UMSI

SI 630

Natural Language Processing:

Algorithms and People

Lecture 3: Language Models
Jan. 22, 2019

David Jurgens
jurgens@umich.edu

Some slides courtesy of Dan Jurafsky and David Bamman

Important stuff first

Important stuff first

“I think there is too much material covered in today's class. The pre-requirement for this course is just SI507 plus some stats and math knowledge, but for those who are not familiar with ML and DL algorithms, like me, accepting so much new stuff is hard within the three hours. I am concerned about whether our HW would need all of the material we are taught in the class. And I hope the professor could reduce the speed for those are new to ML next time.” — *anon*

Important stuff first

“I think there is too much material covered in today's class. The pre-requirement for this course is just SI507 plus some stats and math knowledge, but for those who are not familiar with ML and DL algorithms, like me, accepting so much new stuff is hard within the three hours. I am concerned about whether our HW would need all of the material we are taught in the class. And I hope the professor could reduce the speed for those are new to ML next time.” — *anon*

Thank you for posting this!

My thoughts

My thoughts

- This *is* a challenging course

My thoughts

- This *is* a challenging course
- You are *great* students

My thoughts

- This *is* a challenging course
- You are *great* students
- You can be capable of passing but it just a matter of time and preparedness—not everyone is at the same level

My thoughts

- This *is* a challenging course
- You are *great* students
- You can be capable of passing but it just a matter of time and preparedness—not everyone is at the same level
- It is not a personal or moral failure if you drop this class

My thoughts

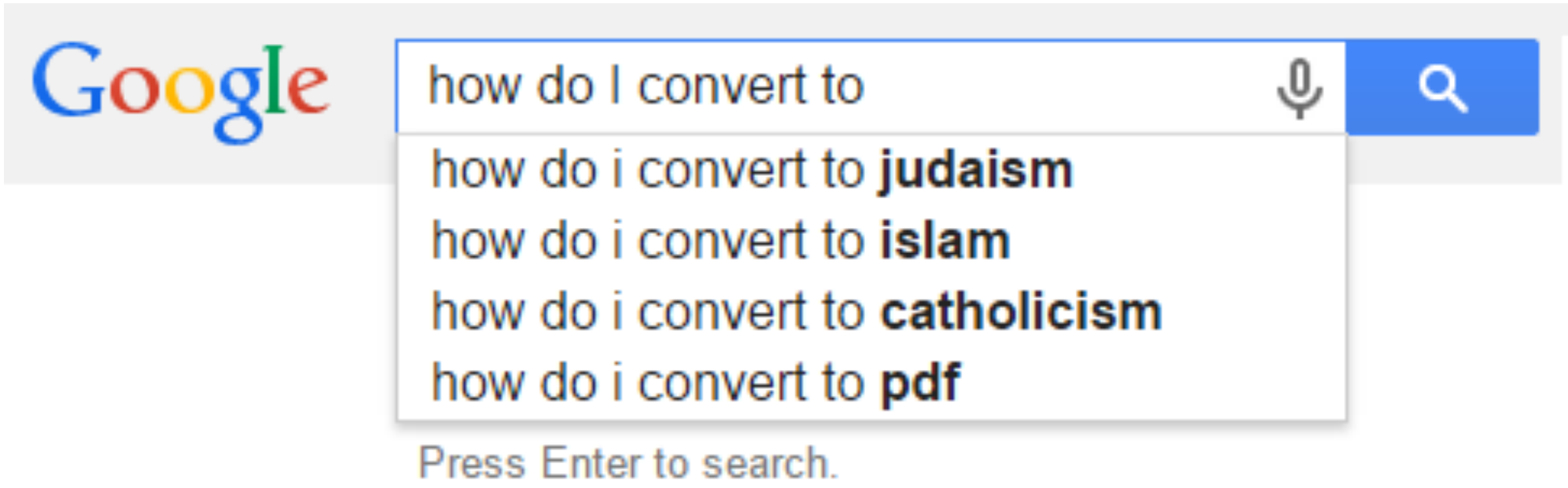
- This *is* a challenging course
- You are *great* students
- You can be capable of passing but it just a matter of time and preparedness—not everyone is at the same level
    - It is not a personal or moral failure if you drop this class
- Enroll in SI 511-630 if you need it. We designed it *for you*

My thoughts

- This *is* a challenging course
- You are *great* students
- You can be capable of passing but it just a matter of time and preparedness—not everyone is at the same level
    - It is not a personal or moral failure if you drop this class
- Enroll in SI 511-630 if you need it. We designed it *for you*
- Take your mental, personal, and physical health seriously

Oh you may
not think I'm pretty,
But don't judge on
what you see, I'll eat
myself if
you can
find A
smarter hat than
me. You can keep
your bowlers black,
Your top hats sleek and
tall, For I'm the Hogwarts
Sorting Hat And
I can cap them all.
There's nothing hidden
in your head The
Sorting Hat
can't see,
So try me on and
I will tell you Where you
ought to be. You might belong
in Gryffindor,
Where dwell the
brave at
heart, Their daring,
nerve, and chivalry Set Gryffindors
apart; You might belong in Hufflepuff, Where they
are just and loyal, Those patient Hufflepuffs are true And unafraid
of toil; Or yet in wise old Ravenclaw, if you've a ready mind,
Where those of wit and learning, Will always find their
kind; Or perhaps in Slytherin You'll make your
real friends, Those cunning folks use any means To achieve
their ends. So put me on! Don't be afraid! And don't
get in a flap! You're in safe hands (though I
have none) For I'm a Thinking Cap!


AUTOCORRECT
DUCK
imgflip.com


Google
how do I convert to
how do i convert to judaism
how do i convert to islam
how do i convert to catholicism
how do i convert to pdf
Press Enter to search.

Which of these phrases is more likely?

- The child sat on the mat
- The child sat on the cat
- The child sat on the idea

Which of these phrases is more likely?

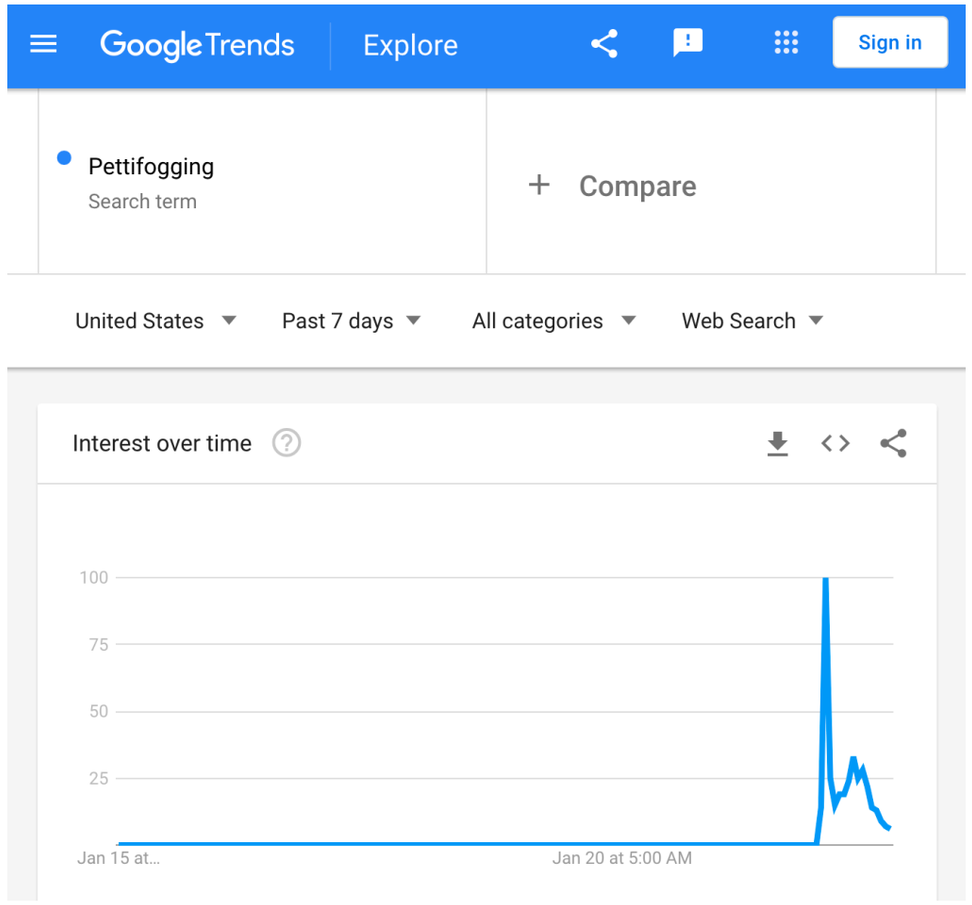
- The thinker sat on the mat
- The thinker sat on the cat
- The thinker sat on the idea

