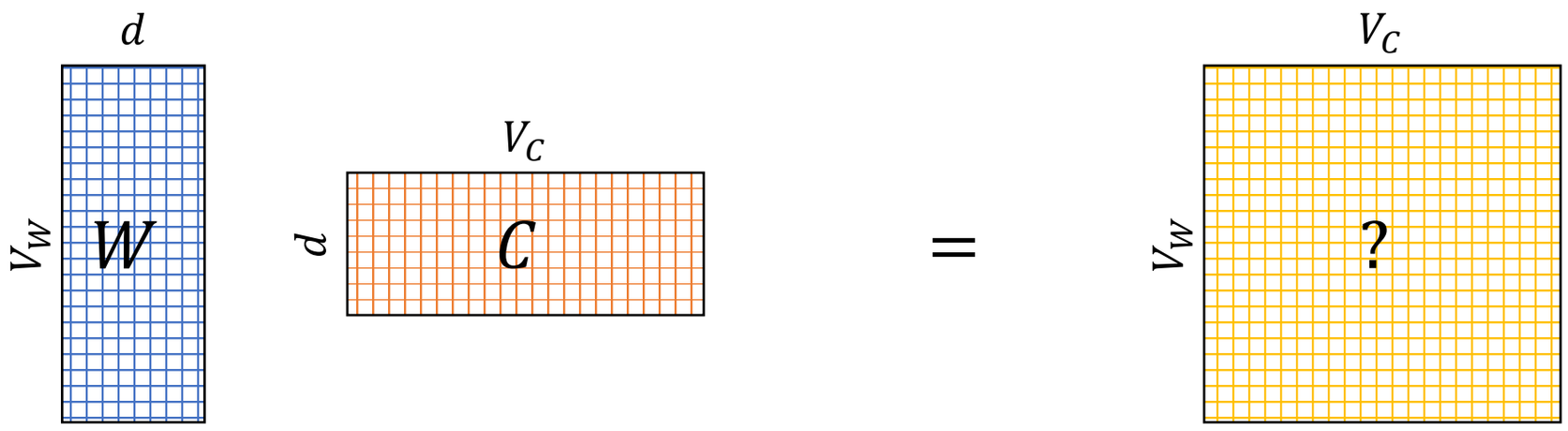
Very similar to SVD over distributional representation:



What is SGNS learning?

- A $V_W \times V_C$ matrix
- Each cell describes the relation between a specific word-context pair

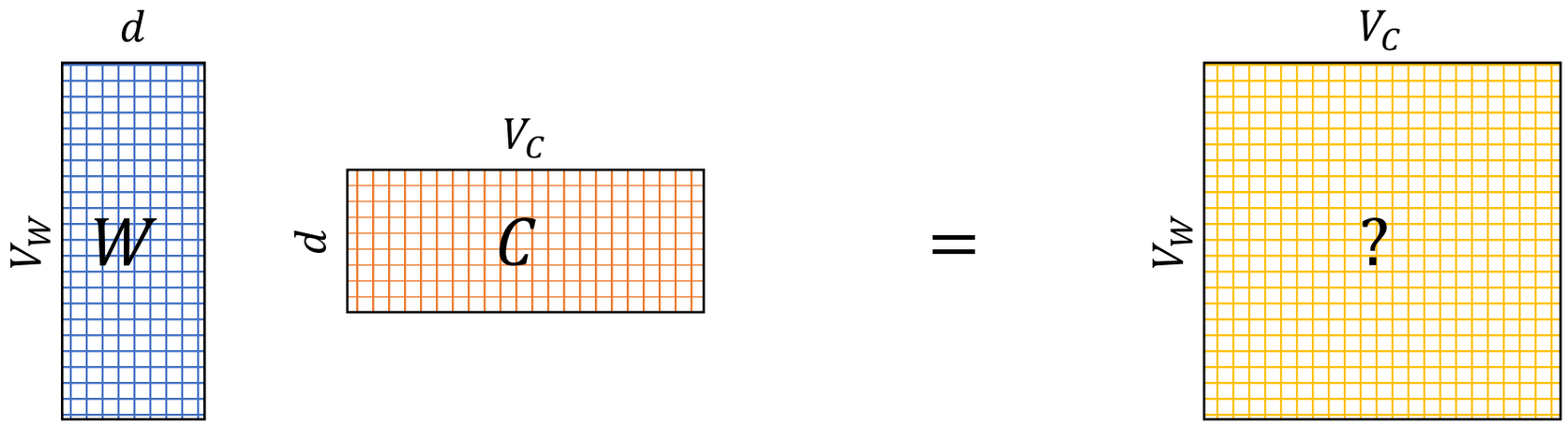
$$\vec{w} \cdot \vec{c} = ?$$



"Neural Word Embeddings as Implicit Matrix Factorization"
Levy & Goldberg, NIPS 2014

What is SGNS learning?

