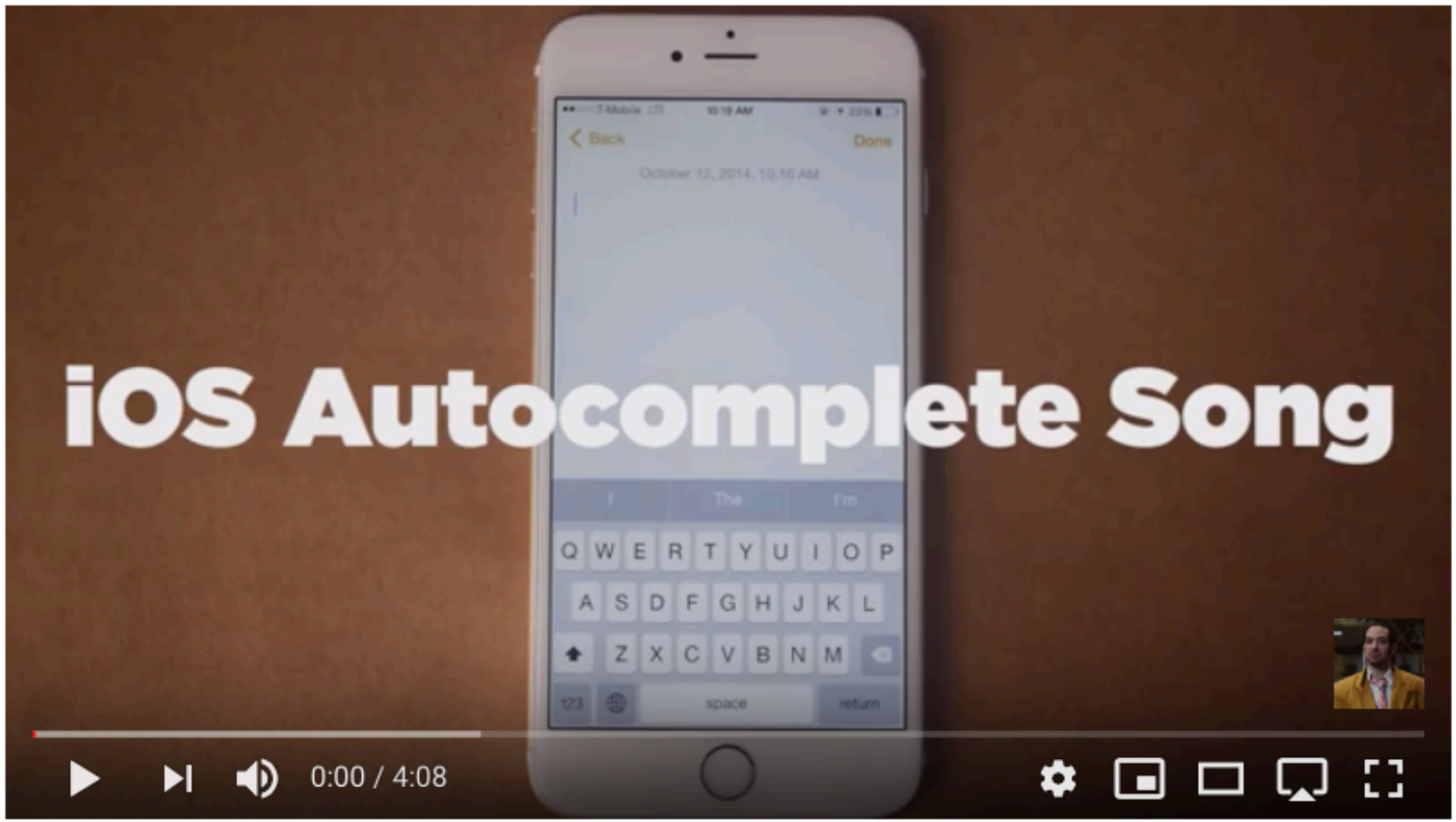
Probabilities for Language Modeling

Speech and Language Processing (3rd Edition draft)

Chapter 3 “Language Modeling with N-Grams”



♫ iOS Autocomplete Song | Song A Day #2110

https://www.youtube.com/watch?v=M8MJFrdfGe0

Probabilistic Language Models

- Today's goal: assign a probability to a sentence
- Autocomplete for texting
- Machine Translation
- Spelling Correction
- Speech Recognition
- Other Natural Language Generation tasks: summarization, question-answering, dialog systems

Probabilistic Language Modeling

- Goal: compute the probability of a sentence or sequence of words

- Related task: probability of an upcoming word

- A model that computes either of these

- Better: **the grammar** But **language model** or **LM** is standard

Probabilistic Language Modeling

- Goal: compute the probability of a sentence or sequence of words

$$P(W) = P(w_1,w_2,w_3,w_4,w_5 \ldots w_n)$$

- Related task: probability of an upcoming word

$$P(w_5|w_1,w_2,w_3,w_4)$$

- A model that computes either of these

$P(W)$ or $P(w_n|w_1,w_2 \ldots w_{n-1})$ is called a **language model**.

- Better: **the grammar** But **language model** or **LM** is standard

How to compute P(W)

- How to compute this joint probability:
  - P(the, underdog, Philadelphia, Eagles, won)
- Intuition: let’s rely on the Chain Rule of Probability

The Chain Rule

The Chain Rule

- Recall the definition of conditional probabilities

  $p(B|A) = P(A,B)/P(A)$ Rewriting: **$P(A,B) = P(A)P(B|A)$**

- More variables:

  $P(A,B,C,D) = P(A)P(B|A)P(C|A,B)P(D|A,B,C)$

- The Chain Rule in General

$P(x_1,x_2,x_3,\ldots,x_n) = P(x_1)P(x_2|x_1)P(x_3|x_1,x_2)\ldots P(x_n|x_1,\ldots,x_{n-1})$

The Chain Rule applied to compute joint probability of words in sentence

The Chain Rule applied to compute joint probability of words in sentence

$$P(w_1 w_2 \ldots w_n) = \prod_i P(w_i \mid w_1 w_2 \ldots w_{i-1})$$

P("the underdog Philadelphia Eagles won") =

P(the) × P(underdog|the) × P(Philadelphia|the underdog)

× P(Eagles|the underdog Philadelphia)

× P(won|the underdog Philadelphia Eagles)

How to estimate these probabilities

- Could we just count and divide?

How to estimate these probabilities

- Could we just count and divide? Maximum likelihood estimation (MLE)

$$P(\text{won} \mid \text{the underdog team}) = \frac{\text{Count(the underdog team won)}}{\text{Count(the underdog team)}}$$

- Why doesn't this work?

Simplifying Assumption = Markov Assumption

Simplifying Assumption = Markov Assumption

- $P(won|the\ underdog\ team) \approx P(won|team)$
- Only depends on the previous k words, not the whole context
- $\approx P(won|underdog\ team)$
- $\approx P(w_i|w_{i-2}\ w_{i-1})$
- $P(w_1w_2w_3w_4\ldots w_n) \approx \prod_i^n P(w_i|w_{i-k}\ldots w_{i-1})$
- K is the number of context words that we take into account

How much history should we use?

unigram	**no history**	$\prod_{i}^{n} \mathrm{p}(w_i)$	$p(w_i) = \frac{count(w_i)}{all\ words}$
bigram	1 word as history	$\prod_{i}^{n} \mathrm{p}(w_i \mid w_{i-1})$	$p(w_i \mid w_{i-1}) = \frac{count(w_{i-1} w_i)}{count(w_{i-1})}$
trigram	2 words as history	$\prod_{i}^{n} \mathrm{p}(w_i \mid w_{i-2} w_{i-1})$	$p(w_i \mid w_{i-2} w_{i-1}) = \frac{count(w_{i-2} w_{i-1} w_i)}{count(w_{i-2} w_{i-1})}$
4-gram	3 words as history	$\prod_{i}^{n} \mathrm{p}(w_i \mid w_{i-3} w_{i-2} w_{i-1})$	$p(w_i \mid w_{i-3} w_{i-2} w_{i-1}) = \frac{count(w_{i-3} w_{i-2} w_{i-1} w_i)}{count(w_{i-3} w_{i-3} w_{i-1})}$

Historical Notes

1913	**Andrei Markov counts 20k letters in *Eugene Onegin***
1948	Claude Shannon uses n-grams to approximate English
1956	Noam Chomsky decries finite-state Markov Models
1980s	Fred Jelinek at IBM TJ Watson uses n-grams for ASR, think about 2 other ideas for models: (1) MT, (2) stock market prediction
1993	Jelinek at team develops statistical machine translation $argmax_e p(e\|f) = p(e)\, p(f\|e)$
	Jelinek left IBM to found CLSP at JHU Peter Brown and Robert Mercer move to Renaissance Technology

Simplest case: Unigram model

$$P(w_1 w_2 \ldots w_n) \approx \prod_i P(w_i)$$

Some automatically generated sentences from a unigram model

```
fifth an of futures the an incorporated a a the
inflation most dollars quarter in is mass

thrift did eighty said hard 'm july bullish

that or limited the
```

Bigram model

- Condition on the previous word:

$$P(w_i \mid w_1 w_2 \ldots w_{i-1}) \approx P(w_i \mid w_{i-1})$$

```
texaco rose one in this issue is pursuing growth in a boiler
house said mr. gurria mexico 's motion control proposal

without permission from five hundred fifty five yen

outside new car parking lot of the agreement reached

this would be a record november
```

N-gram models

- We can extend to trigrams, 4-grams, 5-grams
- In general this is an insufficient model of language
  - because language has **long-distance dependencies**:

    "The computer(s) which I had just put into the machine room on the fifth floor is (are) crashing."

- But we can often get away with N-gram models

Language Modeling

Estimating N-gram Probabilities

Estimating bigram probabilities

- The Maximum Likelihood Estimate

$$P(w_i \mid w_{i-1}) = \frac{count(w_{i-1}, w_i)}{count(w_{i-1})}$$

$$P(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

An example

$$P(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

<s> I am Sam </s>

<s> Sam I am </s>

<s> I do not like green eggs and ham </s>

$$P(\texttt{I} \mid \texttt{<s>}) = \tfrac{2}{3} = .67 \qquad P(\texttt{Sam} \mid \texttt{<s>}) = \tfrac{1}{3} = .33 \qquad P(\texttt{am} \mid \texttt{I}) = \tfrac{2}{3} = .67$$

$$P(\texttt{</s>} \mid \texttt{Sam}) = \tfrac{1}{2} = 0.5 \qquad P(\texttt{Sam} \mid \texttt{am}) = \tfrac{1}{2} = .5 \qquad P(\texttt{do} \mid \texttt{I}) = \tfrac{1}{3} = .33$$

Problems for MLE

- Zeros

Train	Test
denied the allegations	denied the memo
denied the reports	
denied the claims	
denied the requests	

- P(memo|denied the) = 0
- And we also assign 0 probability to all sentences containing it!

Problems for MLE

- Out of vocabulary items (OOV)
- <unk> to deal with OOVs
- Fixed lexicon L of size V
- Normalize training data by replacing any word not in L with <unk>

- Avoid zeros with smoothing

Practical Issues

- We do everything in log space
  - Avoid underflow
  - (also adding is faster than multiplying)

$$\log(p_1 \times p_2 \times p_3 \times p_4) = \log p_1 + \log p_2 + \log p_3 + \log p_4$$

Language Modeling Toolkits

- SRILM
  - http://www.speech.sri.com/projects/srilm/
- KenLM
  - https://kheafield.com/code/kenlm/

Google N-Gram Release, August 2006

AUG
3

All Our N-gram are Belong to You

Posted by Alex Franz and Thorsten Brants, Google Machine Translation Team

Here at Google Research we have been using word n-gram models for a variety of R&D projects,

…

That's why we decided to share this enormous dataset with everyone. We processed 1,024,908,267,229 words of running text and are publishing the counts for all 1,176,470,663 five-word sequences that appear at least 40 times. There are 13,588,391 unique words, after discarding words that appear less than 200 times.

Google N-Gram Release

- serve as the incoming 92
- serve as the incubator 99
- serve as the independent 794
- serve as the index 223
- serve as the indication 72
- serve as the indicator 120
- serve as the indicators 45
- serve as the indispensable 111
- serve as the indispensible 40
- serve as the individual 234

http://googleresearch.blogspot.com/2006/08/all-our-n-gram-are-belong-to-you.html

Google Book N-grams

- http://ngrams.googlelabs.com/

Language Modeling

Evaluation and Perplexity

Evaluation: How good is our model?

- Does our language model prefer good sentences to bad ones?
  - Assign higher probability to “real” or “frequently observed” sentences
    - Than “ungrammatical” or “rarely observed” sentences?
- We train parameters of our model on a **training set**.
- We test the model’s performance on data we haven’t seen.
  - A **test set** is an unseen dataset that is different from our training set, totally unused.
  - An **evaluation metric** tells us how well our model does on the test set.

Training on the test set

- We can’t allow test sentences into the training set
- We will assign it an artificially high probability when we set it in the test set
- “Training on the test set”
- Bad science!
- And violates the honor code

Extrinsic evaluation of N-gram models

Difficulty of extrinsic (task-based) evaluation of N-gram models

- Extrinsic evaluation
  - Time-consuming; can take days or weeks
- So
  - Sometimes use **intrinsic** evaluation: **perplexity**
  - Bad approximation
    - unless the test data looks **just** like the training data
    - So **generally only useful in pilot experiments**
  - But is helpful to think about.

Intuition of Perplexity

- The Shannon Game:
  - How well can we predict the next word?

I always order pizza with cheese and ____

Intuition of Perplexity

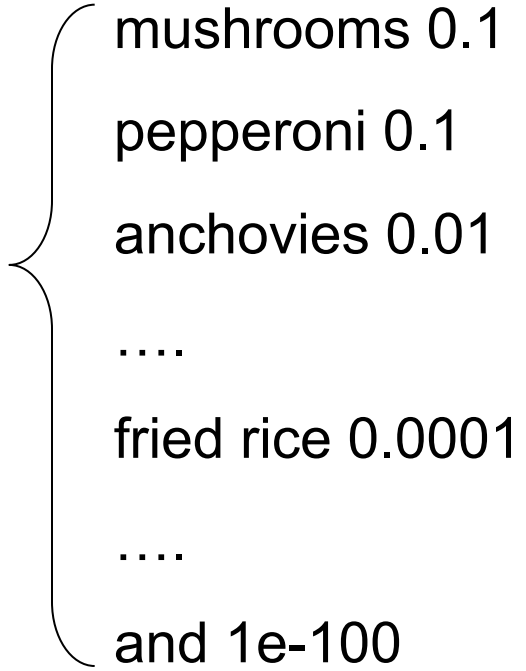
- The Shannon Game:
  - How well can we predict the next word?