Which of these phrases is more likely?

I have had enough of these annoying people

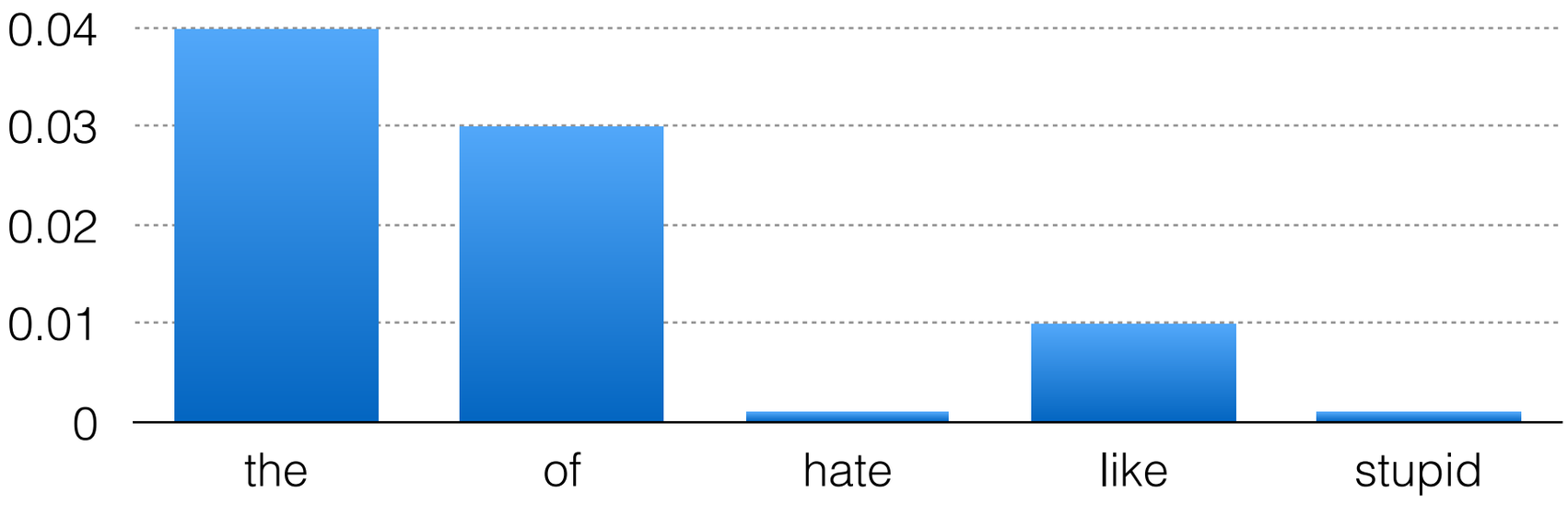
I have had enough of these pettifogging people

"In the 1905 Swayne trial, a senator objected when one of the managers used the word 'pettifogging' and the presiding officer said the word ought not to have been used. I don't think we need to aspire to that high of a standard, but I do think those addressing the Senate should remember where they are."