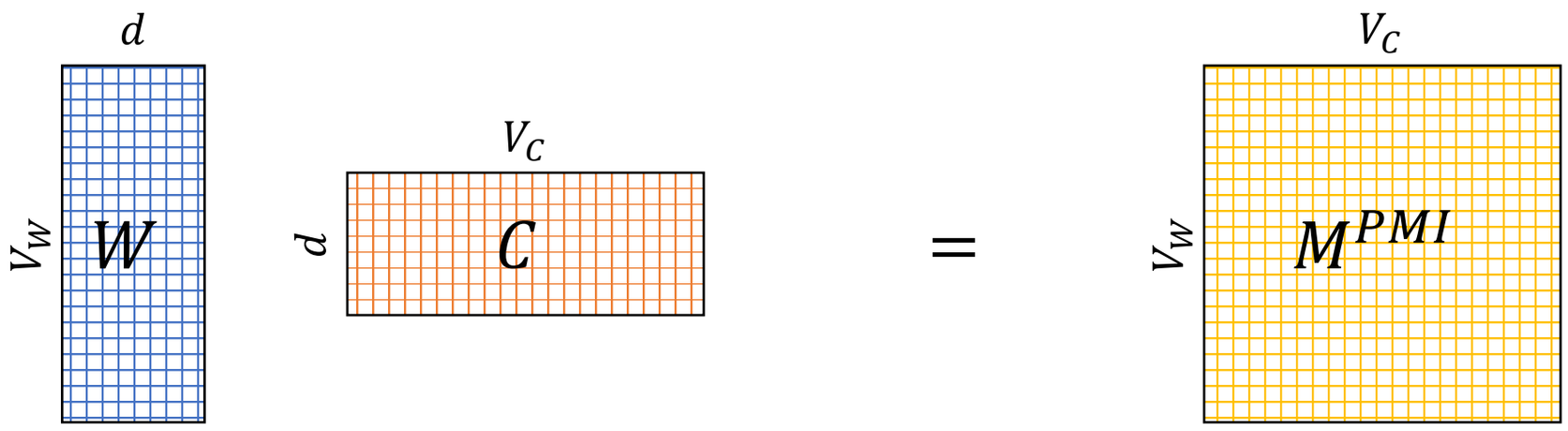
- We **prove** that for large enough d and enough iterations



"Neural Word Embeddings as Implicit Matrix Factorization"
Levy & Goldberg, NIPS 2014

What is SGNS learning?

- We **prove** that for large enough d and enough iterations
- We get the word-context PMI matrix