    I always order pizza with cheese and ____

    The 33rd President of the US was ____

    I saw a ____

  - Unigrams are terrible at this game. (Why?)



- A better model of a text
  - is one which assigns a higher probability to the word that actually occurs

Perplexity

The best language model is one that best predicts an unseen test set

- Gives the highest P(sentence)

Perplexity is the inverse probability of the test set, normalized by the number of words

Minimizing perplexity is the same as maximizing probability

Perplexity

The best language model is one that best predicts an unseen test set

- Gives the highest P(sentence)

Perplexity is the inverse probability of the test set, normalized by the number of words

$$PP(W) = P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}}$$

$$= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}$$

Chain rule:

$$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_1 \ldots w_{i-1})}}$$

For bigrams:

$$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_{i-1})}}$$

Minimizing perplexity is the same as maximizing probability

Perplexity as branching factor

- Let’s suppose a sentence consisting of random digits
- What is the perplexity of this sentence according to a model that assign P=1/10 to each digit?

Perplexity as branching factor

- Let's suppose a sentence consisting of random digits
- What is the perplexity of this sentence according to a model that assign P=1/10 to each digit?

$$
\begin{aligned}
\mathrm{PP}(W) &= P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}} \\
&= \left(\frac{1}{10}^{N}\right)^{-\frac{1}{N}} \\
&= \frac{1}{10}^{-1} \\
&= 10
\end{aligned}
$$

Lower perplexity = better model

Minimizing perplexity is the same as maximizing probability

- Training 38 million words, test 1.5 million words, WSJ

N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

Language Modeling

Generalization and zeros

The Shannon Visualization Method

- Choose a random bigram

  (<s>, w) according to its probability
- Now choose a random bigram (w, x) according to its probability
- And so on until we choose </s>
- Then string the words together

```
<s> I
    I want
      want to
           to eat
              eat Chinese
                  Chinese food
                          food  </s>
I want to eat Chinese food
```

Approximating Shakespeare

1 gram	–To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have
	–Hill he late speaks; or! a more to leg less first you enter
2 gram	–Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow.
	–What means, sir. I confess she? then all sorts, he is trim, captain.
3 gram	–Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, ’tis done.
	–This shall forbid it should be branded, if renown made it empty.
4 gram	–King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv’d in;
	–It cannot be but so.

Shakespeare as corpus

- N=884,647 tokens, V=29,066 types

Shakespeare as corpus

- N=884,647 tokens, V=29,066
- Shakespeare produced 300,000 bigram types out of V^2= 844 million possible bigrams.
  - So 99.96% of the possible bigrams were never seen (have zero entries in the table)
- 4-grams worse: What's coming out looks like Shakespeare because it ***is*** Shakespeare

The wall street journal is not shakespeare (no offense)

1 gram	Months the my and issue of year foreign new exchange's september were recession exchange new endorsed a acquire to six executives
2 gram	Last December through the way to preserve the Hudson corporation N. B. E. C. Taylor would seem to complete the major central planners one point five percent of U. S. E. has already old M. X. corporation of living on information such as more frequently fishing to keep her
3 gram	They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and Brazil on market conditions

Can you guess the author of these random 3-gram sentences?

They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and gram Brazil on market conditions

This shall forbid it should be branded, if renown made it empty.

“You are uniformly charming!” cried he, with a smile of associating and now and then I bowed and they perceived a chaise and four to wish for.

The perils of overfitting

- N-grams only work well for word prediction if the test corpus looks like the training corpus
  - In real life, it often doesn’t
  - We need to train robust models that generalize!
  - One kind of generalization: Zeros!
    - Things that don’t ever occur in the training set
      - But occur in the test set

Zero probability bigrams

- Bigrams with zero probability
  - mean that we will assign 0 probability to the test set!
- And hence we cannot compute perplexity (can’t divide by 0)!

Language Modeling

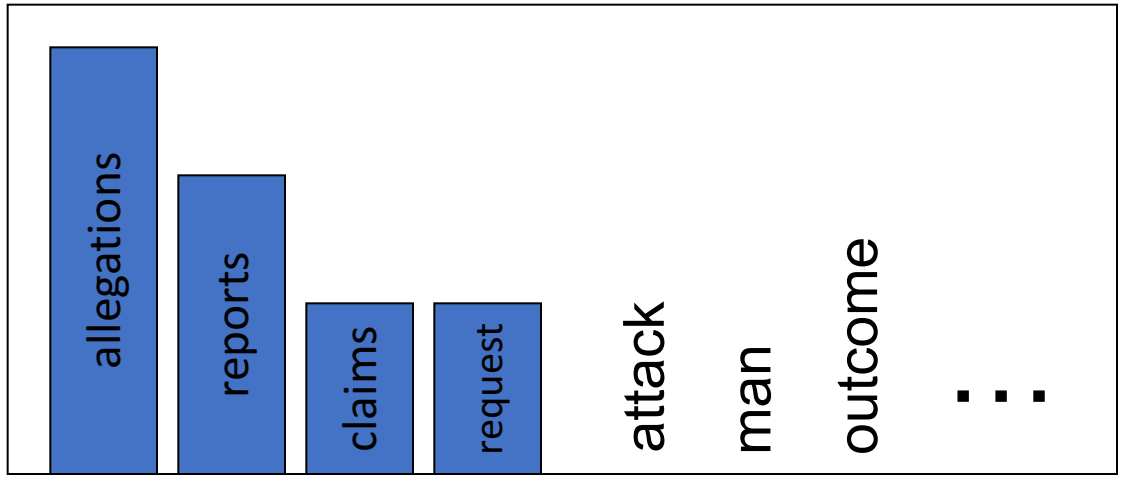
Smoothing: Add-one (Laplace) smoothing

The intuition of smoothing (from Dan Klein)

When we have sparse statistics:

P(w | denied the)
3 allegations
2 reports
1 claims
1 request

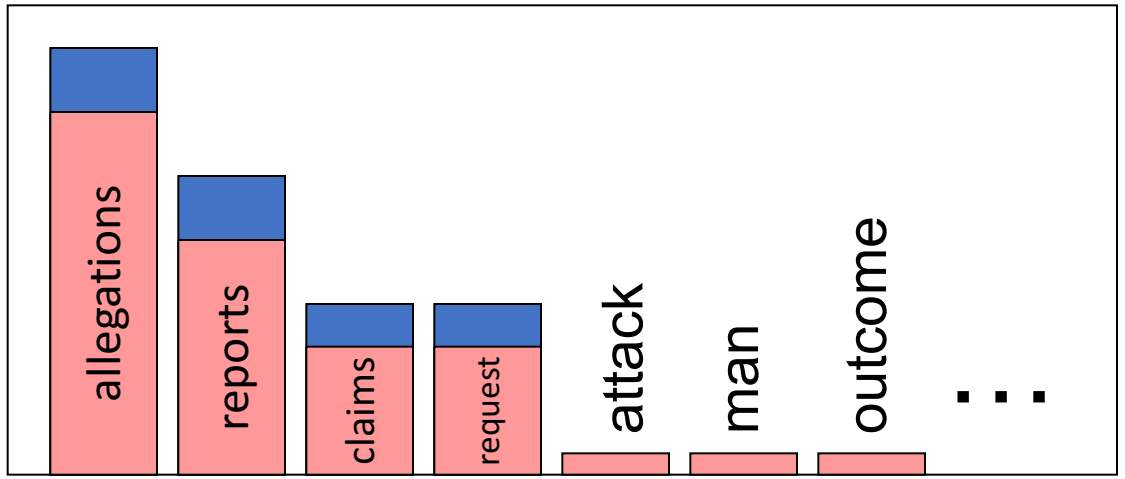
7 total



Steal probability mass to generalize better

P(w | denied the)
2.5 allegations
1.5 reports
0.5 claims
0.5 request
2 other

7 total



Add-one estimation

Also called Laplace smoothing

Pretend we saw each word one more time than we did

Just add one to all the counts!

MLE estimate:

$$P_{MLE}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

Add-1 estimate:

$$P_{Add-1}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i) + 1}{c(w_{i-1}) + V}$$

Maximum Likelihood Estimates

- The maximum likelihood estimate
  - of some parameter of a model M from a training set T
  - maximizes the likelihood of the training set T given the model M
- Suppose the word “bagel” occurs 400 times in a corpus of a million words
- What is the probability that a random word from some other text will be “bagel”?
- MLE estimate is 400/1,000,000 = .0004
- This may be a bad estimate for some other corpus
  - But it is the **estimate** that makes it **most likely** that “bagel” will occur 400 times in a million word corpus.

Add-1 estimation is a blunt instrument

- So add-1 isn't used for N-grams:
  - We'll see better methods
- But add-1 is used to smooth other NLP models
  - For text classification
  - In domains where the number of zeros isn't so huge.

Language Modeling

Interpolation, Backoff, and Web-Scale LMs

Backoff and Interpolation

Sometimes it helps to use **less** context

- Condition on less context for contexts you haven't learned much about

Backoff:

- use trigram if you have good evidence,
- otherwise bigram, otherwise unigram

Interpolation:

- mix unigram, bigram, trigram

Interpolation works better

Linear Interpolation

Simple interpolation

$$\begin{aligned}\hat{P}(w_n|w_{n-2}w_{n-1}) &= \lambda_1 P(w_n|w_{n-2}w_{n-1}) \\ &\quad +\lambda_2 P(w_n|w_{n-1}) \\ &\quad +\lambda_3 P(w_n)\end{aligned}$$

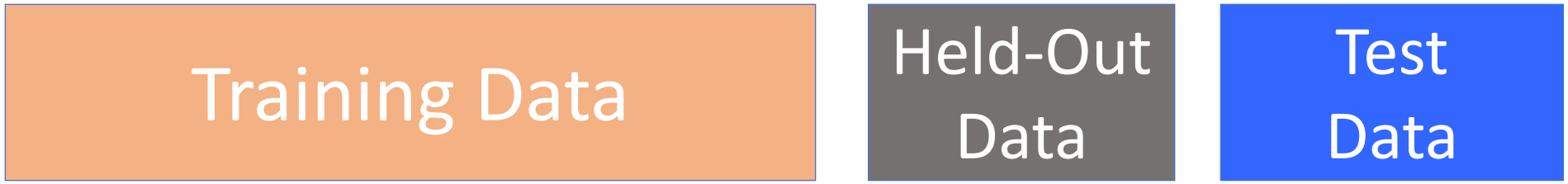
$$\sum_i \lambda_i = 1$$

Lambdas conditional on context:

$$\begin{aligned}\hat{P}(w_n|w_{n-2}w_{n-1}) &= \lambda_1(w_{n-2}^{n-1})P(w_n|w_{n-2}w_{n-1}) \\ &\quad +\lambda_2(w_{n-2}^{n-1})P(w_n|w_{n-1}) \\ &\quad +\lambda_3(w_{n-2}^{n-1})P(w_n)\end{aligned}$$

How to set the lambdas?

- Use a **held-out** corpus



- Choose λs to maximize the probability of held-out data:
  - Fix the N-gram probabilities (on the training data)
  - Then search for λs that give largest probability to held-out set:

$$\log P(w_1 \ldots w_n \mid M(\lambda_1 \ldots \lambda_k)) = \sum_i \log P_{M(\lambda_1 \ldots \lambda_k)}(w_i \mid w_{i-1})$$

Unknown words: Open versus closed vocabulary tasks

- If we know all the words in advanced
  - Vocabulary V is fixed
  - Closed vocabulary task
- Often we don't know this
  - **Out Of Vocabulary** = OOV words
  - Open vocabulary task
- Instead: create an unknown word token <UNK>
  - Training of <UNK> probabilities
    - Create a fixed lexicon L of size V
    - At text normalization phase, any training word not in L changed to <UNK>
    - Now we train its probabilities like a normal word
  - At decoding time
    - If text input: Use UNK probabilities for any word not in training

Huge web-scale n-grams

- How to deal with, e.g., Google N-gram corpus
- Pruning
  - Only store N-grams with count > threshold.
    - Remove singletons of higher-order n-grams
  - Entropy-based pruning
- Efficiency
  - Efficient data structures like tries
  - Bloom filters: approximate language models
  - Store words as indexes, not strings
    - Use Huffman coding to fit large numbers of words into two bytes
  - Quantize probabilities (4-8 bits instead of 8-byte float)

Smoothing for Web-scale N-grams

- "Stupid backoff" (Brants *et al*. 2007)
- No discounting, just use relative frequencies

$$S(w_i \mid w_{i-k+1}^{i-1}) = \begin{cases} \dfrac{\text{count}(w_{i-k+1}^{i})}{\text{count}(w_{i-k+1}^{i-1})} & \text{if} \quad \text{count}(w_{i-k+1}^{i}) > 0 \\ 0.4 S(w_i \mid w_{i-k+2}^{i-1}) & \text{otherwise} \end{cases}$$

$$S(w_i) = \frac{\text{count}(w_i)}{N}$$

N-gram Smoothing Summary

- Add-1 smoothing:
  - OK for text categorization, not for language modeling
- The most commonly used method:
  - Extended Interpolated Kneser-Ney
- For very large N-grams like the Web:
  - Stupid backoff

Advanced Language Modeling

- Discriminative models:
  - choose n-gram weights to improve a task, not to fit the training set
- Parsing-based models
- Caching Models
  - Recently used words are more likely to appear

$$P_{CACHE}(w \mid history) = \lambda P(w_i \mid w_{i-2} w_{i-1}) + (1-\lambda)\frac{c(w \in history)}{\mid history \mid}$$

Language Modeling

Advanced:

Kneser-Ney Smoothing

Absolute discounting: just subtract a little from each count

- Suppose we wanted to subtract a little from a count of 4 to save probability mass for the zeros
- How much to subtract ?

- Church and Gale (1991)'s clever idea
- Divide up 22 million words of AP Newswire
  - Training and held-out set
  - for each bigram in the training set
  - see the actual count in the held-out set!

- It sure looks like c* = (c - .75)

Bigram count in training	Bigram count in heldout set
0	.0000270
1	0.448
2	1.25
3	2.24
4	3.23
5	4.21
6	5.23
7	6.21
8	7.21
9	8.26

Absolute Discounting Interpolation

- Save ourselves some time and just subtract 0.75 (or some d)!

discounted bigram

Interpolation weight

$$P_{\text{AbsoluteDiscounting}}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i) - d}{c(w_{i-1})} + \lambda(w_{i-1})P(w)$$

unigram

- (Maybe keeping a couple extra values of d for counts 1 and 2)
- But should we really just use the regular unigram P(w)?