We already know a very simple language model

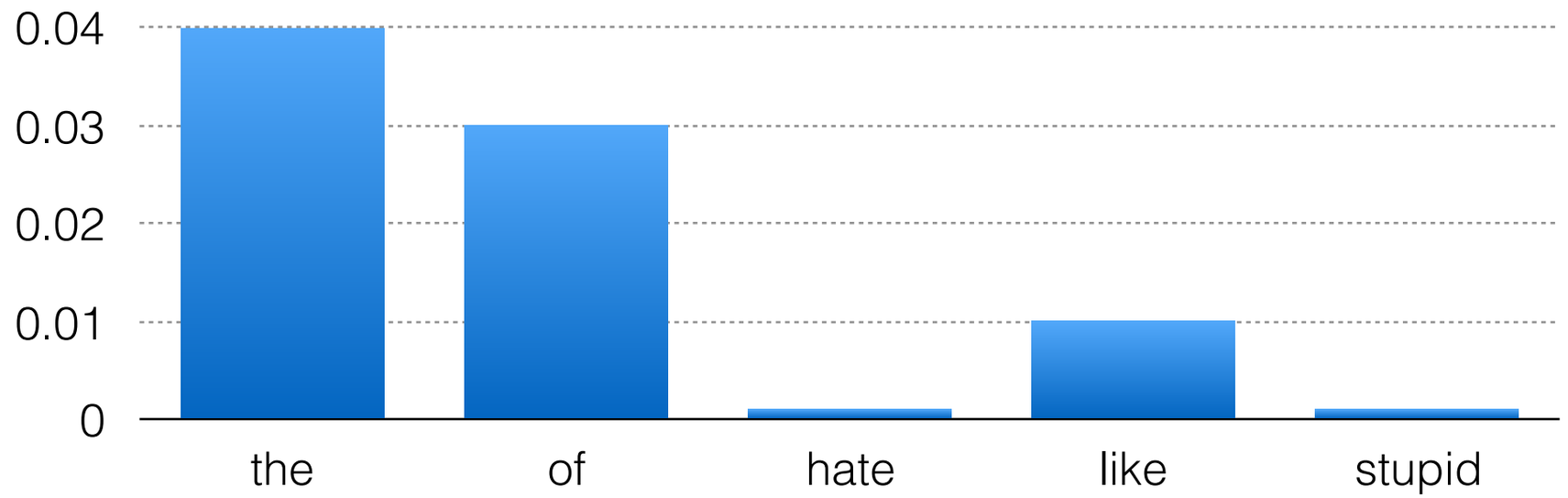
Word choice as weighted dice



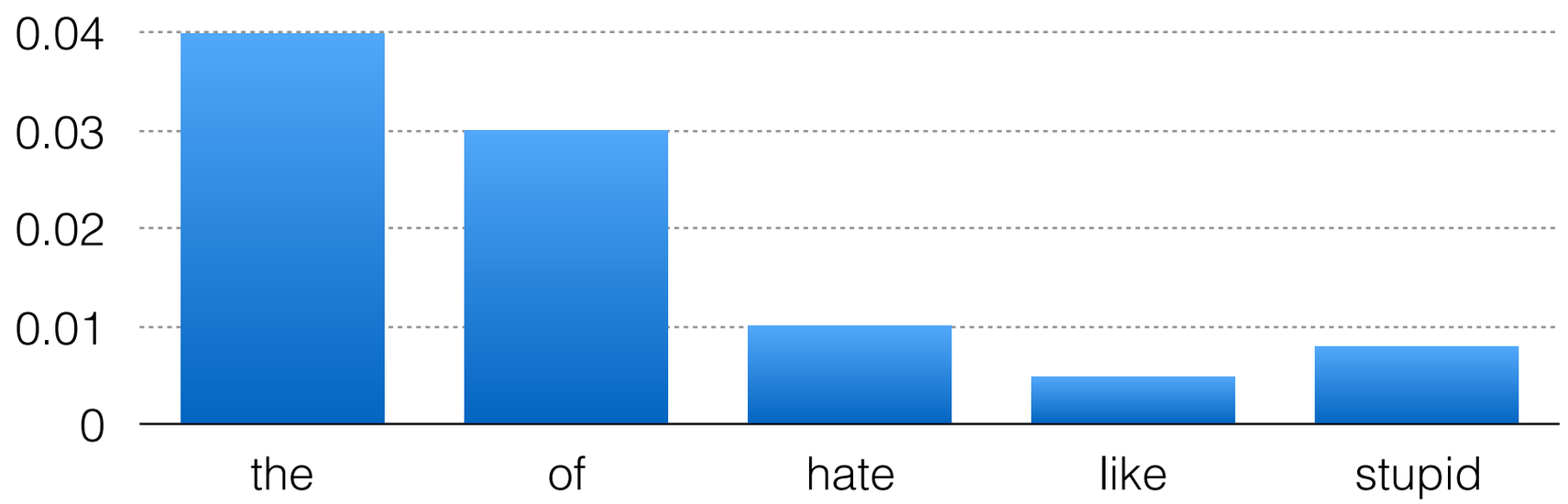
This is a simple language model

Here, we have *two* language models

positive reviews



negative reviews



Language Model

- Vocabulary $\mathscr{V}$ is a finite set of discrete symbols (e.g., words, characters); V = | $\mathscr{V}$ |

- $\mathscr{V}^+$ is the infinite set of sequences of symbols from $\mathscr{V}$; each sequence ends with STOP

- x ∈ $\mathscr{V}^+$

Language Model

$$P(w) = P(w_1, \ldots, w_n)$$

P("Call me Ishmael") =
P(w_1 = "call", w_2 = "me", w_3 = "Ishmael") x P(STOP)

$$\sum_{w \in V^+} P(w) = 1$$

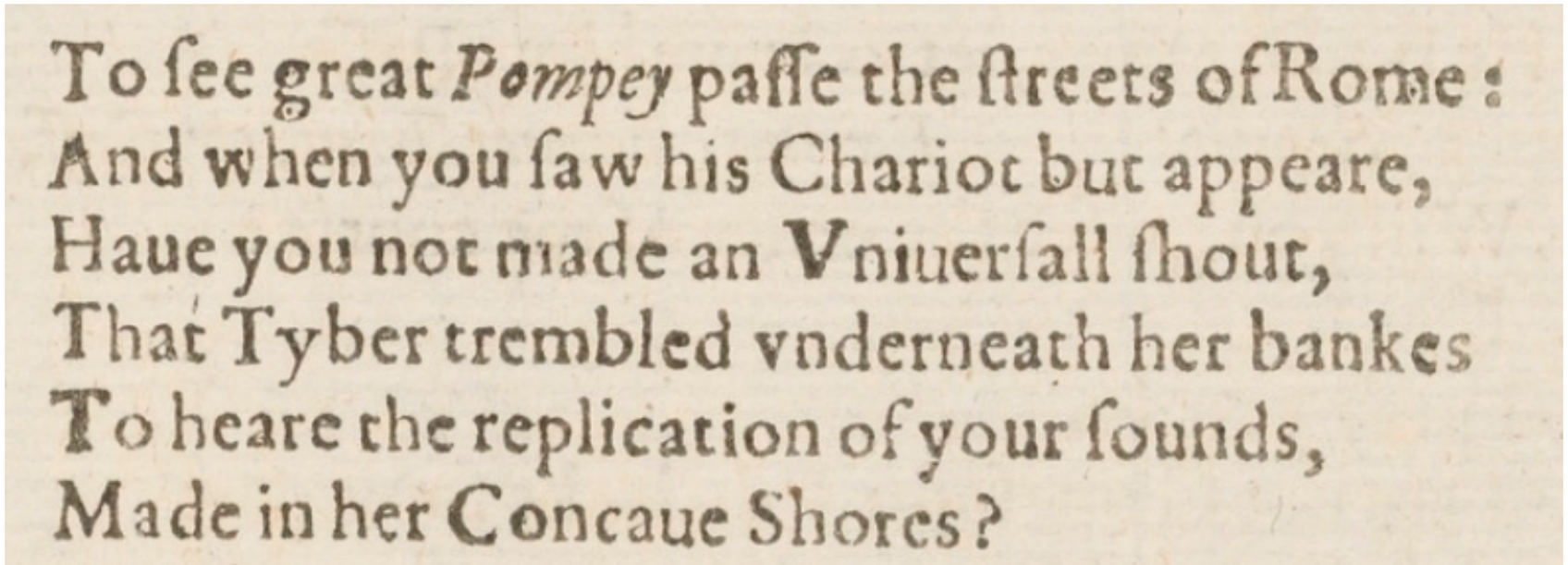
$$0 \leq P(w) \leq 1$$

over all sequence lengths!

Language Model

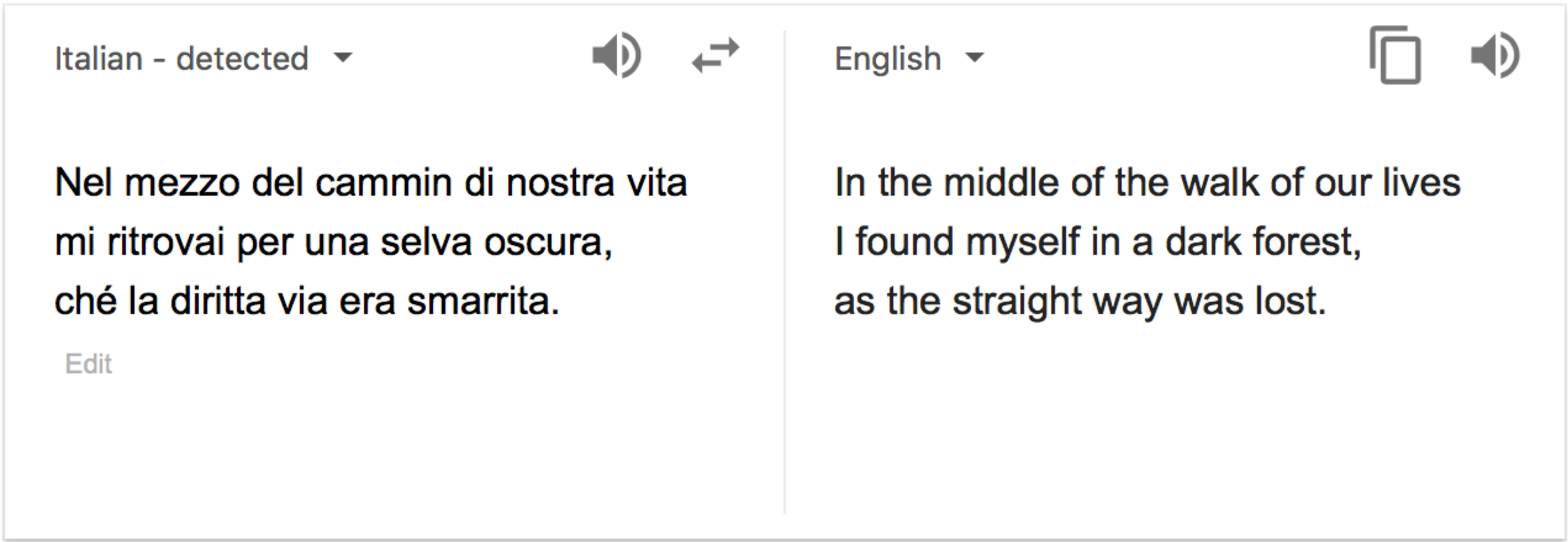
- Language models provide us with a way to quantify the likelihood of sequence — i.e., plausible sentences.