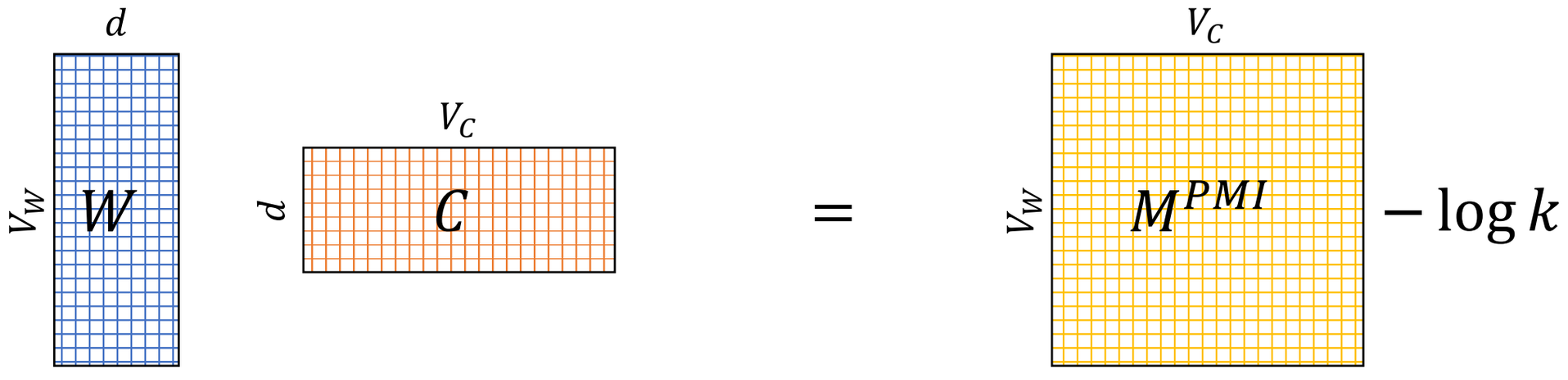


"Neural Word Embeddings as Implicit Matrix Factorization"
Levy & Goldberg, NIPS 2014

What is SGNS learning?

- We **prove** that for large enough d and enough iterations
- We get the word-context PMI matrix, shifted by a global constant

$$Opt(\vec{w} \cdot \vec{c}) = PMI(w, c) - \log k$$



"Neural Word Embeddings as Implicit Matrix Factorization"
Levy & Goldberg, NIPS 2014

What is SGNS learning?

- SGNS is doing something very similar to the older approaches

- SGNS is factorizing the traditional word-context PMI matrix

- So does SVD!

- Do they capture the same similarity function?

SGNS vs SVD

Target Word	SGNS	SVD
cat	dog rabbit cats poodle pig	dog rabbit pet monkey pig

SGNS vs SVD

Target Word	SGNS	SVD
wine	wines	wines
	grape	grape
	grapes	grapes
	winemaking	varietal
	tasting	vintages

SGNS vs SVD

Target Word	SGNS	SVD
November	October December April January July	October December April June March

But `word2vec` is still better, isn't it?

- Plenty of evidence that `word2vec` outperforms traditional methods
  - In particular: "Don't count, predict!" (Baroni et al., 2014)
- How does this fit with our story?

The Big Impact of “Small” Hyperparameters

Hyperparameters

- `word2vec` is more than just an algorithm…

- Introduces many **engineering tweaks** and **hyperpararameter settings**
  - May seem minor, but **make a big difference** in practice
  - Their impact is often more significant than the embedding algorithm's

- These modifications can be ported to distributional methods!

Levy, Goldberg, Dagan (In submission)

Hyperparameters

- Preprocessing
- Association Metric
- Postprocessing

Levy, Goldberg, Dagan (In submission)

Hyperparameters

- Preprocessing
- **Association Metric**
- Postprocessing

Levy, Goldberg, Dagan (In submission)

Association Metric Hyperparameters

- Since SGNS and PMI are strongly related,
  we can import 2 of SGNS's hyperparameters to traditional PMI:

1. Shifted PMI
2. **Negative Sampling Smoothing**

- Both stem from the negative sampling procedure

Levy, Goldberg, Dagan (In submission)

Negative Sampling Smoothing

- Recall that SGNS picks $w' \sim P$ to form negative (w', c) examples

- Our analysis assumes P is the unigram distribution

$$P(w) = \frac{\#w}{\sum_{w' \in V_W} \#w'}$$

Levy, Goldberg, Dagan (In submission)

Negative Sampling Smoothing

- Recall that SGNS picks $w' \sim P$ to form negative (w', c) examples

- Our analysis assumes P is the unigram distribution

- In practice, it's a **smoothed** unigram distribution

$$P^{0.75}(w) = \frac{(\#w)^{0.75}}{\sum_{w' \in V_W} (\#w')^{0.75}}$$

- This little change makes a big difference

Levy, Goldberg, Dagan (In submission)

Negative Sampling Smoothing

- This smoothing has an analogue in PMI

- Replace $P(w)$ with $P^{0.75}(w)$:

$$PMI^{0.75}(w,c) = \log \frac{P(w,c)}{P^{0.75}(w)P(c)}$$

- Yields a **dramatic** improvement with **every method** on **every task**

Levy, Goldberg, Dagan (In submission)

Experiments & Results

- We compared several methods, while controlling for hyperparameters
  - PPMI, SVD(PPMI), SGNS, GloVe
- Methods perform on-par in most tasks
  - Slight advantage to SVD in word similarity
  - SGNS is better at syntactic analogies
  - SGNS is robust in general

- **Negative sampling smoothing** accounts for much of the differences observed in “Don’t count, predict!”