Kneser-Ney Smoothing I

- Better estimate for probabilities of lower-order unigrams!
  - Shannon game: *I can't see without my reading*_____? Francisco glasses
  - "Francisco" is more common than "glasses"
  - … but "Francisco" always follows "San"
- The unigram is useful exactly when we haven't seen this bigram!
- Instead of P(w): "How likely is w"
- $P_{continuation}$(w): "How likely is w to appear as a novel continuation?
  - For each word, count the number of bigram types it completes
  - Every bigram type was a novel continuation the first time it was seen

$$P_{CONTINUATION}(w) \propto \left|\{w_{i-1} : c(w_{i-1}, w) > 0\}\right|$$

Kneser-Ney Smoothing II

- How many times does w appear as a novel continuation:

$$P_{CONTINUATION}(w) \propto \left|\{w_{i-1} : c(w_{i-1}, w) > 0\}\right|$$

- Normalized by the total number of word bigram types

$$\left|\{(w_{j-1}, w_j) : c(w_{j-1}, w_j) > 0\}\right|$$

$$P_{CONTINUATION}(w) = \frac{\left|\{w_{i-1} : c(w_{i-1}, w) > 0\}\right|}{\left|\{(w_{j-1}, w_j) : c(w_{j-1}, w_j) > 0\}\right|}$$

Kneser-Ney Smoothing III

- Alternative metaphor: The number of # of word types seen to precede w

$$|\{w_{i-1} : c(w_{i-1}, w) > 0\}|$$

- normalized by the # of words preceding all words:

$$P_{CONTINUATION}(w) = \frac{\left|\{w_{i-1} : c(w_{i-1}, w) > 0\}\right|}{\sum_{w'} \left|\{w'_{i-1} : c(w'_{i-1}, w') > 0\}\right|}$$

- A frequent word (Francisco) occurring in only one context (San) will have a low continuation probability

Kneser-Ney Smoothing IV

$$P_{KN}(w_i \mid w_{i-1}) = \frac{\max(c(w_{i-1}, w_i) - d, 0)}{c(w_{i-1})} + \lambda(w_{i-1}) P_{CONTINUATION}(w_i)$$

λ is a normalizing constant; the probability mass we've discounted

$$\lambda(w_{i-1}) = \frac{d}{c(w_{i-1})} \left| \{w : c(w_{i-1}, w) > 0\} \right|$$

the normalized discount

The number of word types that can follow w_{i-1}
= # of word types we discounted
= # of times we applied normalized discount

Kneser-Ney Smoothing: Recursive formulation

$$P_{KN}(w_i \mid w_{i-n+1}^{i-1}) = \frac{\max(c_{KN}(w_{i-n+1}^{i}) - d, 0)}{c_{KN}(w_{i-n+1}^{i-1})} + \lambda(w_{i-n+1}^{i-1}) P_{KN}(w_i \mid w_{i-n+2}^{i-1})$$

$$c_{KN}(\bullet) = \begin{cases} count(\bullet) & \text{for the highest order} \\ continuationcount(\bullet) & \text{for lower order} \end{cases}$$

Continuation count = Number of unique single word contexts for •

Vector Space Semantics

Read Chapter 6 in the draft 3rd edition of Jurafsky and Martin

What does ongchoi mean?

Suppose you see these sentences:

Ong choi is delicious **sautéed with garlic**.

Ong choi is superb **over rice**

Ong choi **leaves** with salty sauces

And you've also seen these:

…spinach **sautéed with garlic over rice**

Chard stems and **leaves** are **delicious**

Collard greens and other **salty** leafy greens

Conclusion:

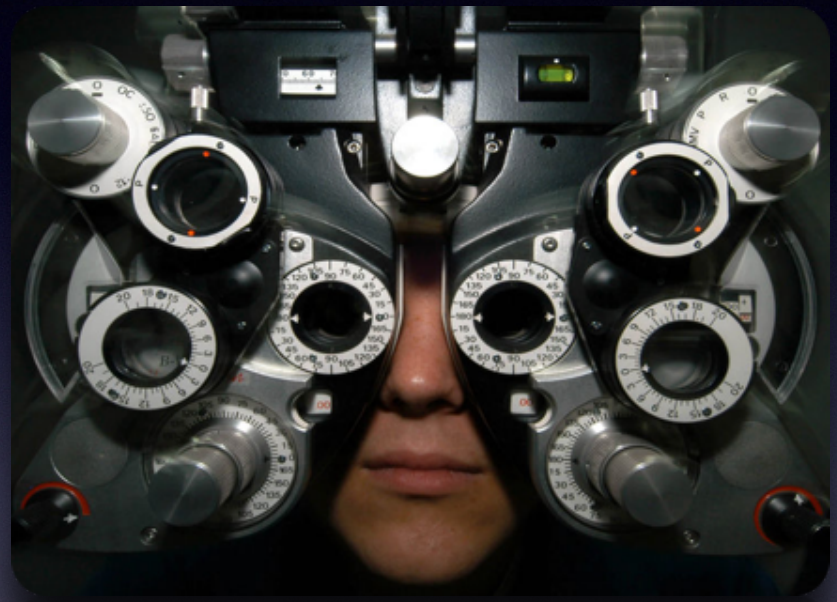
Ongchoi is a leafy green like spinach, chard, or collard greens

Ong choi: *Ipomoea aquatica "Water Spinach"*

Yamaguchi, Wikimedia Commons, public domain

Distributional Hypothesis

If we consider **optometrist** and **eye-doctor** we find that, as our corpus of utterances grows, these two occur in almost the same environments. In contrast, there are many sentence environments in which **optometrist** occurs but **lawyer** does not...



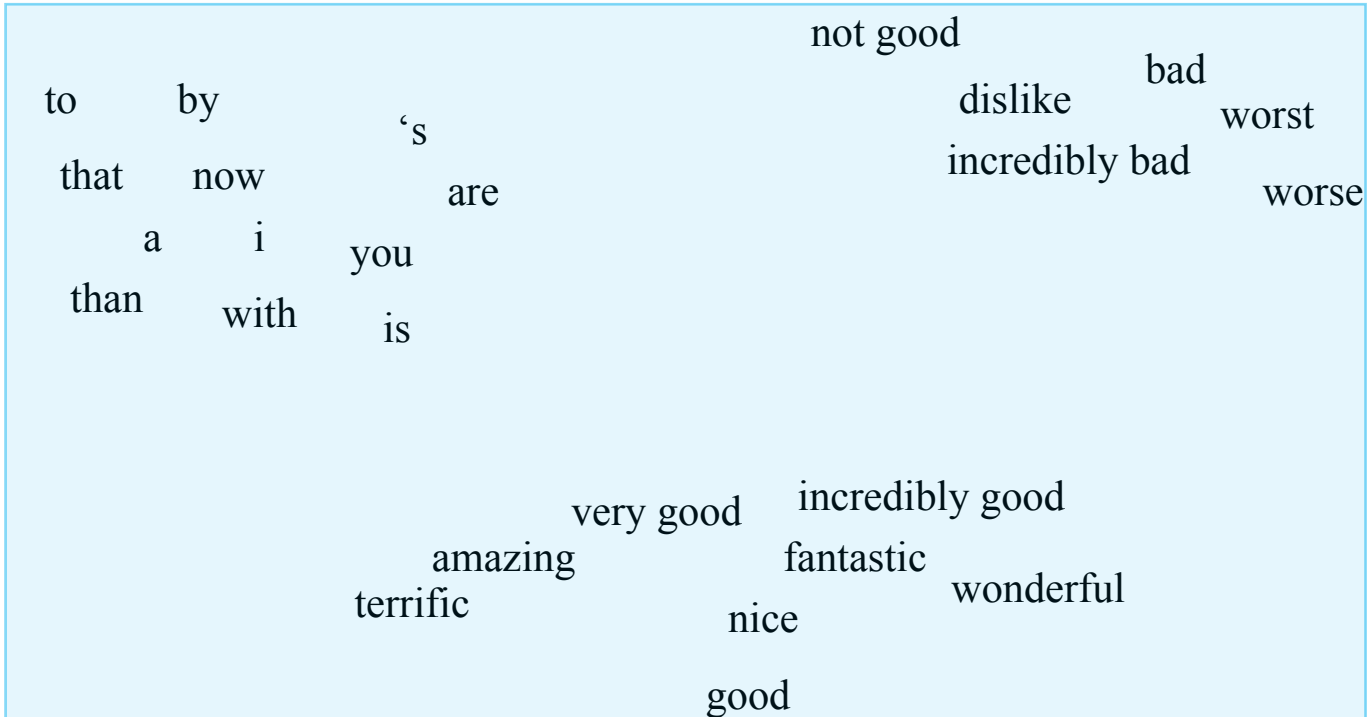
It is a question of the relative frequency of such environments, and of what we will obtain if we ask an informant to substitute any word he wishes for **oculist** (not asking what words have the same meaning).

These and similar tests all measure the probability of particular environments occurring with particular elements... If A and B have almost identical environments we say that they are synonyms.

–Zellig Harris (1954)

We'll build a new representation of words that encodes their similarity

- Each word = a vector
- Similar words are "nearby in space"



We define a word as a vector

- Called an "embedding" because it's embedded into a space
- The standard way to represent meaning in NLP
- Fine-grained model of meaning for similarity
  - NLP tasks like sentiment analysis
    - With words, requires **same** word to be in training and test
    - With embeddings: ok if **similar** words occurred!!!
  - Question answering, conversational agents, etc

We'll introduce 2 kinds of embeddings

- Tf-idf
  - A common baseline model
  - Sparse vectors
  - Words are represented by a simple function of the counts of nearby words
- Word2vec
  - Dense vectors
  - Representation is created by training a classifier to distinguish nearby and far-away words

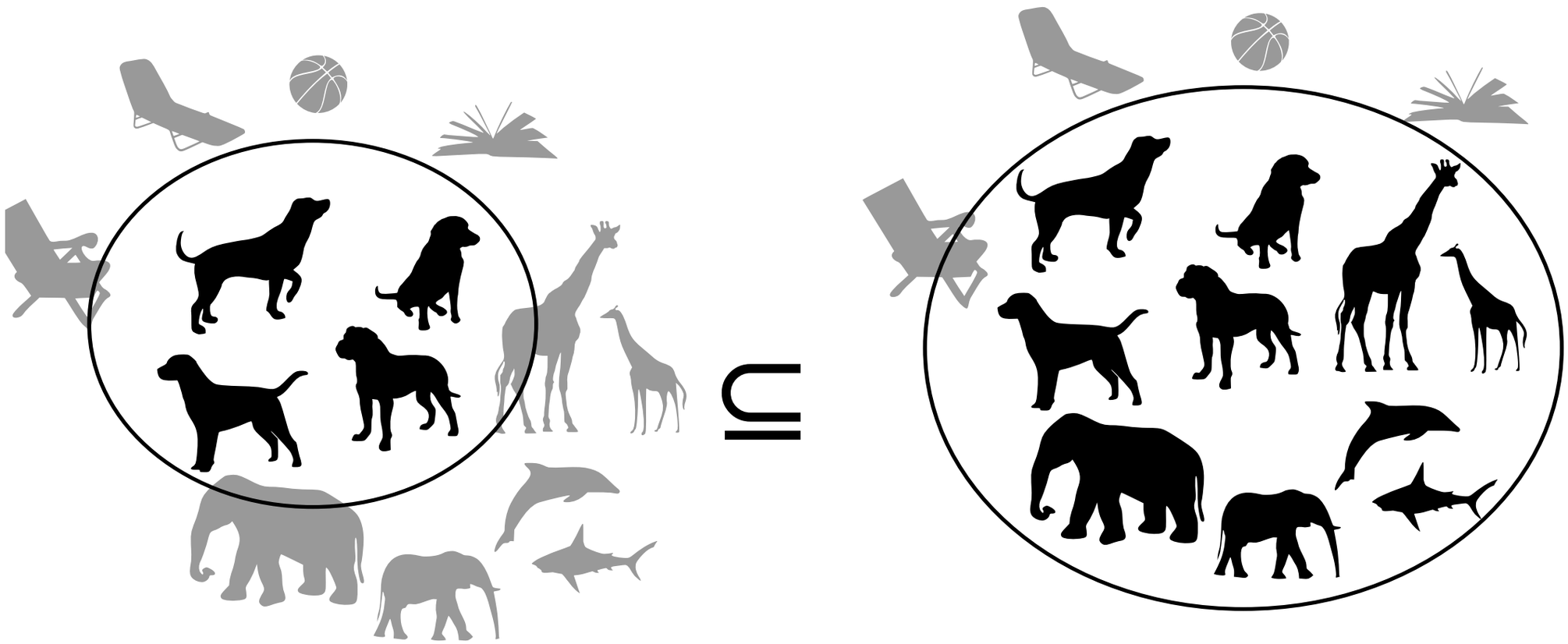
Representing Word Meaning with Vectors

Chris Callison-Burch
University of Pennsylvania
June 5, 2019

https://cis.upenn.edu/~ccb/

Representing Word Meaning with Vectors

Entailment in formal semantics



Dog ⊆ Animal

Entailment in formal semantics

All animals have an ulnar artery

⇒

All dogs have an ulnar artery

+ Mathematically well-understood
+ Powerful machinery for handling logical operations
- Knowledge must come from somewhere else

WordNet Search - 3.1

- WordNet home page - Glossary - Help

Word to search for: dog 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) **dog**, domestic dog, Canis familiaris (a member of the genus Canis (probably descended from the common wolf) that has been domesticated by man since prehistoric times; occurs in many breeds) *"the dog barked all night"*
- S: (n) frump, **dog** (a dull unattractive unpleasant girl or woman) *"she got a reputation as a frump"; "she's a real dog"*
- S: (n) **dog** (informal term for a man) *"you lucky dog"*
- S: (n) cad, bounder, blackguard, **dog**, hound, heel (someone who is morally reprehensible) *"you dirty dog"*
- S: (n) frank, frankfurter, hotdog, hot dog, **dog**, wiener, wienerwurst, weenie (a smooth-textured sausage of minced beef or pork usually smoked; often served on a bread roll)
- S: (n) pawl, detent, click, **dog** (a hinged catch that fits into a notch of a ratchet to move a wheel forward or prevent it from moving backward)
- S: (n) andiron, firedog, **dog**, dog-iron (metal supports for logs in a fireplace) *"the andirons were too hot to touch"*

Verb

- S: (v) chase, chase after, trail, tail, tag, give chase, **dog**, go after, track (go after with the intent to catch) *"The policeman chased the mugger down the alley"; "the dog chased the rabbit"*