Optical Character Recognition (OCR)


To ſee great *Pompey* paſſe the ſtreets of Rome:
And when you ſaw his Chariot but appeare,
Haue you not made an Vniuerſall ſhout,
That Tyber trembled vnderneath her bankes
To heare the replication of your ſounds,
Made in her Concaue Shores?

- to fee great Pompey paffe the Areets of Rome:
- to see great Pompey passe the streets of Rome:

Machine translation



- Fidelity (to source text)
- Fluency (of the translation)

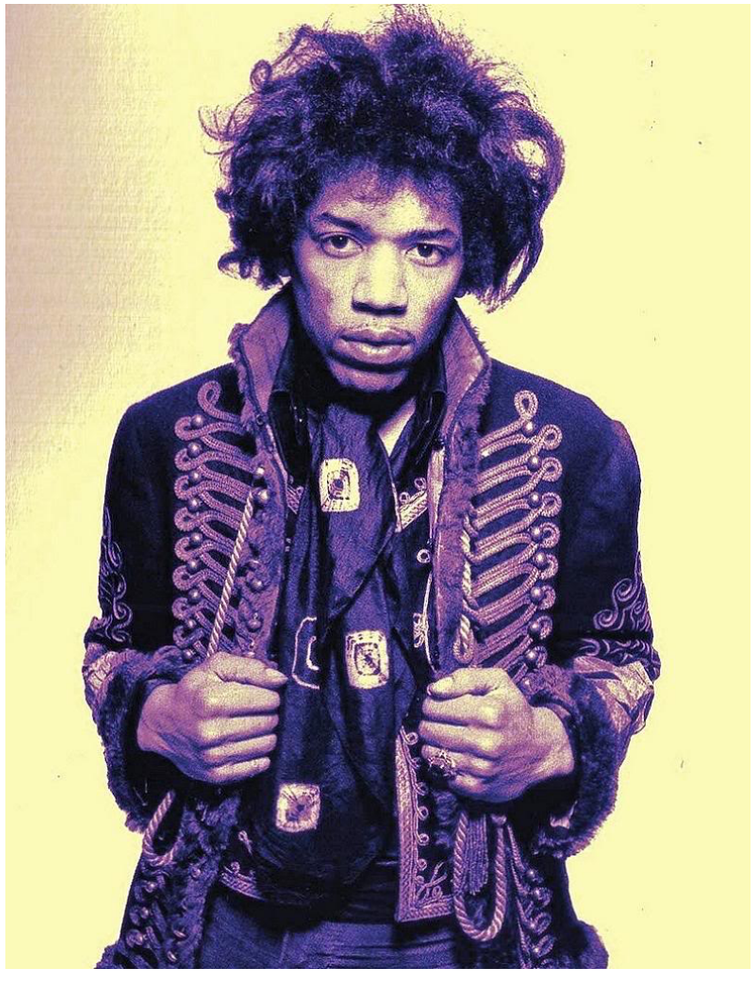
natural lan

natural lan**guage processing**
natural lan**guage understanding**
natural lan**guage processing with python**
natural lan**guage generation**