Levy, Goldberg, Dagan (In submission)

Other Hyperparameters

- There are many other hyperparameters that can be investigated

- Perhaps the most interesting one is **the type of context**

What’s in a Context?

What's in a Context?

- Importing ideas from embeddings improves distributional methods

- Can distributional ideas also improve embeddings?

- **Idea:** change SGNS's default **BoW contexts** into **dependency contexts**

"Dependency-Based Word Embeddings"
Levy & Goldberg, ACL 2014

Example

Australian scientist discovers star with telescope

“Dependency-Based Word Embeddings”
Levy & Goldberg, ACL 2014

Target Word

Australian scientist discovers star with telescope

“Dependency-Based Word Embeddings”
Levy & Goldberg, ACL 2014

Bag of Words (BoW) Context

Australian scientist discovers star with telescope

“Dependency-Based Word Embeddings”
Levy & Goldberg, ACL 2014

Bag of Words (BoW) Context

Australian scientist discovers star with telescope

“Dependency-Based Word Embeddings”
Levy & Goldberg, ACL 2014

Bag of Words (BoW) Context

Australian scientist discovers star with telescope

"Dependency-Based Word Embeddings"
Levy & Goldberg, ACL 2014

Syntactic Dependency Context

Australian scientist discovers star with telescope

“Dependency-Based Word Embeddings”
Levy & Goldberg, ACL 2014

Syntactic Dependency Context

"Dependency-Based Word Embeddings"
Levy & Goldberg, ACL 2014

Syntactic Dependency Context

"Dependency-Based Word Embeddings"
Levy & Goldberg, ACL 2014

Embedding Similarity with Different Contexts

Target Word	Bag of Words (k=5)	Dependencies
Hogwarts (Harry Potter's school)	Dumbledore hallows half-blood Malfoy Snape	Sunnydale Collinwood Calarts Greendale Millfield
	Related to Harry Potter	**Schools**

"Dependency-Based Word Embeddings"
Levy & Goldberg, ACL 2014

Embedding Similarity with Different Contexts

Target Word	Bag of Words (k=5)	Dependencies
Turing (computer scientist)	nondeterministic non-deterministic computability deterministic finite-state	Pauling Hotelling Heting Lessing Hamming
	Related to computability	**Scientists**

"Dependency-Based Word Embeddings"
Levy & Goldberg, ACL 2014

Embedding Similarity with Different Contexts

Target Word	Bag of Words (k=5)	Dependencies
dancing (dance gerund)	singing dance dances dancers tap-dancing	singing rapping breakdancing miming busking
	Related to dance	**Gerunds**

"Dependency-Based Word Embeddings"
Levy & Goldberg, ACL 2014

What is the effect of different context types?

- Thoroughly studied in distributional methods
  - Lin (1998), Padó and Lapata (2007), and many others…

General Conclusion:

- Bag-of-words contexts induce ***topical*** similarities
- Dependency contexts induce ***functional*** similarities
  - Share the same semantic type
  - Cohyponyms

- Holds for **embeddings** as well

"Dependency-Based Word Embeddings"
Levy & Goldberg, ACL 2014

- Same algorithm, different inputs -- very different kinds of similarity.

- Inputs matter much more than algorithm.

- **Think about your inputs.**

Peeking into Skip-Gram's Black Box

- In explicit representations, we can **look** at the features and analyze
- But embeddings are a black box!
- Dimensions are latent and don't necessarily have any meaning

"Dependency-Based Word Embeddings"
Levy & Goldberg, ACL 2014

Peeking into Skip-Gram's Black Box

- Skip-Gram allows a peek…

- Contexts are embedded in the same space!