WordNet Search - 3.1

- WordNet home page - Glossary - Help

Word to search for: dog 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) **dog**, domestic dog, Canis familiaris (a member of the genus Canis (probably descended from the common wolf) that has been domesticated by man since prehistoric times; occurs in many breeds) *"the dog barked all night"*
  - *direct hyponym* / *full hyponym*
  - *part meronym*
  - *member holonym*
  - ***direct hypernym*** / *inherited hypernym* / *sister term*
    - S: (n) canine, canid (any of various fissiped mammals with nonretractile claws and typically long muzzles)
    - S: (n) domestic animal, domesticated animal (any of various animals that have been tamed and made fit for a human environment)
- S: (n) frump, **dog** (a dull unattractive unpleasant girl or woman) *"she got a reputation as a frump"; "she's a real dog"*
- S: (n) **dog** (informal term for a man) *"you lucky dog"*
- S: (n) cad, bounder, blackguard, **dog**, hound, heel (someone who is morally reprehensible) *"you dirty dog"*
- S: (n) frank, frankfurter, hotdog, hot dog, **dog**, wiener, wienerwurst, weenie (a smooth-textured sausage of minced beef or pork usually smoked; often served on a bread roll)
- S: (n) pawl, detent, click, **dog** (a hinged catch that fits into a notch of a ratchet to move a wheel forward or prevent it from moving backward)
- S: (n) andiron, firedog, **dog**, dog-iron (metal supports for logs in a fireplace)

Word to search for: dog Search WordNet

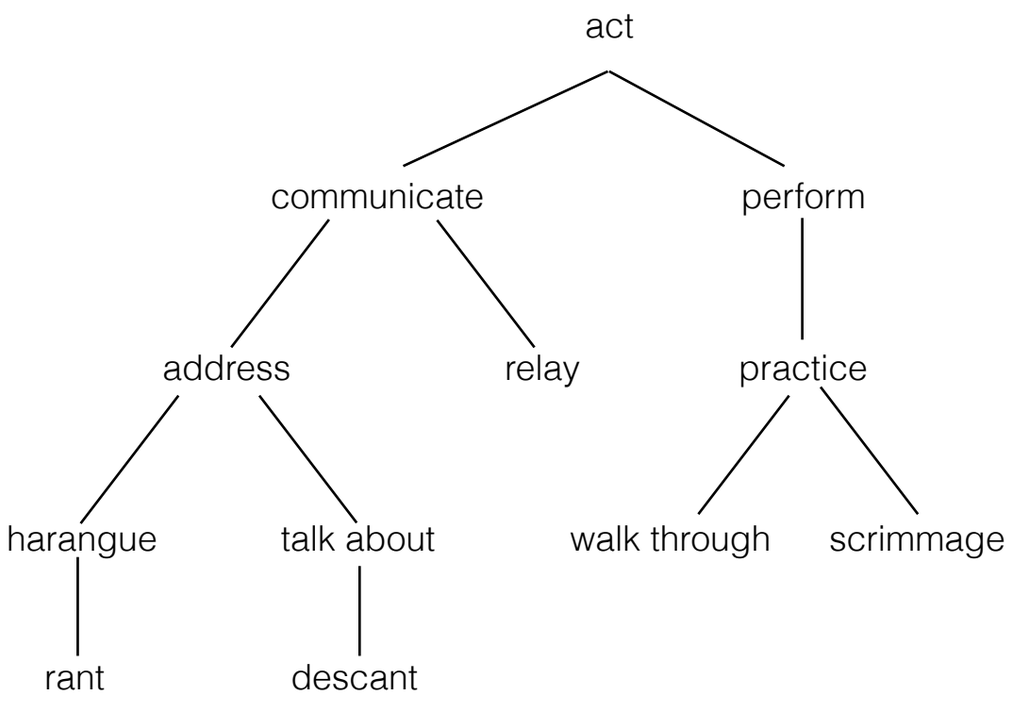
Display Options: (Select option to change) Change

Key: "S:" = Show Synset (semantic) relations, "W:" = Show Word (lexical) relations
Display options for sense: (gloss) "an example sentence"

Noun

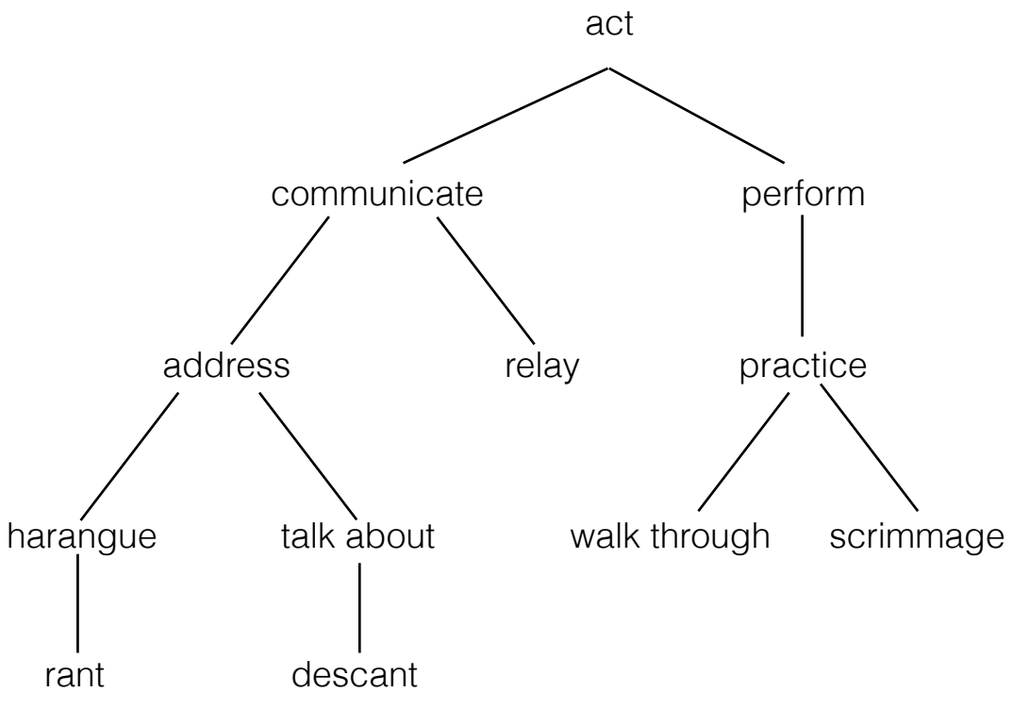
- S: (n) **dog**, domestic dog, Canis familiaris (a member of the genus Canis (probably descended from the common wolf) that has been domesticated by man since prehistoric times; occurs in many breeds) *"the dog barked all night"*
  - *direct hyponym* / *full hyponym*
  - *part meronym*
  - *member holonym*
  - *direct hypernym* / ***inherited hypernym*** / *sister term*
    - S: (n) canine, canid (any of various fissiped mammals with nonretractile claws and typically long muzzles)
      - S: (n) carnivore (a terrestrial or aquatic flesh-eating mammal) *"terrestrial carnivores have four or five clawed digits on each limb"*
        - S: (n) placental, placental mammal, eutherian, eutherian mammal (mammals having a placenta; all mammals except monotremes and marsupials)
          - S: (n) mammal, mammalian (any warm-blooded vertebrate having the skin more or less covered with hair; young are born alive except for the small subclass of monotremes and nourished with milk)
            - S: (n) vertebrate, craniate (animals having a bony or cartilaginous skeleton with a segmented spinal column and a large brain enclosed in a skull or cranium)
              - S: (n) chordate (any animal of the phylum Chordata having a notochord or spinal column)
                - S: (n) animal, animate being, beast, brute, creature, fauna (a living