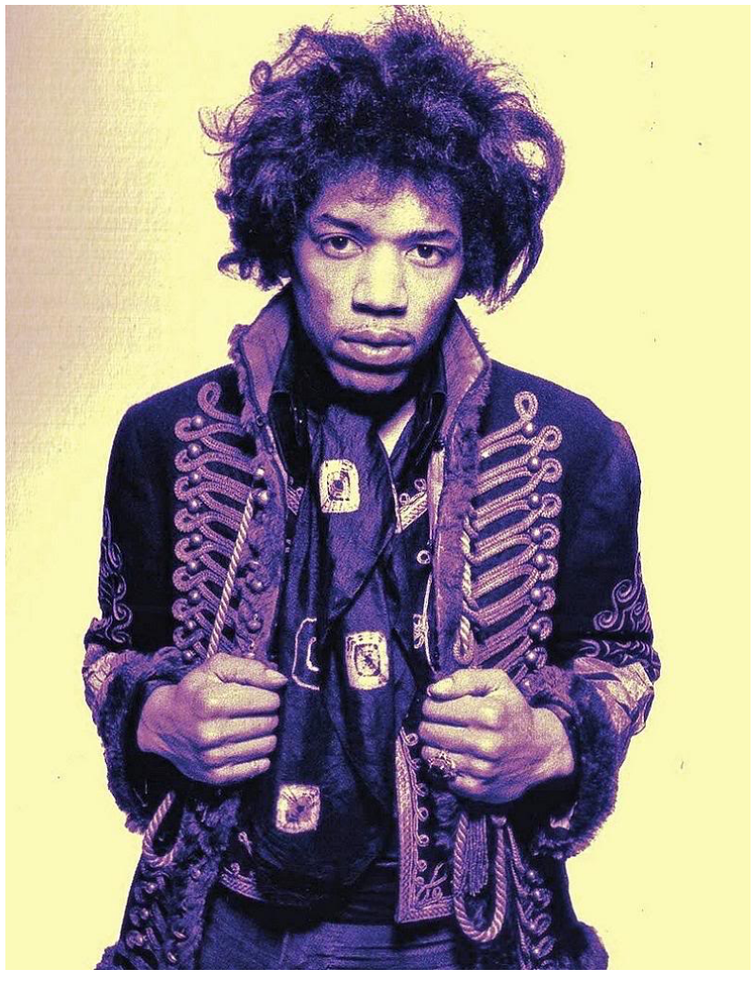
Google Search I'm Feeling Lucky

Report inappropriate predictions

Speech Recognition

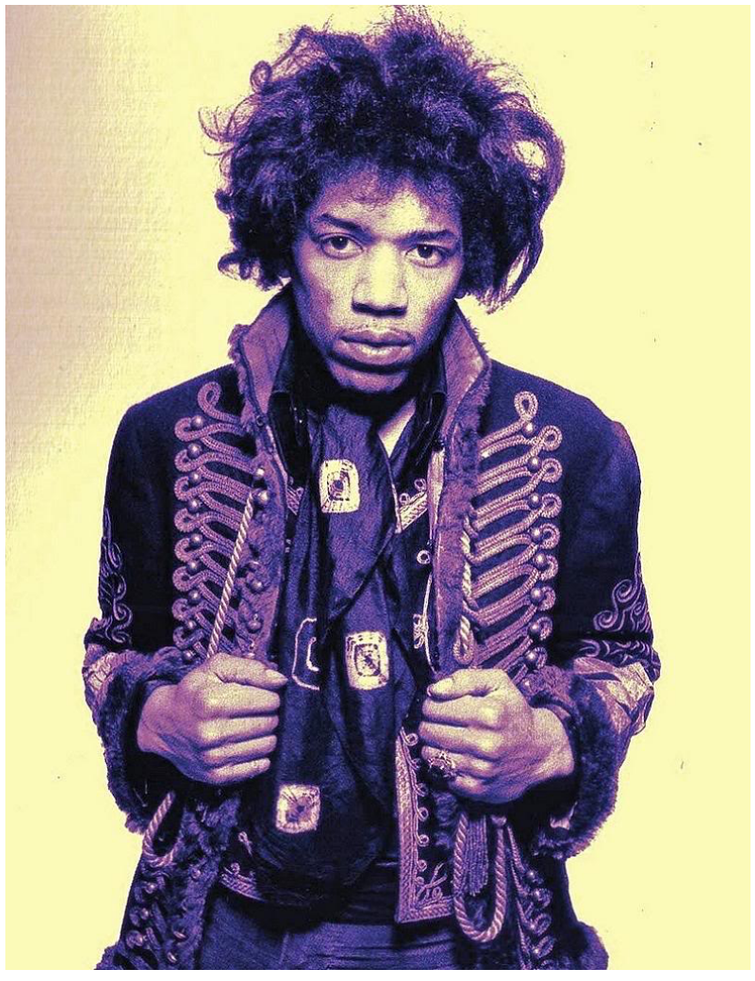


Speech Recognition



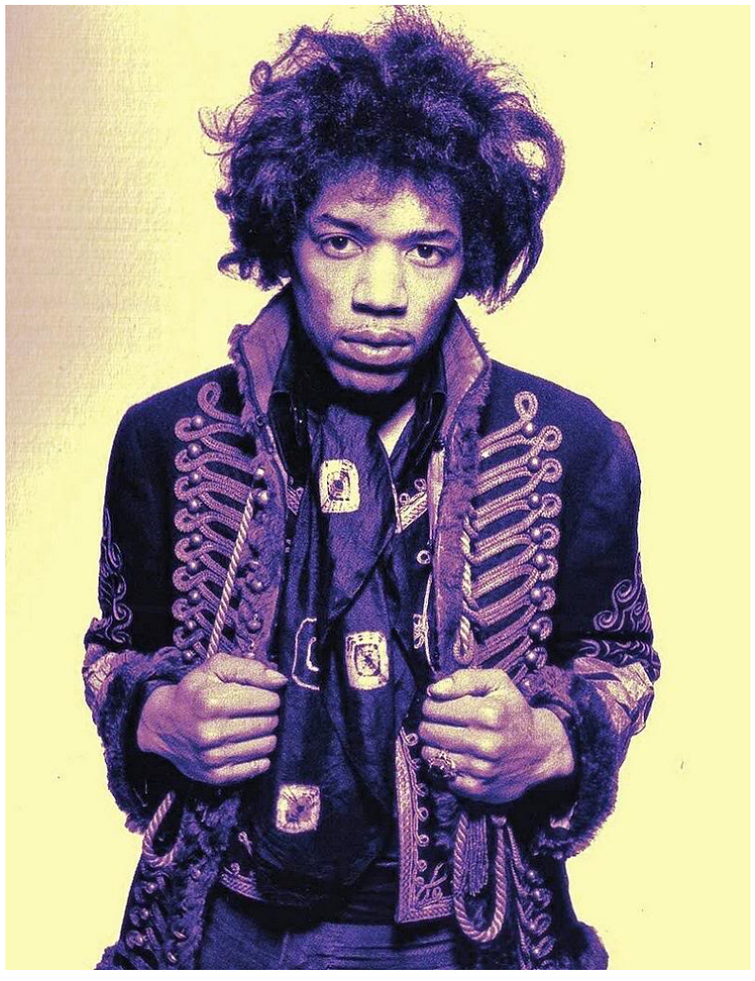
- 'Scuse me while I kiss the sky.

Speech Recognition



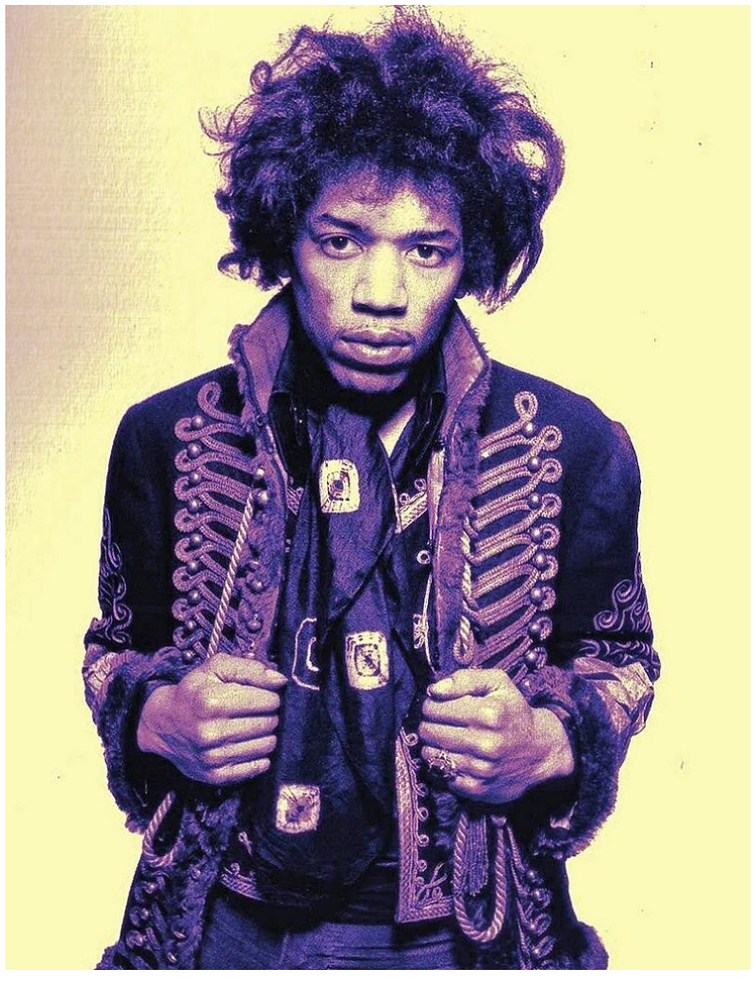
- 'Scuse me while I kiss the sky.
- 'Scuse me while I kiss this guy

Speech Recognition



- 'Scuse me while I kiss the sky.
- 'Scuse me while I kiss this guy
- 'Scuse me while I kiss this fly.