- Given a word w, find the contexts c it "activates" most:

$$\arg\max_{c}(\vec{w} \cdot \vec{c})$$

"Dependency-Based Word Embeddings"
Levy & Goldberg, ACL 2014

Associated Contexts

Target Word	Dependencies
Hogwarts	students/prep_at^{-1} educated/prep_at^{-1} student/prep_at^{-1} stay/prep_at^{-1} learned/prep_at^{-1}

"Dependency-Based Word Embeddings"
Levy & Goldberg, ACL 2014

Associated Contexts

Target Word	Dependencies
Turing	machine/nn^{-1} test/nn^{-1} theorem/$poss^{-1}$ machines/nn^{-1} tests/nn^{-1}

"Dependency-Based Word Embeddings"
Levy & Goldberg, ACL 2014

Associated Contexts

Target Word	Dependencies
dancing	dancing/conj dancing/conj^{-1} singing/conj^{-1} singing/conj ballroom/nn

"Dependency-Based Word Embeddings"
Levy & Goldberg, ACL 2014

let's take a step back

- We don't really care about the vectors.
- We care about the **similarity function** they induce.
  - (or, maybe we want to use them in an external task)
- We want similar words to have similar vectors.
- So evaluating on word-similarity tasks is great.
- **But what does similar mean?**

many faces of similarity

- dog -- cat
- dog -- poodle
- dog -- animal
- dog -- bark
- dog -- leash

many faces of similarity

- dog -- cat
- dog -- poodle
- dog -- animal
- dog -- bark
- dog -- leash
- dog -- chair
- dog -- dig
- dog -- god
- dog -- fog
- dog -- 6op

many faces of similarity

- dog -- cat
- dog -- poodle
- dog -- animal
- dog -- bark
- dog -- leash

- dog -- chair — same POS
- dog -- dig — edit distance
- dog -- god — same letters
- dog -- fog — rhyme
- dog -- 6op — shape

some forms of similarity look more useful than they really are

- Almost every algorithm you come up with will be good at capturing:
  - countries
  - cities
  - months
  - person names

some forms of similarity look more useful than they really are

- Almost every algorithm you come up with will be good at capturing:
  - countries
  - cities
  - months
  - person names

useful for tagging/parsing/NER

some forms of similarity look more useful than they really are

- Almost every algorithm you come up with will be good at capturing:
  - countries
  - cities
  - months
  - person names

useful for tagging/parsing/NER

but do we really want
"John went to China in June"
to be similar to
"Carl went to Italy in February"
??

there is no single downstream task

- Different tasks require different kinds of similarity.
- Different vector-inducing algorithms produce different similarity functions.
- **No single representation for all tasks.**
- If your vectors do great on task X, I don't care that they suck on task Y.

Context matters

Choose the correct contexts for your application

- larger window sizes – more topical
- dependency relations – more functional

Context matters

Choose the correct contexts for your application

- larger window sizes – more topical
- dependency relations – more functional
- only noun-adjective relations
- only verb-subject relations

Context matters

Choose the correct contexts for your application

- larger window sizes – more topical
- dependency relations – more functional
- only noun-adjective relations
- only verb-subject relations
- context: time of the current message
- context: user who wrote the message

Context matters

Choose the correct contexts for your application

- larger window sizes – more topical
- dependency relations – more functional
- only noun-adjective relations
- only verb-subject relations
- context: time of the current message
- context: user who wrote the message
- . . .
- the sky is the limit

what happens in Hebrew?

(based on my work with Oded Avraham)

what happens when we look outside of English?

- Things don't work nearly as well.
- Known problems from English become more extreme.
- We get some new problems as well.

a quick look at Hebrew

word senses

ספר

book(N). barber(N). counted(V). tell!(V). told(V).

חומה

brown (feminine, singular)
wall (noun)
her fever (possessed noun)

multi-word units

- עורך דין
- בית ספר
- שומר ראש
- יושב ראש
- ראש עיר
- בית שימוש

words vs. tokens

וכשמהבית

and when from the house

words vs. tokens

וכשמהבית

and when from the house

בצל

in shadow

בצל

onion

and of course: inflections

- nouns, pronouns and adjectives
  --> are inflected for *number* and *gender*

- verbs
  --> are inflected for *number*, *gender*, *tense*, *person*

- syntax requires *agreement* between
  - nouns and adjectives
  - verbs and subjects