Lexical Semantics

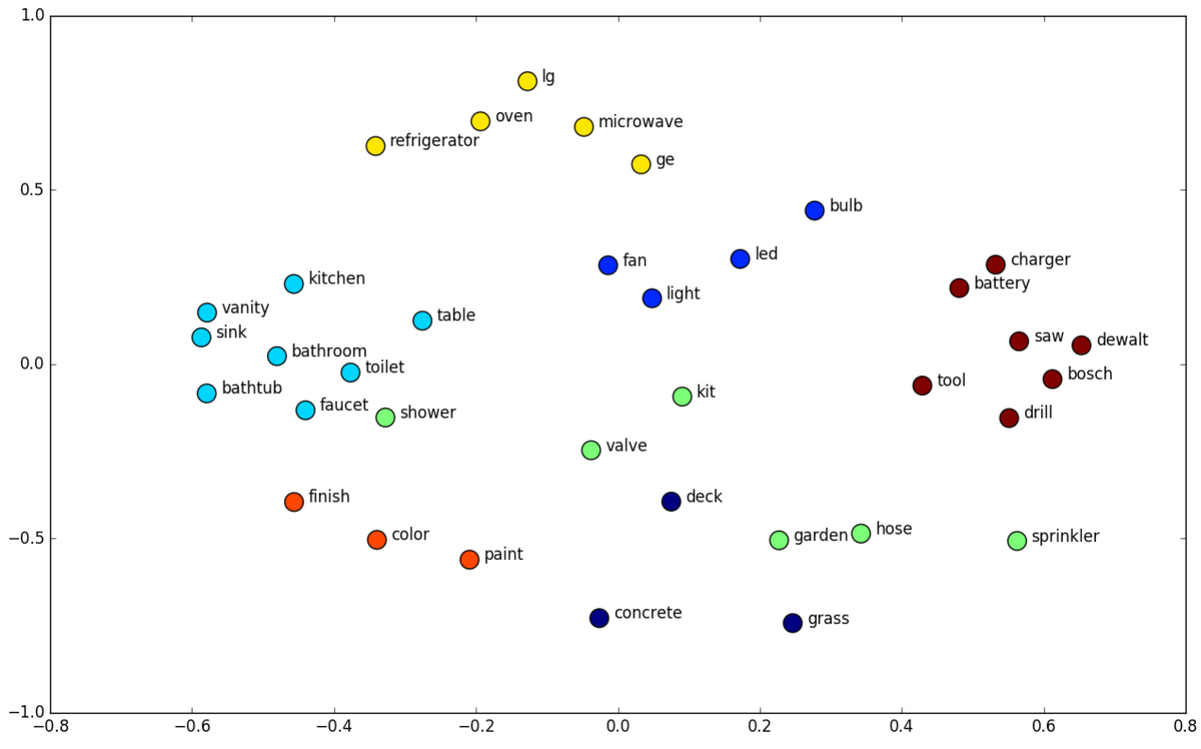


WordNet

Lexical Semantics

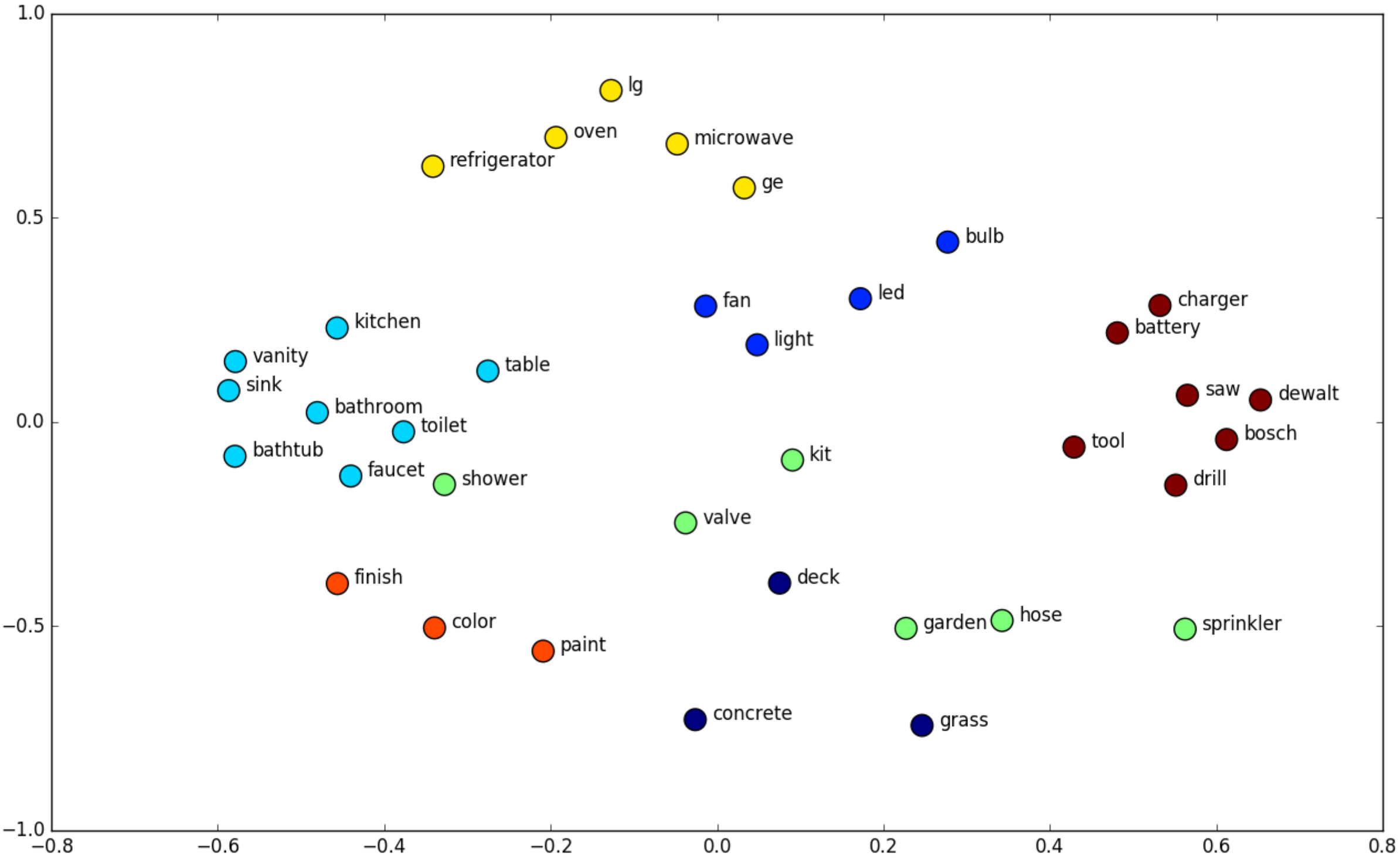


WordNet



Vector Space Models

Vector Space Models



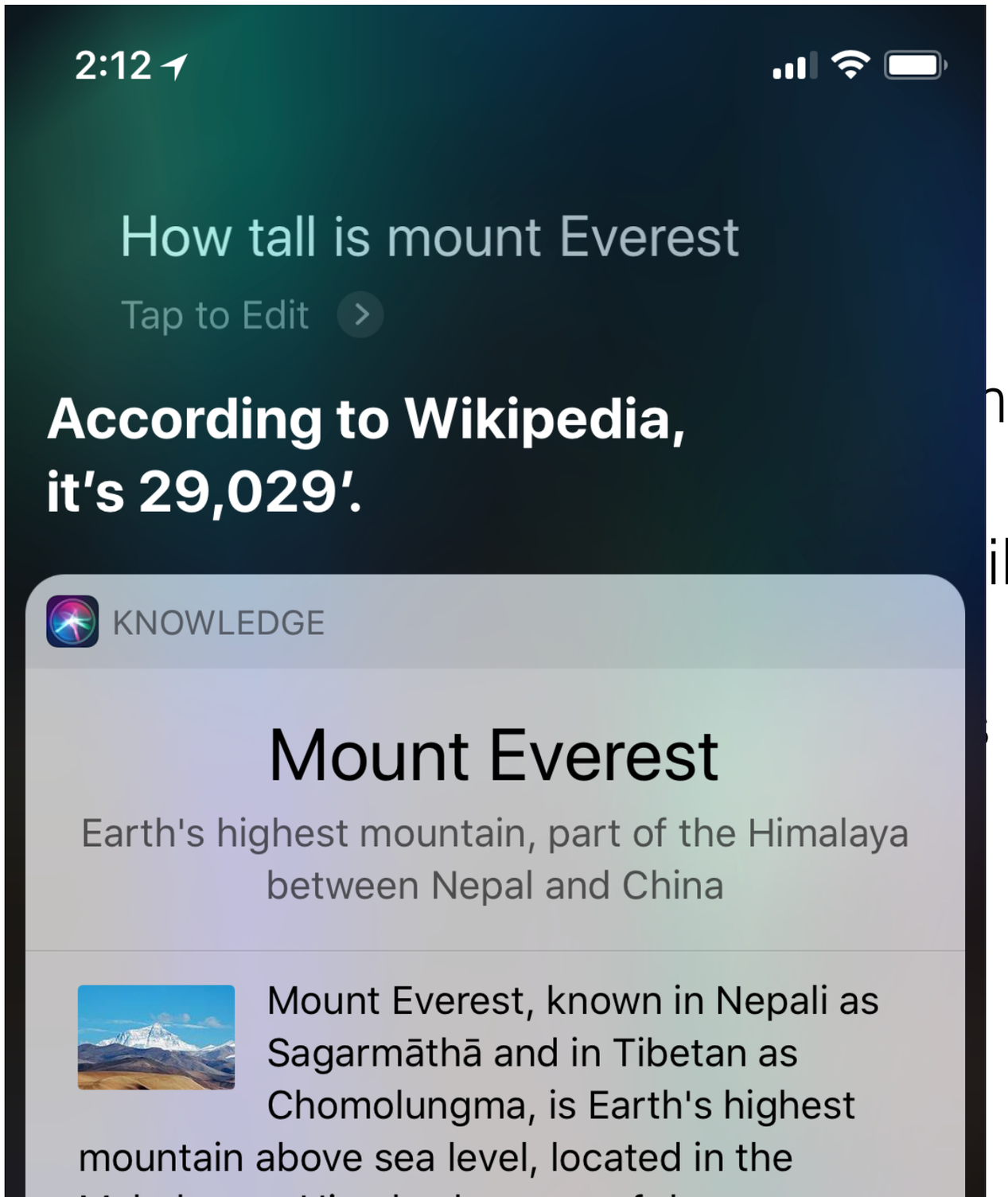
Why vector models of meaning? computing the similarity between words

"**fast**" is similar to "**rapid**"

"**tall**" is similar to "**height**"

Useful for applications like Question Answering

Why vector models of meaning? computing the similarity between words

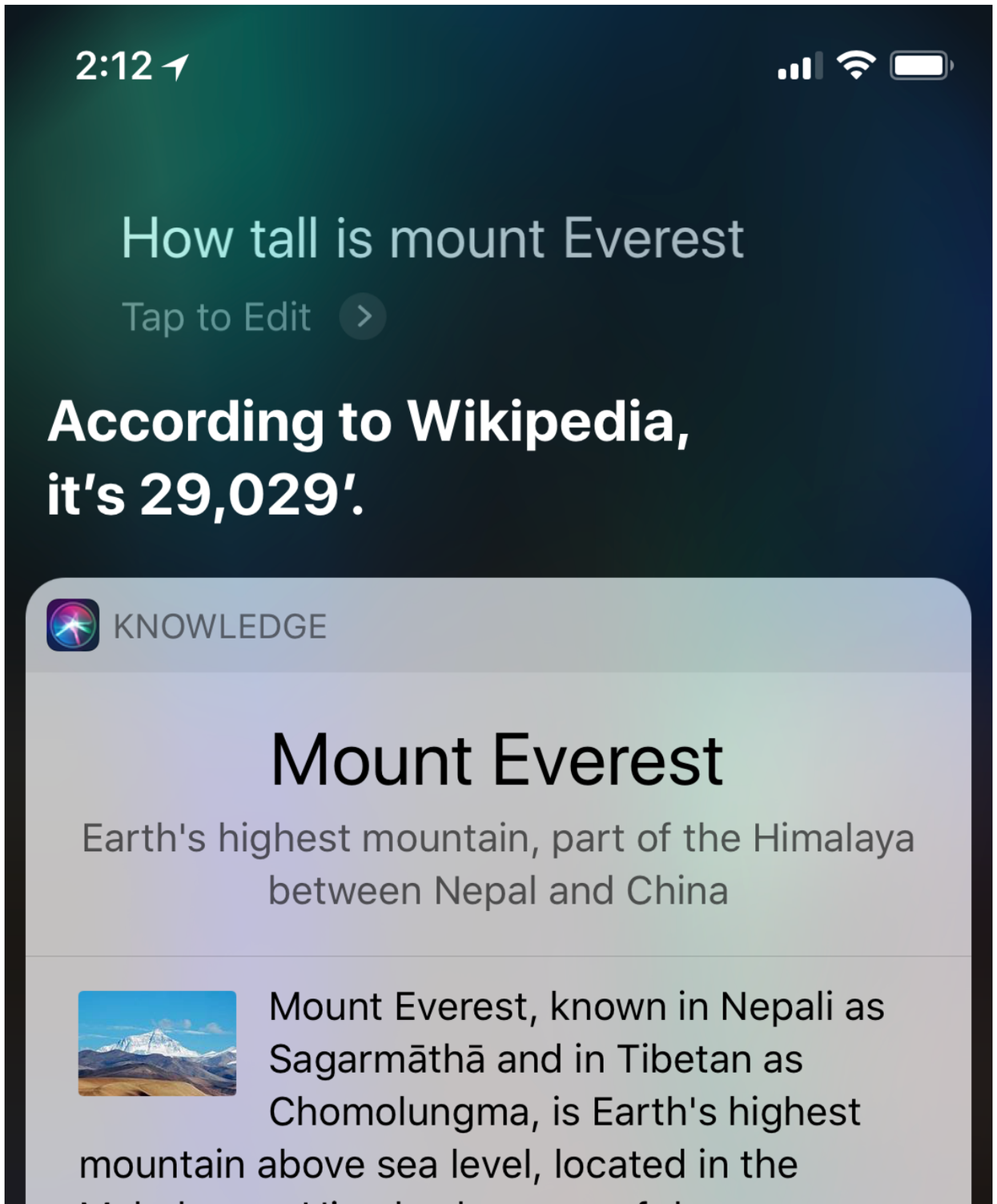


nilar to "**rapid**"

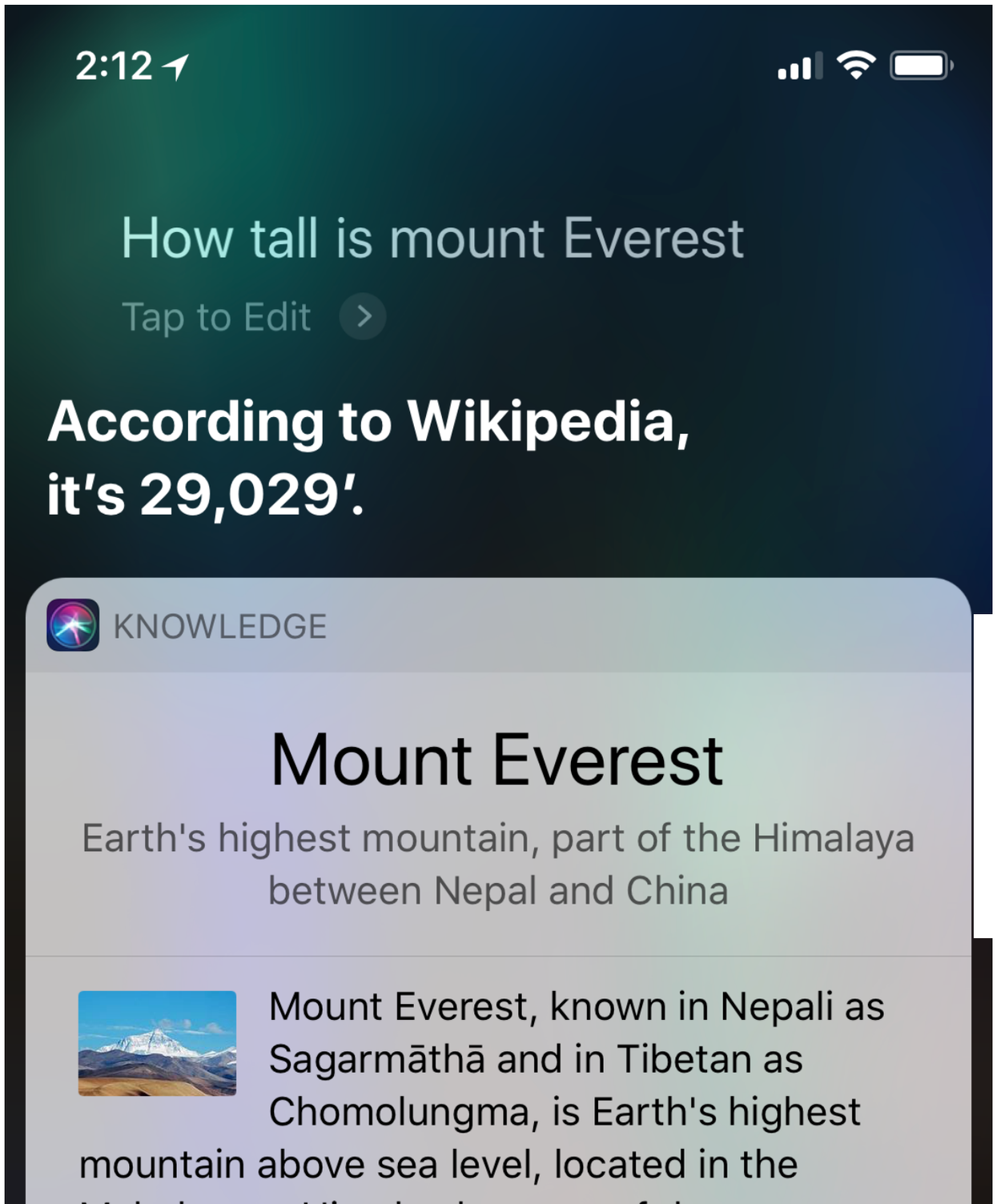
ilar to "**height**"

like Question Answering

Why vector models of meaning? computing the similarity between words



Why vector models of meaning? computing the similarity between words



Intuition of distributional word similarity

- Nida (1975) example:

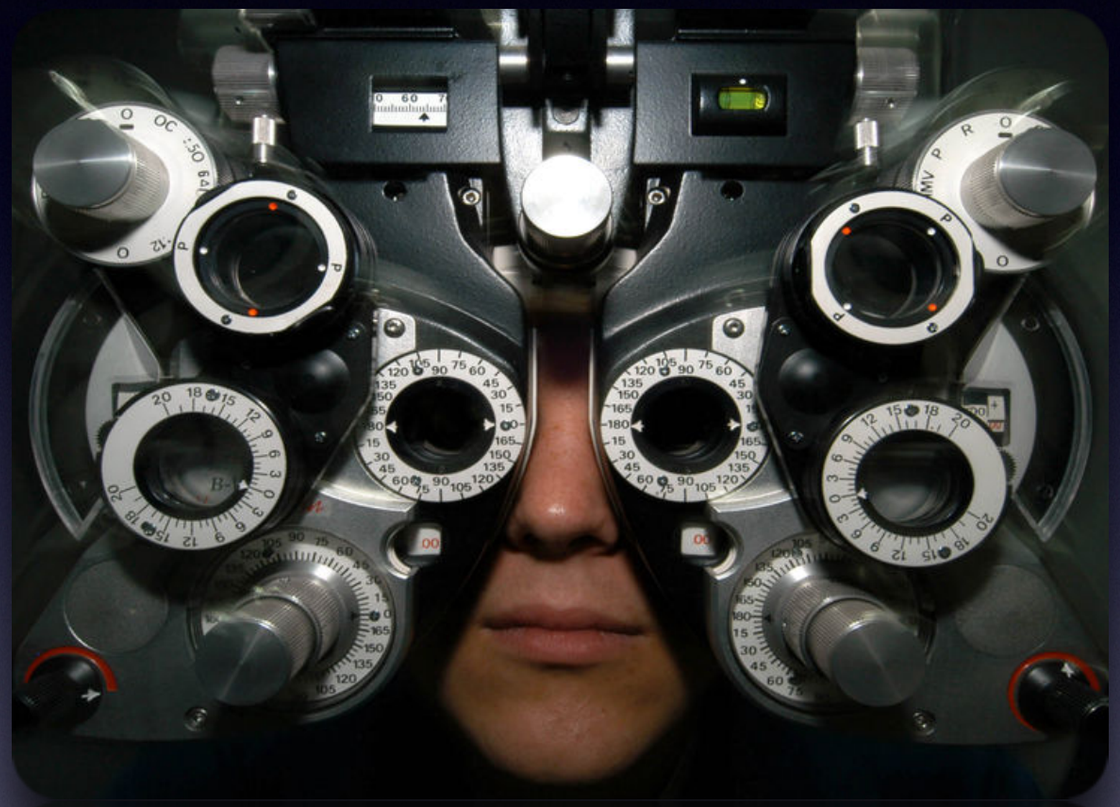
A bottle of **tesgüino** is on the table
Everybody likes **tesgüino**
Tesgüino makes you drunk
We make **tesgüino** out of corn.

- From context words humans can guess **tesgüino** means *an alcoholic beverage like beer*

- Intuition for algorithm:
  Two words are similar if they have similar word contexts.

Distributional Hypothesis

If we consider **optometrist** and **eye-doctor** we find that, as our corpus of utterances grows, these two occur in almost the same environments. In contrast, there are many sentence environments in which **optometrist** occurs but **lawyer** does not...



It is a question of the relative frequency of such environments, and of what we will obtain if we ask an informant to substitute any word he wishes for **oculist** (not asking what words have the same meaning).

These and similar tests all measure the probability of particular environments occurring with particular elements... If A and B have almost identical environments we say that they are synonyms.