Speech Recognition



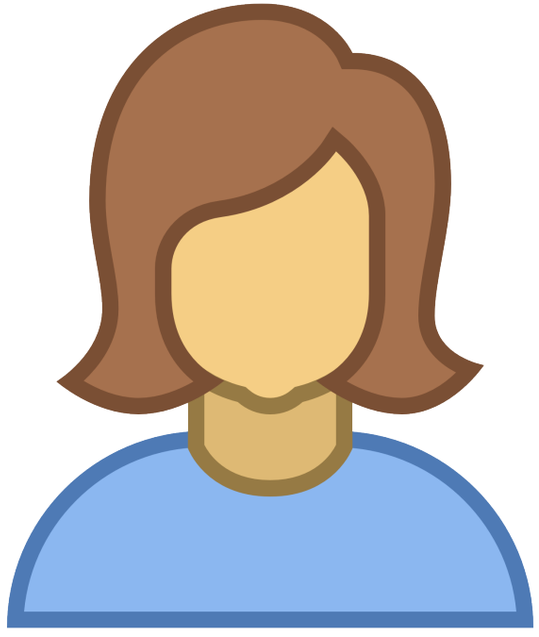
- 'Scuse me while I kiss the sky.
- 'Scuse me while I kiss this guy
- 'Scuse me while I kiss this fly.
- 'Scuse me while my biscuits fry

Dialogue generation

Baseline mutual information model (Li et al. 2015)	**Proposed reinforcement learning model**
A: Where are you going? (1) B: I'm going to the restroom. (2) A: See you later. (3) B: See you later. (4) A: See you later. (5) B: See you later. (6)	A: Where are you going? (1) B: I'm going to the police station. (2) A: I'll come with you. (3) B: No, no, no, no, you're not going anywhere. (4) A: Why? (5) B: I need you to stay here. (6) A: I don't know what you are talking about. (7) ...
A: how old are you? (1) B: I'm 16. (2) A: 16? (3) B: I don't know what you are talking about. (4) A: You don't know what you are saying. (5) B: I don't know what you are talking about . (6) A: You don't know what you are saying. (7) ...	A: How old are you? (1) B: I'm 16. Why are you asking? (2) A I thought you were 12. (3) B: What made you think so? (4) A: I don't know what you are talking about. (5) B: You don't know what you are saying. (6) ... ...

Li et al. (2016), "Deep Reinforcement Learning for Dialogue Generation" (EMNLP)

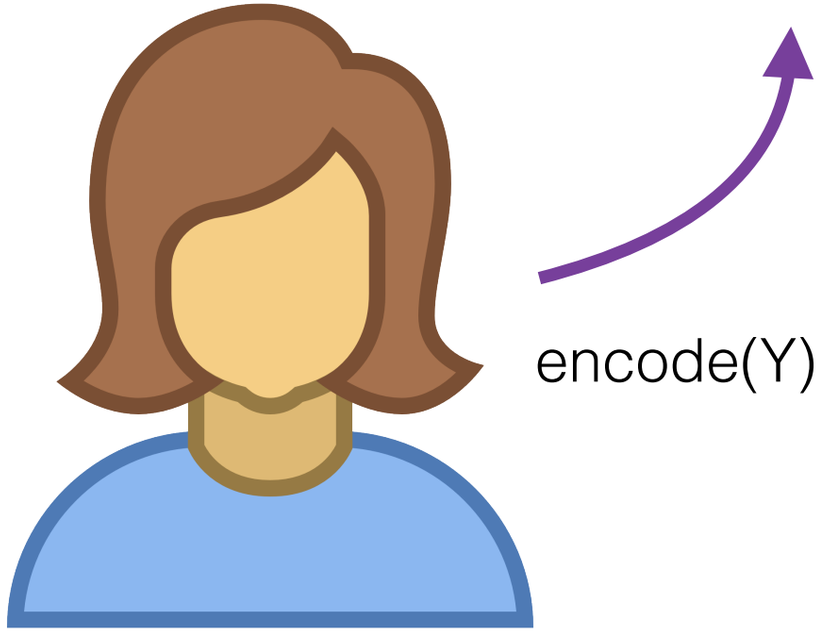
Information theoretic view



Shannon 1948

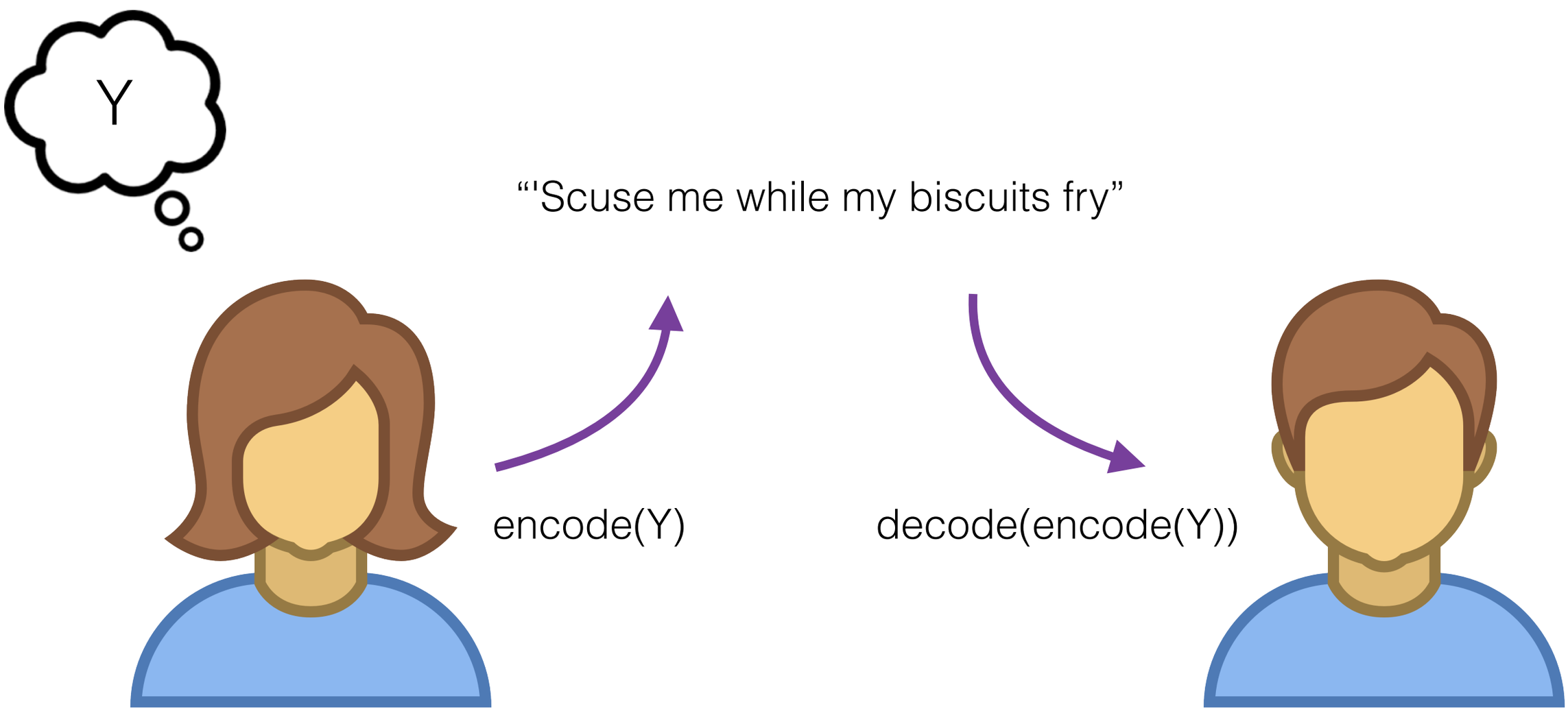
Information theoretic view

“'Scuse me while my biscuits fry”



Shannon 1948

Information theoretic view



Shannon 1948

Noisy Channel

	X	Y
Automated Speech Recognition (ASR)	speech signal	transcription
Machine Translation (MT)	target text	source text
Optical Character Recognition (OCR)	pixel densities	transcription

$$P(Y \mid X) \propto \underbrace{P(X \mid Y)}_{\text{channel model}} \quad \underbrace{P(Y)}_{\text{source model}}$$

Language Model

- Language modeling is the task of estimating P(w)
- Why is this hard?