and of course: inflections

she **saw** a **brown** fox

he **saw** a **brown** fence

and of course: inflections

[fem] **[masc]**

she **saw** a **brown** fox

he **saw** a **brown** fence

[masc] **[fem]**

and of course: inflections

[fem] **[masc]**

she **saw** a **brown** fox

היא **ראתה** שועל **חום**

הוא **ראה** גדר **חומה**

he **saw** a **brown** fence

[masc] **[fem]**

inflections and dist-sim

- More word forms -- more sparsity

- But more importantly: *agreement patterns affect the resulting similarities.*

adjectives

green [m,sg] **ירוק**	**green [f,sg]** **ירוקה**	**green [m,pl]** **ירוקים**
כחול blue [m,sg]	אפורה gray [f,sg]	אפורים gray [m,pl]
כתום orange [m,sg]	כתומה orange [f,sg]	כחולים blue [m,pl]
צהוב yellow [m,sg]	צהובה yellow [f,sg]	שחורים black [m,pl]
אדום red [m,sg]	קסומה magical [f,g]	שמימיים heavenly [m,pl]

verbs

(he) walked **הלך**	**(she) thought** **חשבה**	**(they) ate** **אכלו**
הלכו (they) walked	חושבת (she) is thinking	יאכלו (they) will eat
הולך (he) is walking	הרגישה (she) felt	אוכלים (they) are eating
פנה (he) turned	משוכנעת (she) is convinved	אכל (he) ate
התקרב (he) came closer	התעקשה (she) insisted	שתה (they) drank

nouns

Doctor [m,sg] **רופא**	**Doctor [f, sg]** **רופאה**
פסיכיאטר psychiatrist [m,sg]	סטודנטית student [f, sg]
פסיכולוג psychologist [m, sg]	נזירה nun [f, sg]
נוירולוג neurologist [m, sg]	מלצרית waitress [f, sg]
מהנדס engineer [m, sg]	צלמת photographer [f, sg]

Bias

- The word vectors capture language use.
- Language use may contain biases.
- We need to be aware of it when using the vectors.

nouns

sweater **סוודר**		**shirt** **חולצה**	
ג׳קט	jacket	חליפה	suit
חלוק	down	גלימה	robe
אוברול	overall	שמלה	dress
טורבאן	turban	קסדה	helmet

nouns

sweater **סוודר**		**shirt** **חולצה**	
ג׳קט	jacket	חליפה	suit
חלוק	down	גלימה	robe
אוברול	overall	שמלה	dress
טורבאן	turban	קסדה	helmet
masculine		**feminine**	

nouns

sweater **סוודר**		**shirt** **חולצה**	
ג׳קט	jacket	חליפה	suit
חלוק	down	גלימה	robe
אוברול	overall	שמלה	dress
טורבאן	turban	קסדה	helmet
masculine		**feminine**	

completely arbitrary

When arbitrary gender does matter

נפלה נפל

When arbitrary gender does matter

נפל	נפלה
נהרג	נכבשה

inflections and dist-sim

- Inflections and agreement really influence the results.
- We get a mix of syntax and semantics.
- Which aspect of the similarity we care about? what does it mean to be similar?
- Need better control of the different aspects.

inflections and dist-sim

- Work with lemmas instead of words!!
- Sure, but where do you get the lemmas?
- ...for unknown words?
- And what should you lemmatize? everything? somethings? context-dependent?
- Ongoing work in my lab -- but still much to do.

Software

`word2vecf`

`https://bitbucket.org/yoavgo/word2vecf`

- Extension of `word2vec`.
- Allows saving the context matrix.
- Allows using **arbitraty contexts**.
  - Input is a (large) file of word context pairs.

Software

`hyperwords`

`https://bitbucket.org/omerlevy/hyperwords/`

- Python library for working with either sparse or dense word vectors (similarity, analogies).
- Scripts for creating dense representations using word2vecf or SVD.
- Scripts for creating sparse distributional representations.

Software

`dissect`
`http://clic.cimec.unitn.it/composes/toolkit/`

- Given vector representation of words. . .
- . . . derive vector representation of phrases/sentences
- Implements various composition methods

Summary

Distributional Semantics

- Words in similar contexts have similar meanings.
- Represent a word by the contexts it appears in.
- But what is a context?

Neural Models (word2vec)

- Represent each word as dense, low-dimensional vector.
- Same intuitions as in distributional vector-space models.
- Efficient to run, scales well, modest memory requirement.
- Dense vectors are convenient to work with.
- Still helpful to think of the context types.

Software

- Build your own word representations.