–Zellig Harris (1954)

Information Retrieval

- Vector Space Models were initially developed in the SMART information retrieval system (Salton, 1971)

- Each document in a collection is represented as point in a space (a vector in a vector space)

- A user's query is a pseudo-document and is represented as a point in the same space as the documents

- Perform IR by retrieving documents whose vectors are close together in this space to the query vector

Term-Document Matrix

	D1	D2	D3	D4	D5
abandon					
abdicate					
abhor					
academic					
...					
zygodactyl					
zymurgy					

Term-Document Matrix

	D1	D2	D3	D4	D5
abandon					
abdicate					
abhor					
academic					
...					
zygodactyl					
zymurgy					

Each column vector represents a Document

Term-Document Matrix

	D1	D2	D3	D4	D5
abandon					
abdicate					
abhor					
academic					
...					
zygodactyl					
zymurgy					

Each row vector represents a Term

Term-Document Matrix

	D1	D2	D3	D4	D5
abandon					
abdicate		3			
abhor					
academic					
...					
zygodactyl					
zymurgy					

The value in a cell is based on how often that term occurred in that document

Term-Document Matrix

	D1	D2	D3	D4	D5
abandon					
abdicate					
abhor					
academic					
...					
zygodactyl					
zymurgy					

The length of the document vectors is the size of the vocabulary

Term-Document Matrix

	D1	D2	D3	D4	D5
abandon		0			
abdicate		3			
abhor		0			
academic		0			
...		...			
zygodactyl		0			
zymurgy		0			

Document vectors can be sparse (most values are 0)

Term-Document Matrix

	D1	D2	D3	D4	D5
abandon					
abdicate					
abhor					
academic					
...					
zygodactyl					
zymurgy					

We can measure how similar two documents are by comparing their column vectors

What can document similarity let you do?

Word similarity for plagiarism detection

MAINFRAMES

Mainframes are primarily referred to large computers with rapid, advanced processing capabilities that can execute and perform tasks equivalent to many Personal Computers (PCs) machines networked together. It is characterized with high quantity Random Access Memory (RAM), very large secondary storage devices, and high-speed processors to cater for the needs of the computers under its service.

Consisting of advanced components, mainframes have the capability of running multiple large applications required by many and most enterprises and organizations. This is one of its advantages. Mainframes are also suitable to cater for those applications (programs) or files that are of very high demand by its users (clients).

MAINFRAMES

Mainframes usually are referred those computers with fast, advanced processing capabilities that could perform by itself tasks that may require a lot of Personal Computers (PC) Machines. Usually mainframes would have lots of RAMs, very large secondary storage devices, and very fast processors to cater for the needs of those computers under its service.

Due to the advanced components mainframes have, these computers have the capability of running multiple large applications required by most enterprises, which is one of its advantage. Mainframes are also suitable to cater for those applications or files that are of very large demand by its users (clients). Examples of

Term-Document Matrix

	D1	D2	D3	D4	D5
abandon					
abdicate					
abhor					
academic					
...					
zygodactyl					
zymurgy					

What does comparing two row vectors do?

Vector comparisons

	doc_X	doc_Y
A	2	4
B	10	15
C	14	10

Vector comparisons

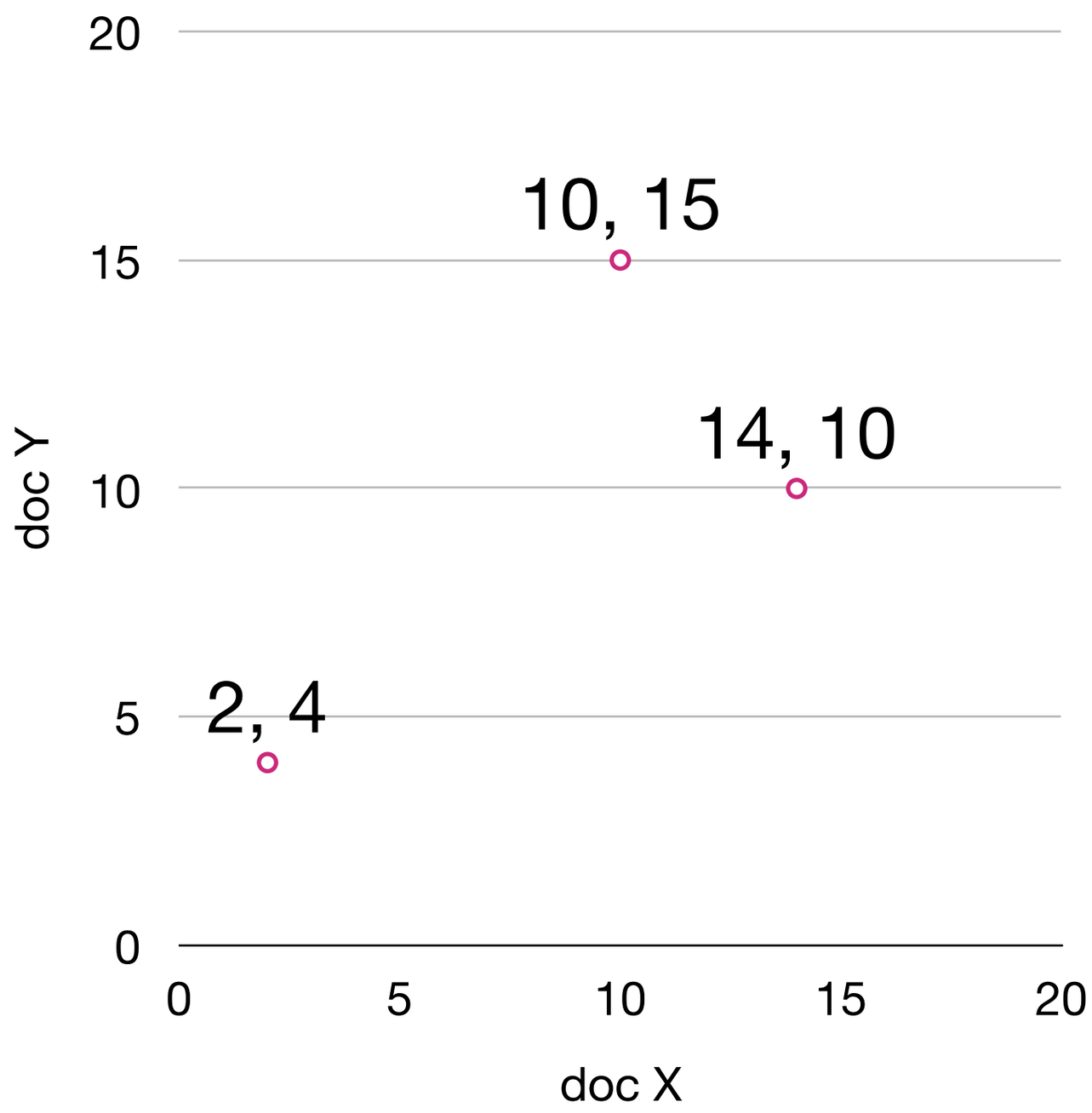
	doc_X	doc_Y
A	2	4
B	10	15
C	14	10

doc_Y is a positive movie review
doc_X is a less positive movie review

A = "superb" positive / low frequency
B = "good" positive / high frequency
C = "disappointing" negative / high frequency

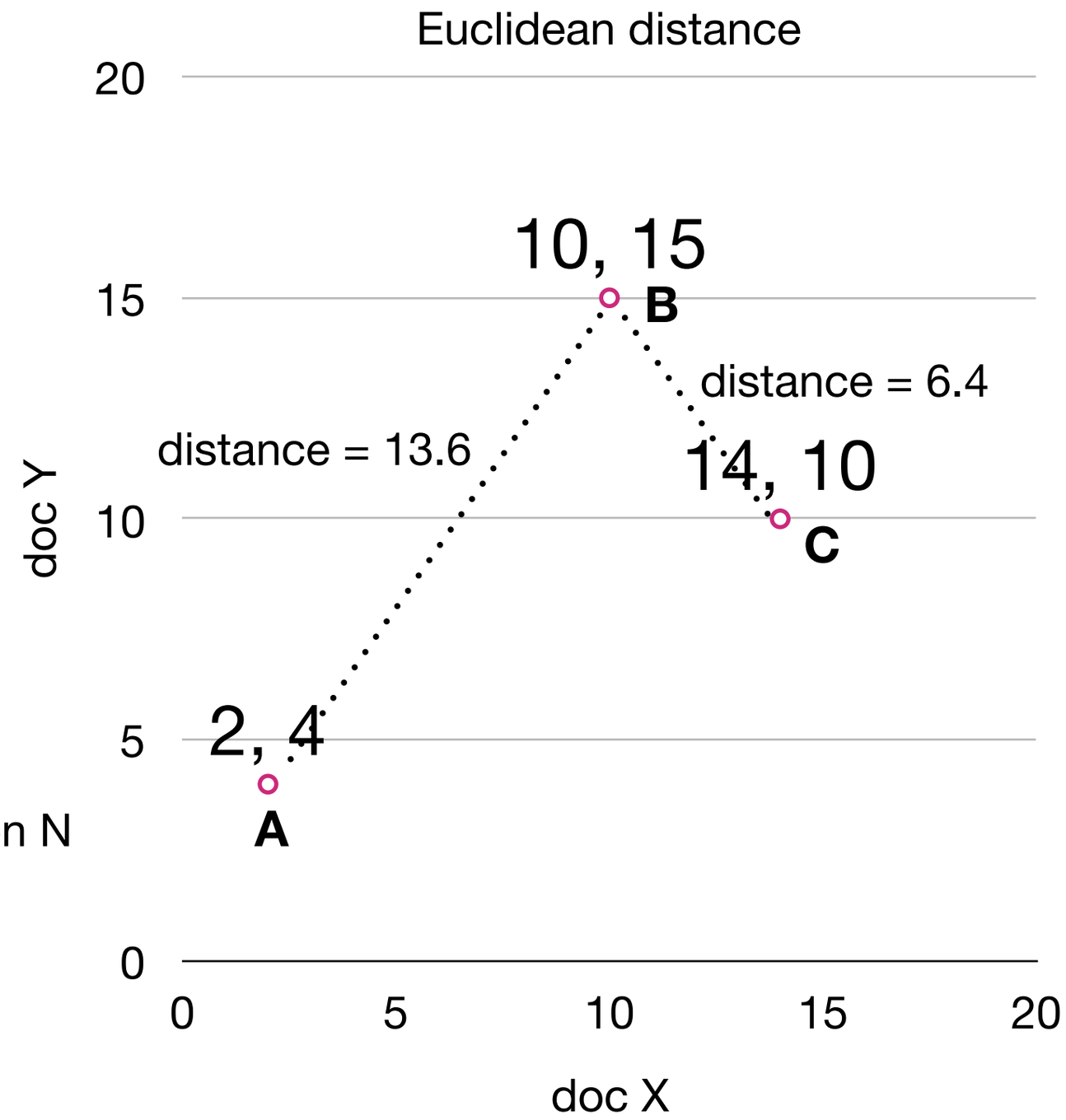
Vector comparisons

	doc_X	doc_Y
A	2	4
B	10	15
C	14	10



Vector comparisons

	doc_X	doc_Y
A	2	4
B	10	15
C	14	10

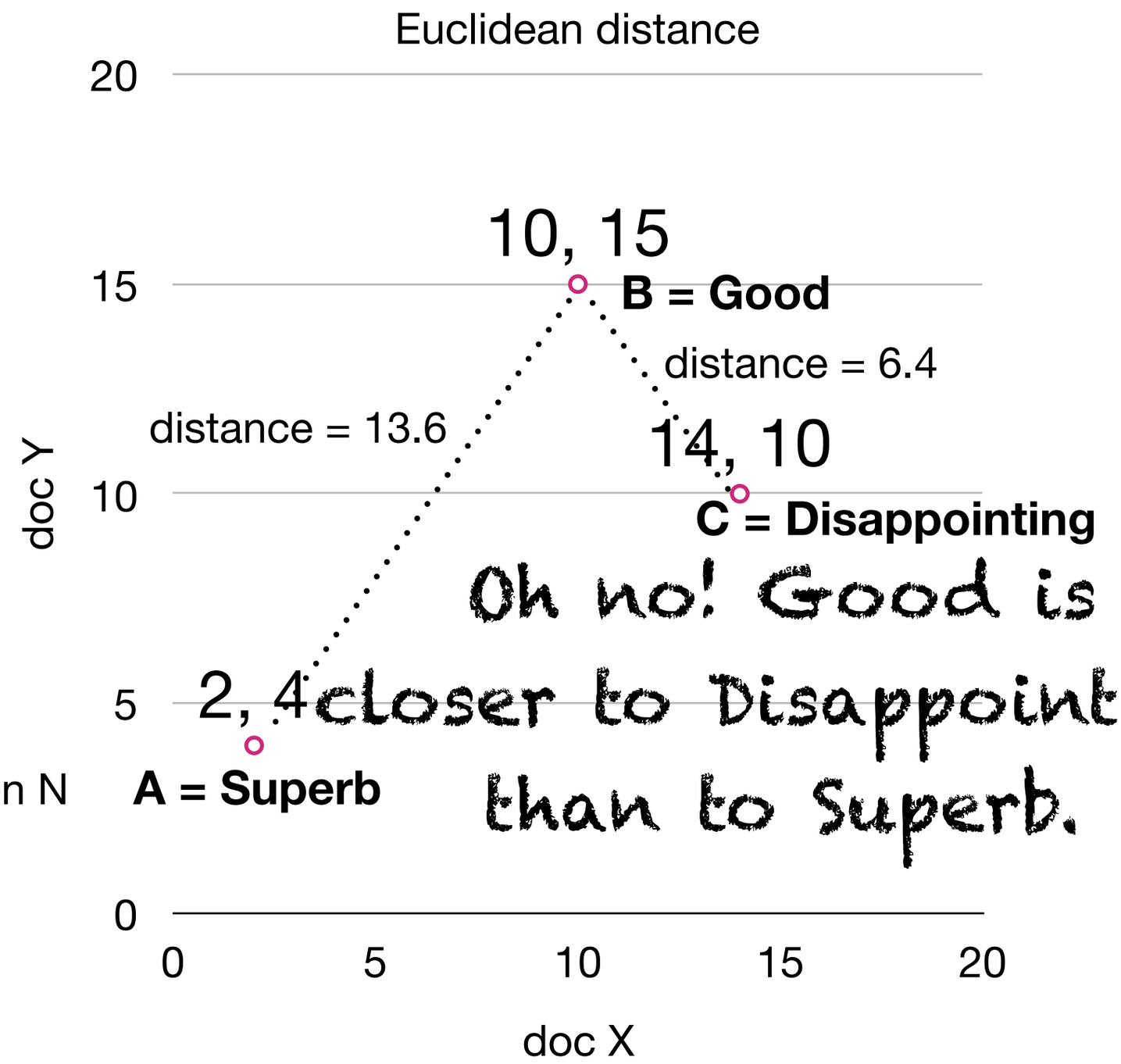


Euclidean distance : vectors u, v of dimension N

$$\sqrt{\sum_{i=1}^{N} |u_i - v_i|^2}$$

Vector comparisons

	doc_X	doc_Y
A	2	4
B	10	15
C	14	10



Euclidean distance : vectors u, v of dimension N

$$\sqrt{\sum_{i=1}^{N} |u_i - v_i|^2}$$

Vector L2 (length) Normalization

	doc_X	doc_Y	$\|u\|$
A	2	4	4.47
B	10	15	18.02
C	14	10	17.20

$$\|u\| = \sqrt{\sum_{i=1}^{n} u_i^2}$$

Vector L2 (length) Normalization

	doc_X	doc_Y	$\|u\|$
A	2/4.47	4/4.47	4.47
B	10/18.02	15/18.02	18.02
C	14/17.2	10/17.2	17.20