P(“It was the best of times, it was the worst of times”)

Chain rule (of probability)

$$
\begin{aligned}
P(x_1, x_2, x_3, x_4, x_5) = P(x_1) \\
&\times P(x_2 \mid x_1) \\
&\times P(x_3 \mid x_1, x_2) \\
&\times P(x_4 \mid x_1, x_2, x_3) \\
&\times P(x_5 \mid x_1, x_2, x_3, x_4)
\end{aligned}
$$

Chain rule (of probability)

P(“It was the best of times, it was the worst of times”)

Chain rule (of probability)

P(“It”)

P(“was” | “It”)

this is easy

this is hard

$$P(w_1)$$

$$P(w_2 \mid w_1)$$

$$P(w_3 \mid w_1, w_2)$$

$$P(w_4 \mid w_1, w_2, w_3)$$

$$P(w_n \mid w_1, \ldots, w_{n-1})$$

P(“times” | “It was the best of times, it was the worst of”)

Markov assumption

first-order

$$P(x_i \mid x_1, \ldots x_{i-1}) \approx P(x_i \mid x_{i-1})$$

second-order

$$P(x_i \mid x_1, \ldots x_{i-1}) \approx P(x_i \mid x_{i-2}, x_{i-1})$$

Markov assumption

bigram model
(first-order markov)

$$\prod_{i}^{n} P(w_i \mid w_{i-1}) \times P(\text{STOP} \mid w_n)$$

trigram model
(second-order markov)

$$\prod_{i}^{n} P(w_i \mid w_{i-2}, w_{i-1})$$

$$\times P(\text{STOP} \mid w_{n-1}, w_n)$$

"It was the best of times, it was the worst of times"

$$P(It \mid \text{START}_1, \text{START}_2)$$

$$P(was \mid \text{START}_2, It)$$

$$P(the \mid It, was)$$

...

$$P(times \mid worst, of)$$

$$P(\text{STOP} \mid of, times)$$

Estimation

unigram

$$\prod_{i}^{n} P(w_i)$$

$$\times P(STOP)$$

Maximum likelihood estimate

$$\frac{c(w_i)}{N}$$

Estimation

unigram

$$\prod_{i}^{n} P(w_i)$$

$$\times P(STOP)$$

bigram

$$\prod_{i}^{n} P(w_i \mid w_{i-1})$$

$$\times P(STOP \mid w_n)$$

Maximum likelihood estimate

$$\frac{c(w_i)}{N}$$

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

Estimation

unigram

$$\prod_{i}^{n} P(w_i) \times P(STOP)$$

bigram

$$\prod_{i}^{n} P(w_i \mid w_{i-1}) \times P(STOP \mid w_n)$$

trigram

$$\prod_{i}^{n} P(w_i \mid w_{i-2}, w_{i-1}) \times P(STOP \mid w_{n-1}, w_n)$$

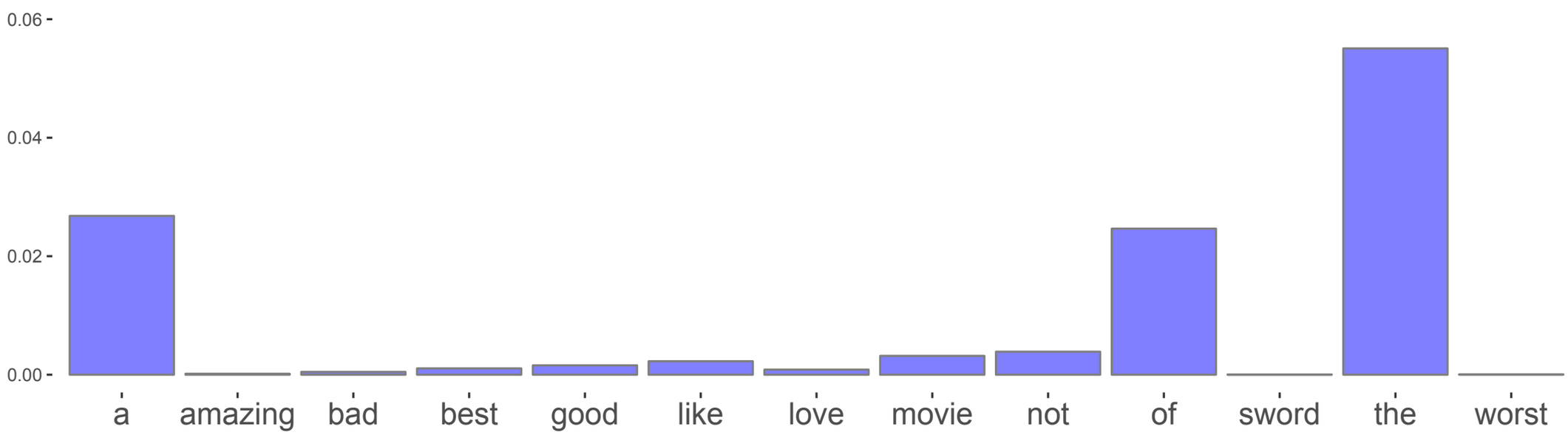
Maximum likelihood estimate

$$\frac{c(w_i)}{N}$$

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

$$\frac{c(w_{i-2}, w_{i-1}, w_i)}{c(w_{i-2}, w_{i-1})}$$

Generating



- What we learn in estimating language models is P(word | context), where context — at least here — is the previous n-1 words (for ngram of order n)

- We have one multinomial over the vocabulary (including STOP) for each context

Generating

- As we sample,
the words we
generate form
the new context
we condition on

Generating

context1	context2	generated word

- As we sample,
the words we
generate form
the new context
we condition on

Generating

- As we sample, the words we generate form the new context we condition on

context1	context2	generated word
START	START	

Generating

- As we sample, the words we generate form the new context we condition on

context1	context2	generated word
START	START	The

Generating

- As we sample, the words we generate form the new context we condition on

context1	context2	generated word
START	START	The
START	The	

Generating

- As we sample, the words we generate form the new context we condition on

context1	context2	generated word
START	START	The
START	The	dog

Generating

- As we sample, the words we generate form the new context we condition on

context1	context2	generated word
START	START	The
START	The	dog
The	dog	

Generating

- As we sample, the words we generate form the new context we condition on

context1	context2	generated word
START	START	The
START	The	dog
The	dog	walked

Generating

- As we sample, the words we generate form the new context we condition on

context1	context2	generated word
START	START	The
START	The	dog
The	dog	walked
dog	walked	