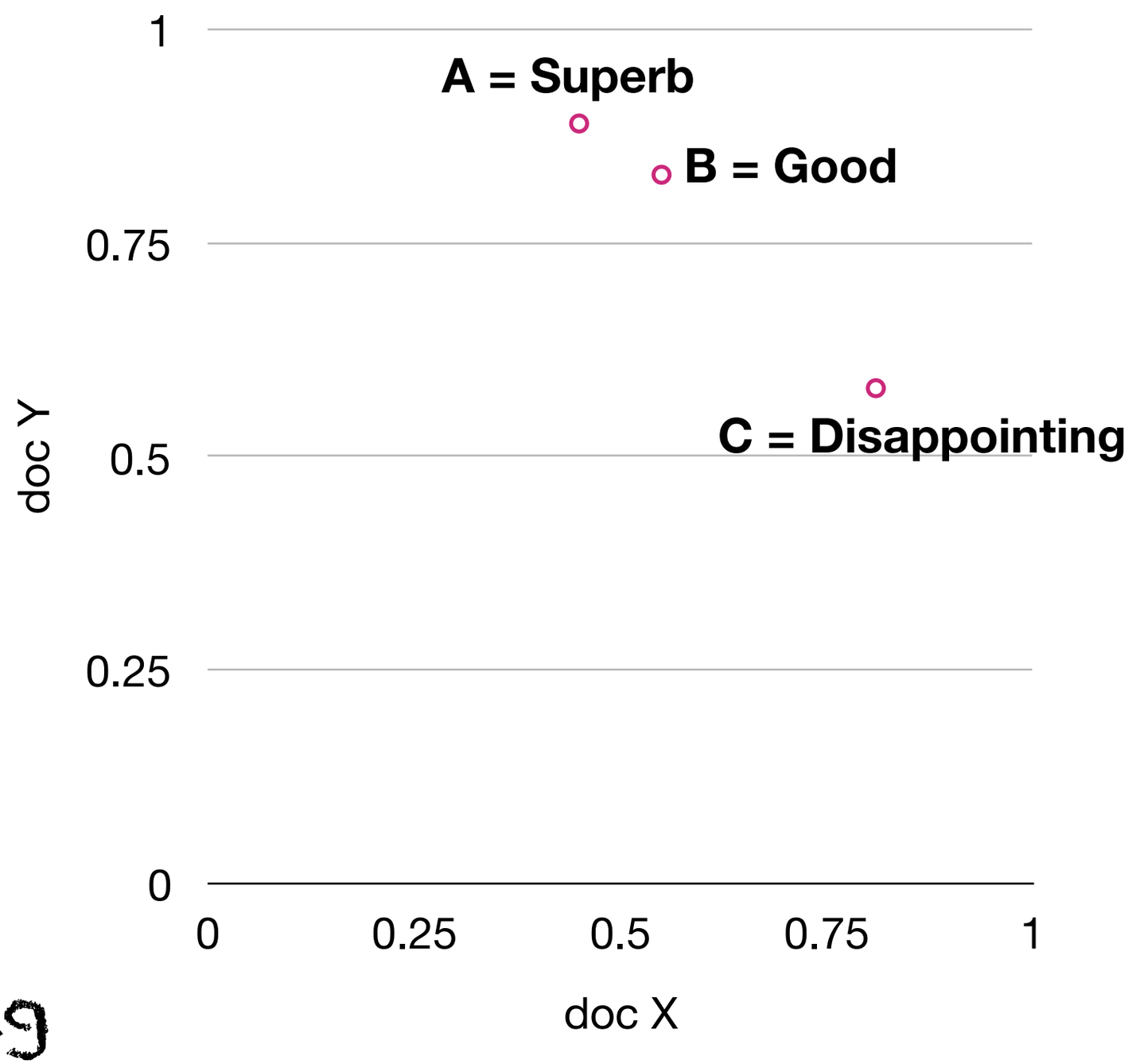
$$\|u\| = \sqrt{\sum_{i=1}^{n} u_i^2}$$

Divide each vector by its L2 length

Vector L2 (length) Normalization

	doc_X	doc_Y
Ȧ	0.45	0.89
Ḃ	0.55	0.83
Ċ	0.81	0.58

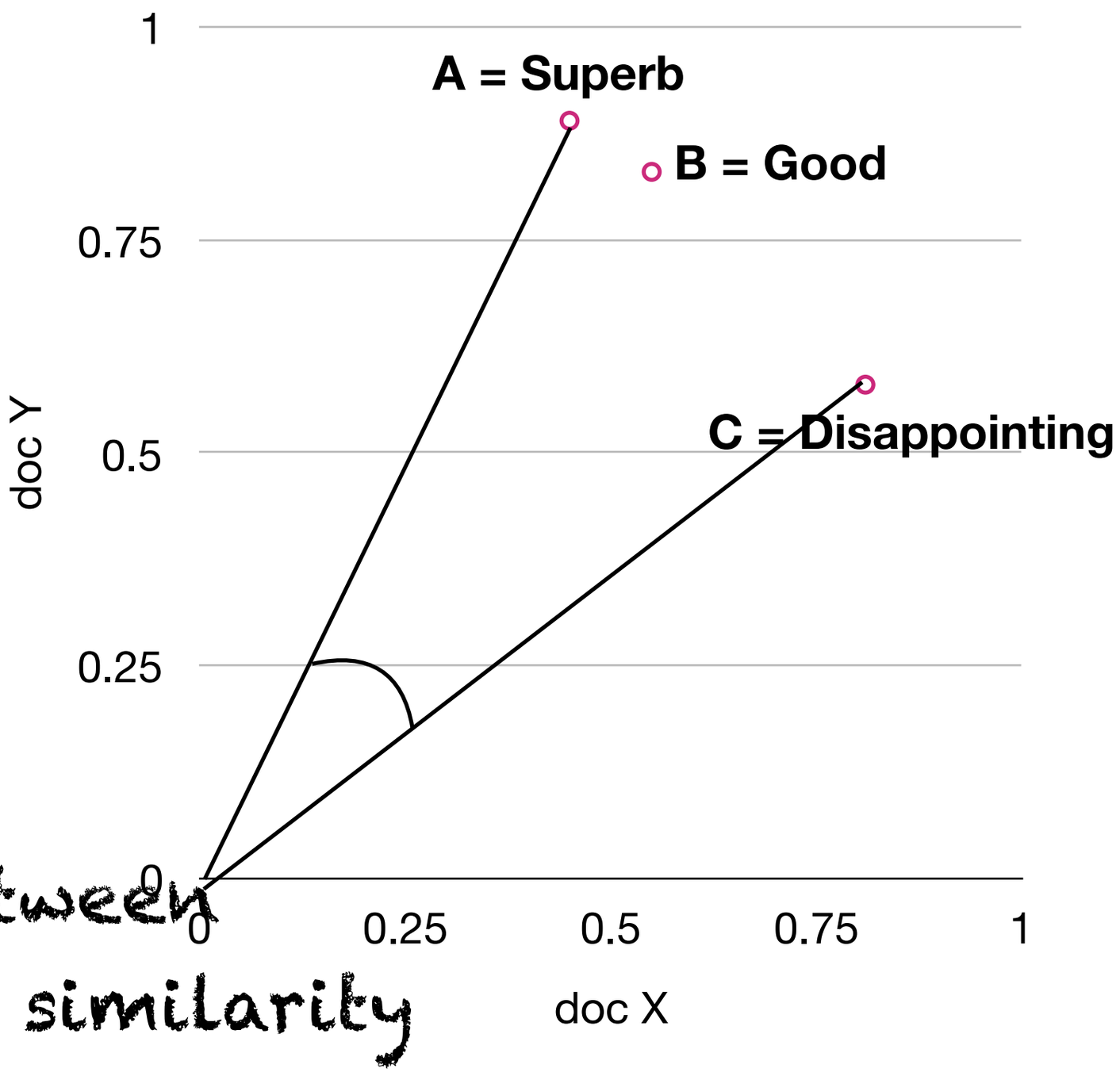
Now Good is closer to Superb than to Disappointing



Cosine Distance

$$1 - \frac{\sum_{i=1}^{n} u_i \times v_i}{\sqrt{\sum_{i=1}^{n} u_i^2} \times \sqrt{\sum_{i=1}^{n} v_i^2}}$$

Cosine does the L2 normalization too



Cosine angle between vectors tells us their similarity

Term-Term Matrix

	abandon	abdicate	abhor	…	zymurgy
abandon					
abdicate					
abhor					
academic					
…					
zygodactyl					
zymurgy					

Term-Term Matrix

AKA Term-Context Matrix

	abandon	abdicate	abhor	…	zymurgy
abandon					
abdicate					
abhor					
academic					
…					
zygodactyl					
zymurgy					

Length of the vector is now |V| instead of number of documents

Term-Term Matrix

AKA Term-Context Matrix

	abandon	abdicate	abhor	…	zymurgy
abandon					
abdicate					
abhor					
academic					
…					
zygodactyl					
zymurgy					

The value in a cell indicates how often abandon appears in a context window surrounding abdicate

Context windows

w-2, w-1 **target_word** w+1 w+2

The government most not **abdicate** responsibility to non-elected
it has led men to **abdicate** their family responsibilities
other demands, but declining to **abdicate** his responsibility
leaders **abdicate** their role and present people with no plans

	his	leaders	not	responsibility	to
abandon	1	1	1	2	3

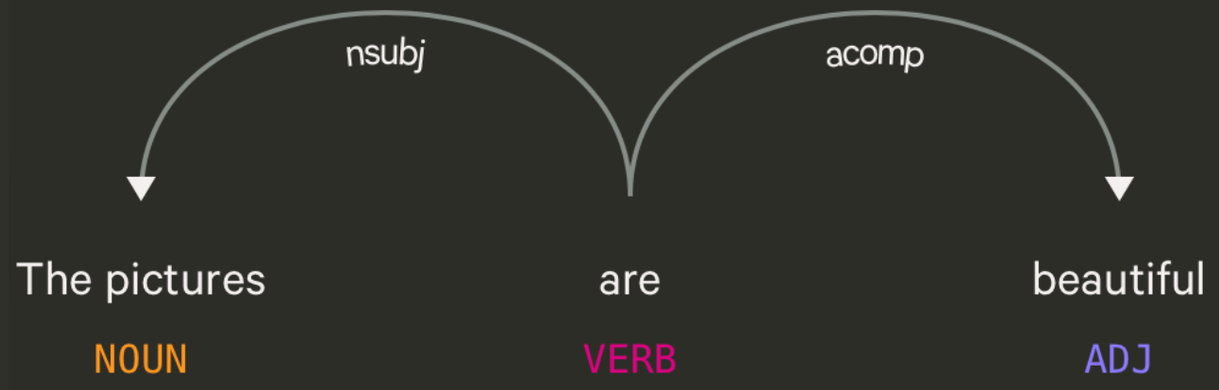
Context windows

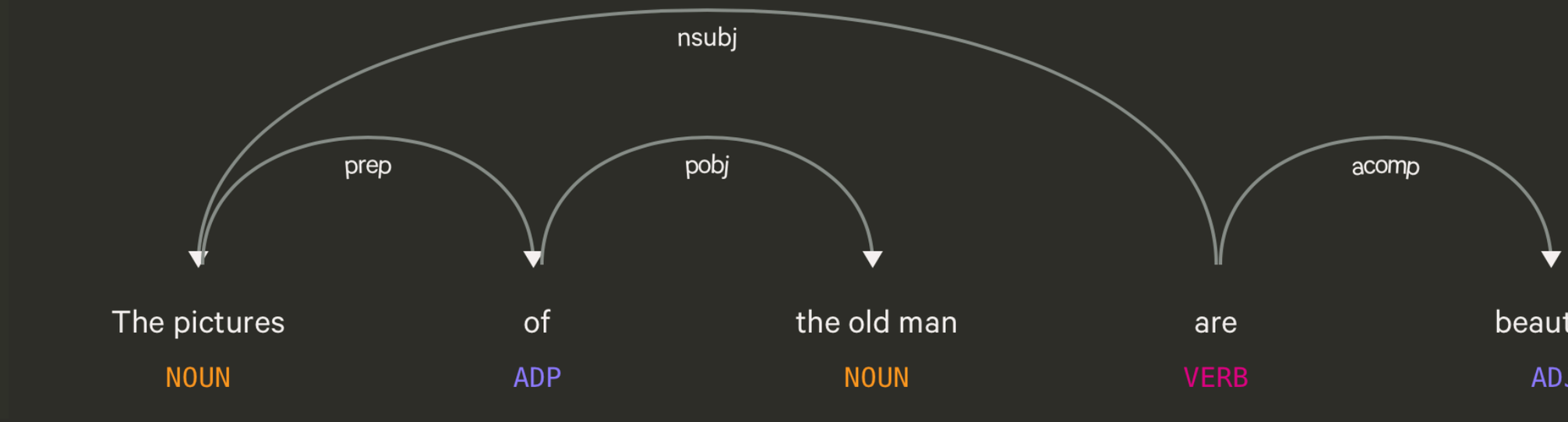
- Occur in a window of +/- 2 words, in the same sentence, in the same document

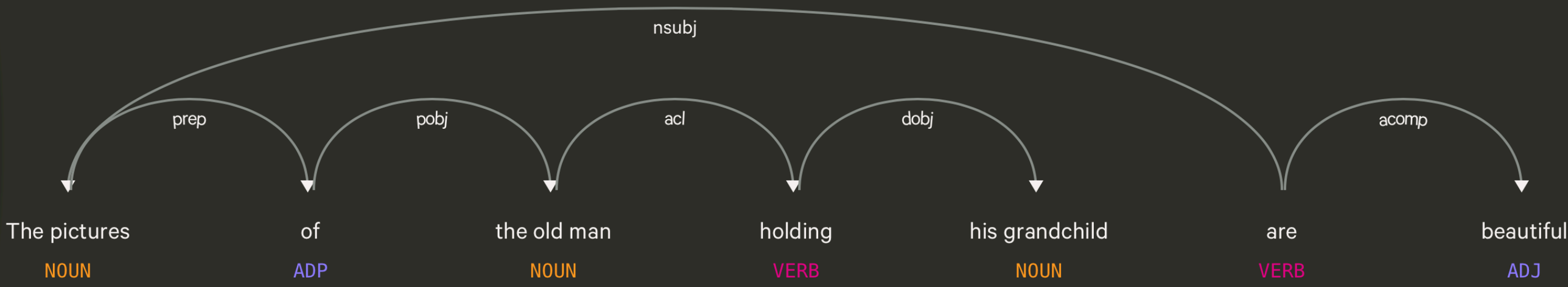
- Instead of window of words use more complex contexts: dependency patters. Subj-of-verb, adj-mod, obj-of-verb

- Languages have long distance dependencies

  *The **pictures are** beautiful.*
  *The **pictures** of the old man **are** beautiful.*
  *The **pictures** of the old man holding his grandchild **are** beautiful.*


nsubj
acomp
The pictures
NOUN
are
VERB
beautiful
ADJ


nsubj
prep
pobj
acomp
The pictures
NOUN
of
ADP
the old man
NOUN
are
VERB
beautiful
ADJ


nsubj
prep
pobj
acl
dobj
acomp
The pictures
NOUN
of
ADP
the old man
NOUN
holding
VERB
his grandchild
NOUN
are
VERB
beautiful
ADJ

Using syntax to define a word's context

- Zellig Harris (1968) "The meaning of entities, and the meaning of grammatical relations among them, is related to the restriction of combinations of these entities relative to other entities"

- **Duty** and **Responsibility** have similar syntactic distributions

Modified by adjectives	additional, administrative, assumed, collective, congressional, constitutional …
Object of verbs	assert, assign, assume, attend to, avoid, become, breach..

Alternates to counts

- Raw word frequency is not a great measure of association between words. It's very skewed "the" and "of" are very frequent, but maybe not the most discriminative

- We'd rather have a measure that asks whether a context word is particularly informative about the target word.

- Instead of raw counts, it's common to transform vectors using TF-IDF or PPMI

TF-IDF

*Term frequency * inverse document frequency*

How often a word occurred in a document

1 over the number of documents that it occurred in

Sparse v. Dense Vectors

- Co-occurrence matrix (weighted by TF-IDF or mutual information)
  - **Long** (length |V| = 50,000+)
  - **Sparse** (most elements are zeros)
- Alternative: learn vectors that are
  - **Short** (length 200-1000)
  - **Dense** (most elements are non-zero)

Sparse v. Dense Vectors

- Why dense vectors? Short vectors may be easier to use as features in machine learning (fewer weights to tune)

- Dense vectors may *generalize* better than storing explicit counts

- Neural networks produce dense vectors as output from their final hidden layer. These "word embeddings" are very popular at the moment

Skip-grams, CBOW

Mikolov et al. 2013

Learn embeddings as part of the process of word prediction.

Train a neural network to predict neighboring words

Inspired by neural net language models.

In so doing, learn dense embeddings for the words in the training corpus.

Advantages:

Fast, easy to train (much faster than SVD)

Available online in the word2vec package

Including sets of pretrained embeddings!

Skip-Grams

- Predict each neighboring word in a context window of 2C of surrounding words
- So for C=2, we are given a word w_t and we try to predict its 4 surrounding words
- $$[w_{t-2}, w_{t-1}, w_{t+1}, w_{t+2}]$$
- Uses "negative sampling" for training

Negative sampling

lemon, a [tablespoon of apricot preserves or] jam

c1 c2 w c3 c4

We want predictions
of these words to be high

And these words to be low

[cement metaphysical dear coaxial apricot attendant whence forever puddle]

n1 n2 n3 n4 n5 n6 n7 n8

Neural Network

Input layer

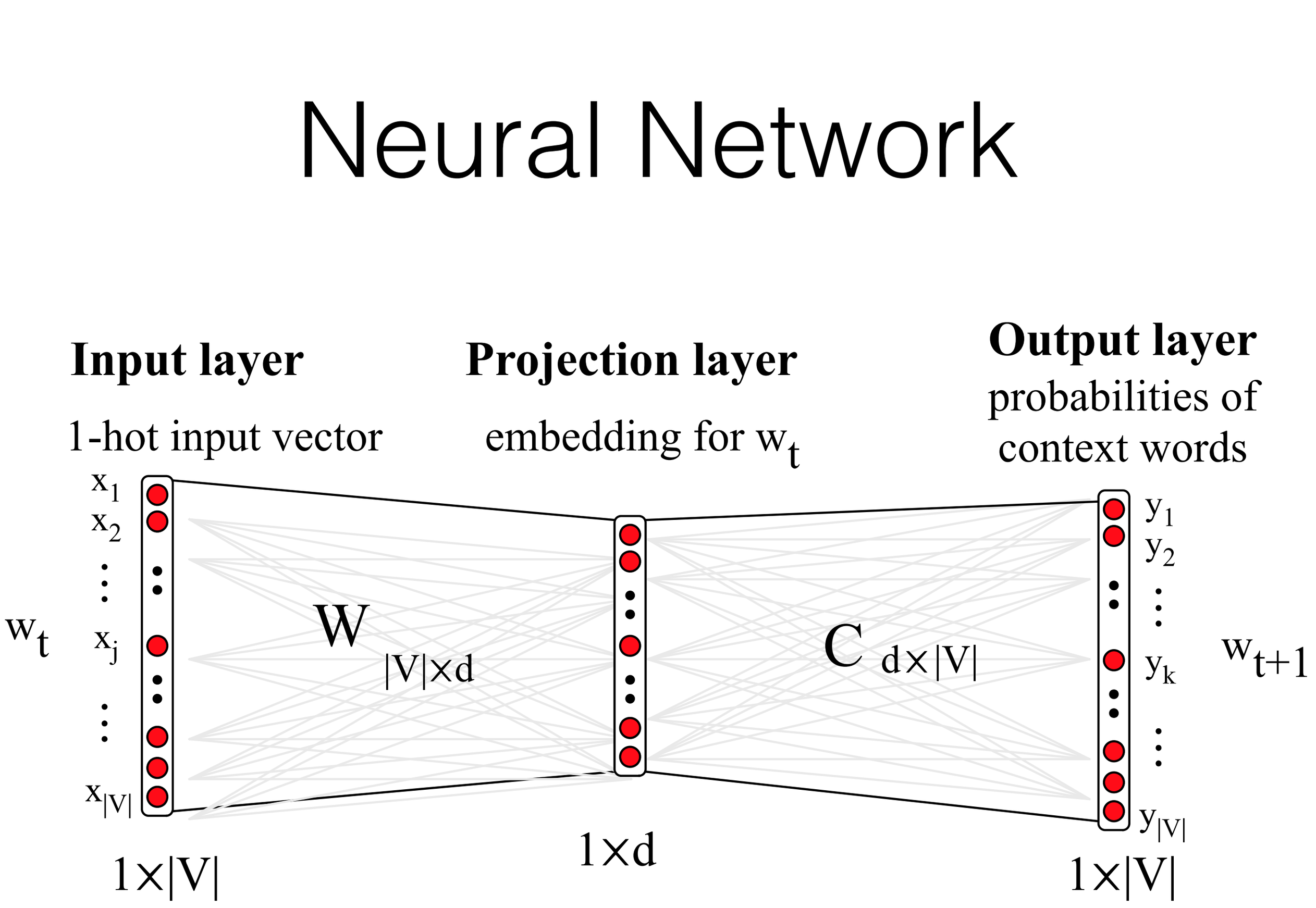
1-hot input vector

Projection layer

embedding for w_t

Output layer

probabilities of context words



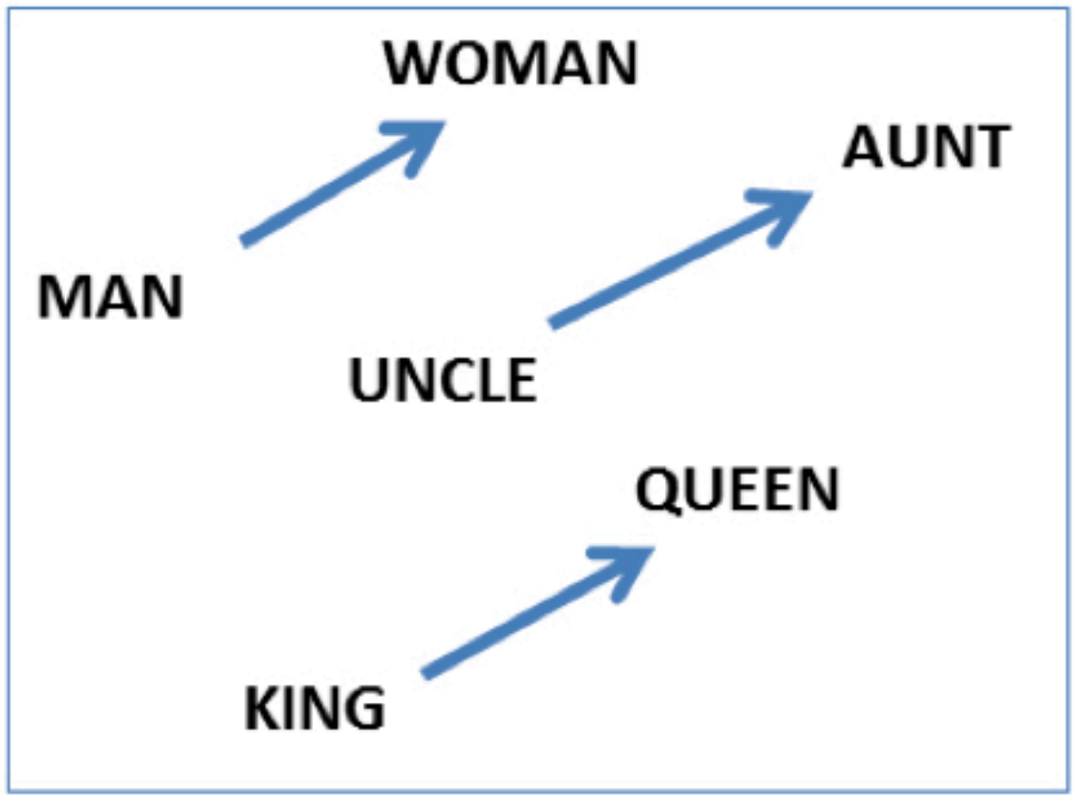
Properties of Embeddings

- Nearest Neighbors are surprisingly good

Redmond	Havel	ninjutsu	graffiti	capitulate
Redmond Wash.	Vaclav Havel	ninja	spray paint	capitulation
Redmond Washington	president Vaclav Havel	martial arts	grafitti	capitulated
Microsoft	Velvet Revolution	swordsmanship	taggers	capitulating

Embeddings capture relational meanings

- .vector('king') - vector('man') + vector('queen') ≅ vector('woman')



Magnitude: A Fast, Efficient Universal Vector Embedding Utility Package

Ajay Patel
Plasticity Inc.
San Francisco, CA
ajay@plasticity.ai

Alexander Sands
Plasticity Inc.
San Francisco, CA
alex@plasticity.ai

Chris Callison-Burch
Computer and Information
Science Department
University of Pennsylvania
ccb@upenn.edu

Marianna Apidianaki
LIMSI, CNRS
Université Paris-Saclay
91403 Orsay, France
marapi@seas.upenn.edu

Abstract

Vector space embedding models like word2vec, GloVe, and fastText are extremely popular representations in natural language processing (NLP) applications. We present Magnitude, a fast, lightweight tool for utilizing and processing embeddings. Magnitude is an open source Python package with a compact vector storage file format that allows for efficient manipulation of huge numbers of embeddings. Magnitude performs common operations up to 60 to 6,000 times faster than Gensim. Magnitude introduces several

Metric	Cold	Warm
Initial load time	97x	–
Single key query	1x	110x
Multiple key query (n=25)	68x	3x
k-NN search query (k=10)	1x	5,935x

Table 1: Speed comparison of Magnitude versus Gensim for common operations. The 'cold' column represents the first time the operation is called. The 'warm' column indicates a subsequent call with the same keys.

file, a 97x speed-up. Gensim uses 5GB of RAM versus 18KB for Magnitude.

Demo of word vectors

```
# Install Magnitude
pip3 install pymagnitude

# Download Google's word2vec vectors
wget http://magnitude.plasticity.ai/word2vec+approx/GoogleNews-vectors-negative300.magnitude
# Warning it's 11GB large

# Start Python, and try the commands
# on the next slide
python3
```

Demo of word vectors

```
from pymagnitude import *
vectors = Magnitude("GoogleNews-vectors-
negative300-2.magnitude")

queen = vectors.query('queen')
king = vectors.query("king")
vectors.similarity(king, queen)
# 0.6510958

vectors.most_similar_approx(king, topn=5)
#[('king', 1.0), ('kings', 0.72),
('prince', 0.62), ('sultan', 0.59),
('ruler', 0.58)]
```

Many possible models

Matrix type
Term-document
Term-context
Pattern-pair

Dim. Reduction
word2vec
GloVe
PCA
LDA
LSA

Reweighting
length norm.
TF-IDF
PPMI
probabilities

Comparisons
cosine
Manhattan
Jaccard
KL divergence
JS distance
DICE

Many possible models

Matrix type
Term-document
Term-context
Pattern-pair

Dim. Reduction
word2vec
GloVe
PCA
LDA
LSA

Reweighting
length norm.
TF-IDF
PPMI
probabilities

Comparisons
cosine
Manhattan
Jaccard
KL divergence
JS distance
DICE

How many dimensions?

What modifications should we make to the input?

How do we pick the right combination?

Evaluating word vectors

2 kinds of evaluation:

1) Extrinsic evaluation = task based

2) Intrinsic

Psycholinguistics Data

Cos sim	U	V	Judgements
…	Love	Sex	6.8
	Tiger	Cat	7.3
	Tiger	Tiger	10
	Fertility	Egg	6.7
	Stock	Egg	1.8
	Professor	Cucumber	0.3

Computing correlation

Human ordering

Love	Sex	>	Professor	Cucumber

System ordering predicts

< … this is a discordant pair

> … this is a concordant pair

Does similarity == semantics?

- Word vectors fail to capture logical implications

  Dog(x) → Animal(x)
  Dog(x) → not (Gorilla(x))

- Word vectors for antonyms or logically exclusive things are often very similar
  sim(boys, girls)
  sim(cats, dogs)
  sim(France, Germany)
  sim(rise, fall)

Acknowledgements

- Thanks to the following people for providing slides

- Dan Jurafsky from his textbook Speech and Language Processing version 3

- Peter Turney and Patrick Pantel From Frequency to Meaning: Vector Space Models of Semantics

- Chris Potts from his course CS224U: Natural Language Understanding

- Denis Paperno, Marco Baroni, German Kruszewski from their talk on Deriving Boolean Structures from Distributional Vectors