Generating

- As we sample, the words we generate form the new context we condition on

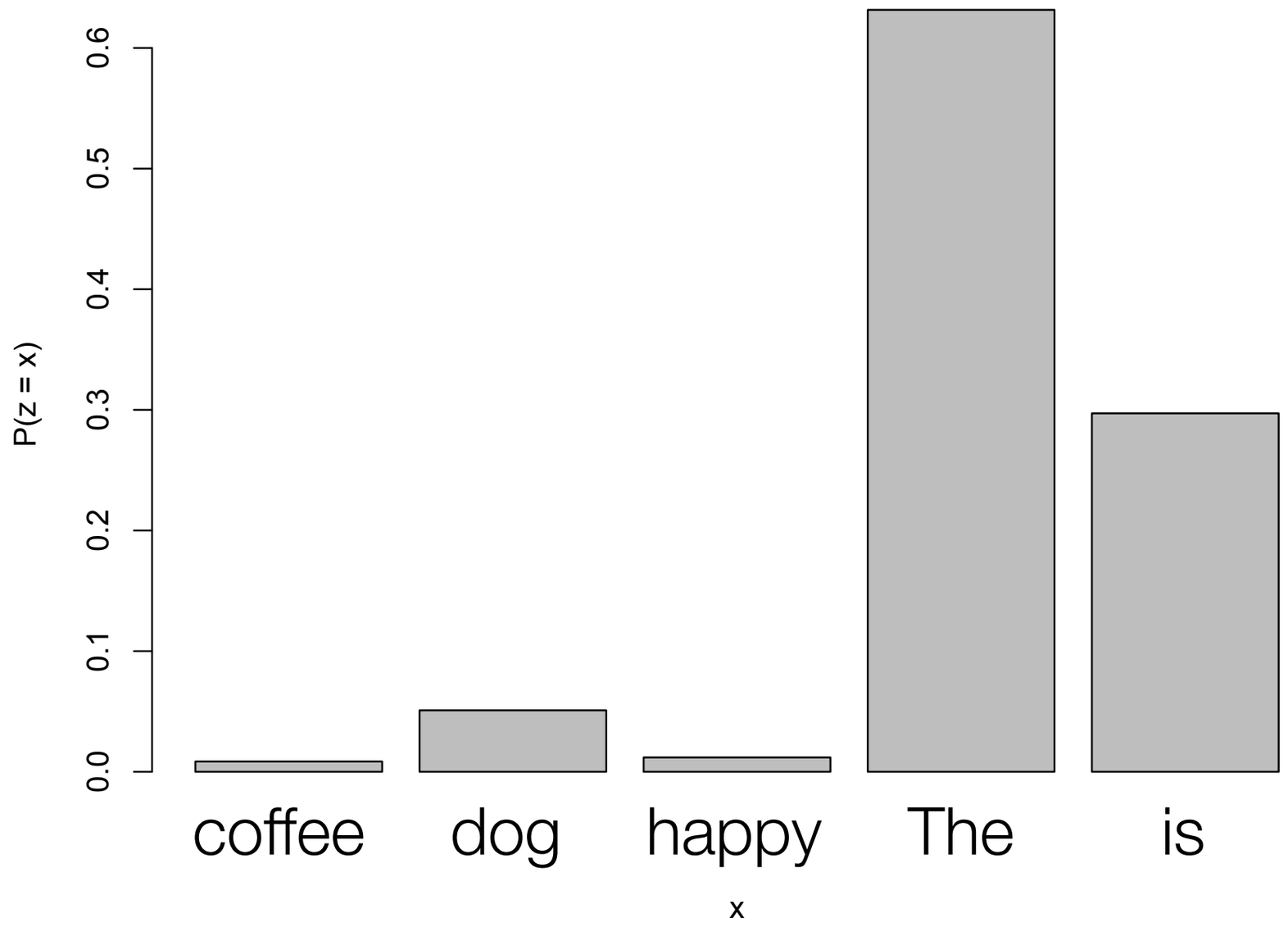
context1	context2	generated word
START	START	The
START	The	dog
The	dog	walked
dog	walked	in

Aside: how do we sample from a multinomial?

Sampling from a Multinomial

Probability mass function (PMF)

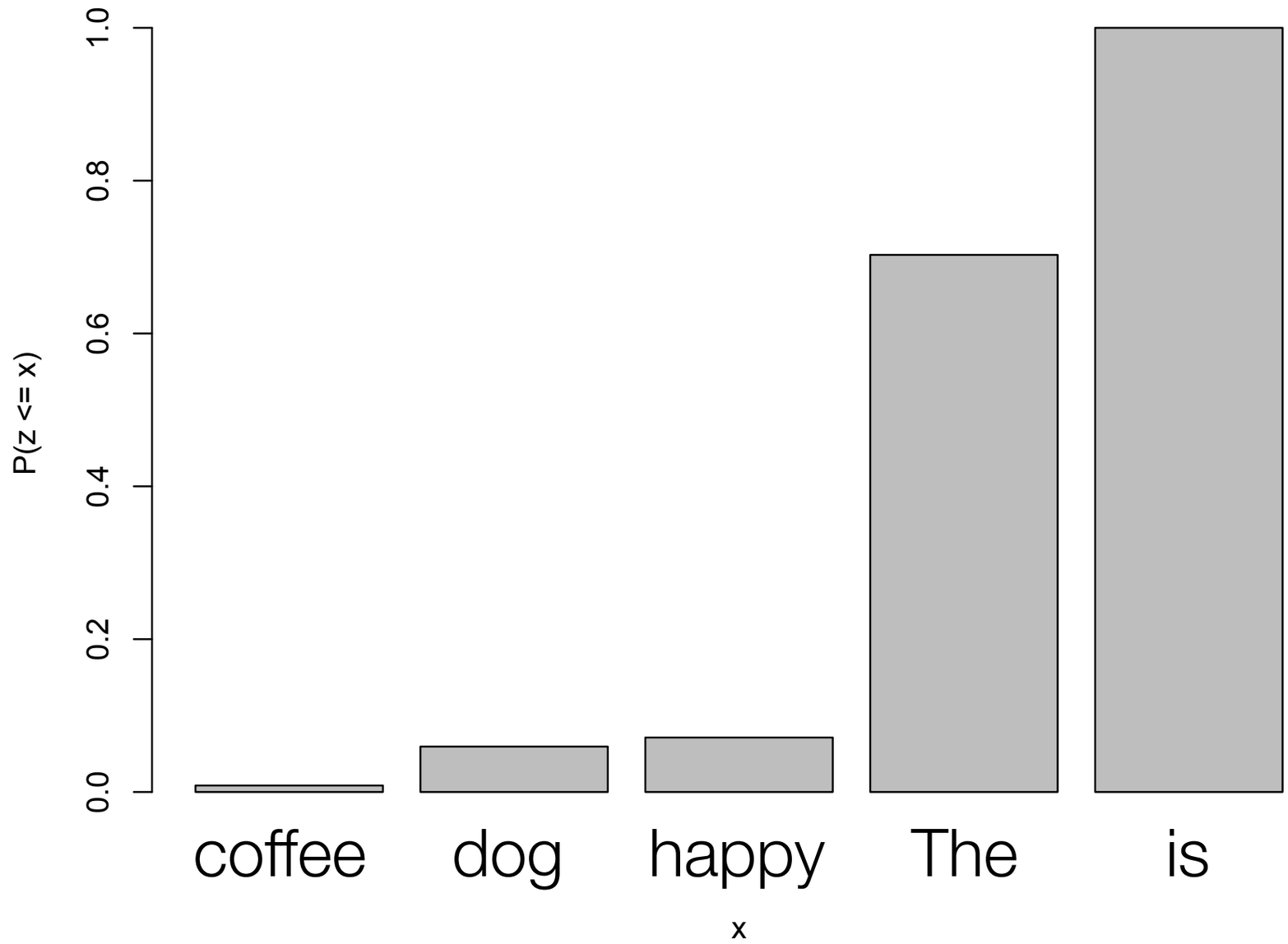
P(z = x) exactly



Sampling from a Multinomial

Cumulative density function (CDF)

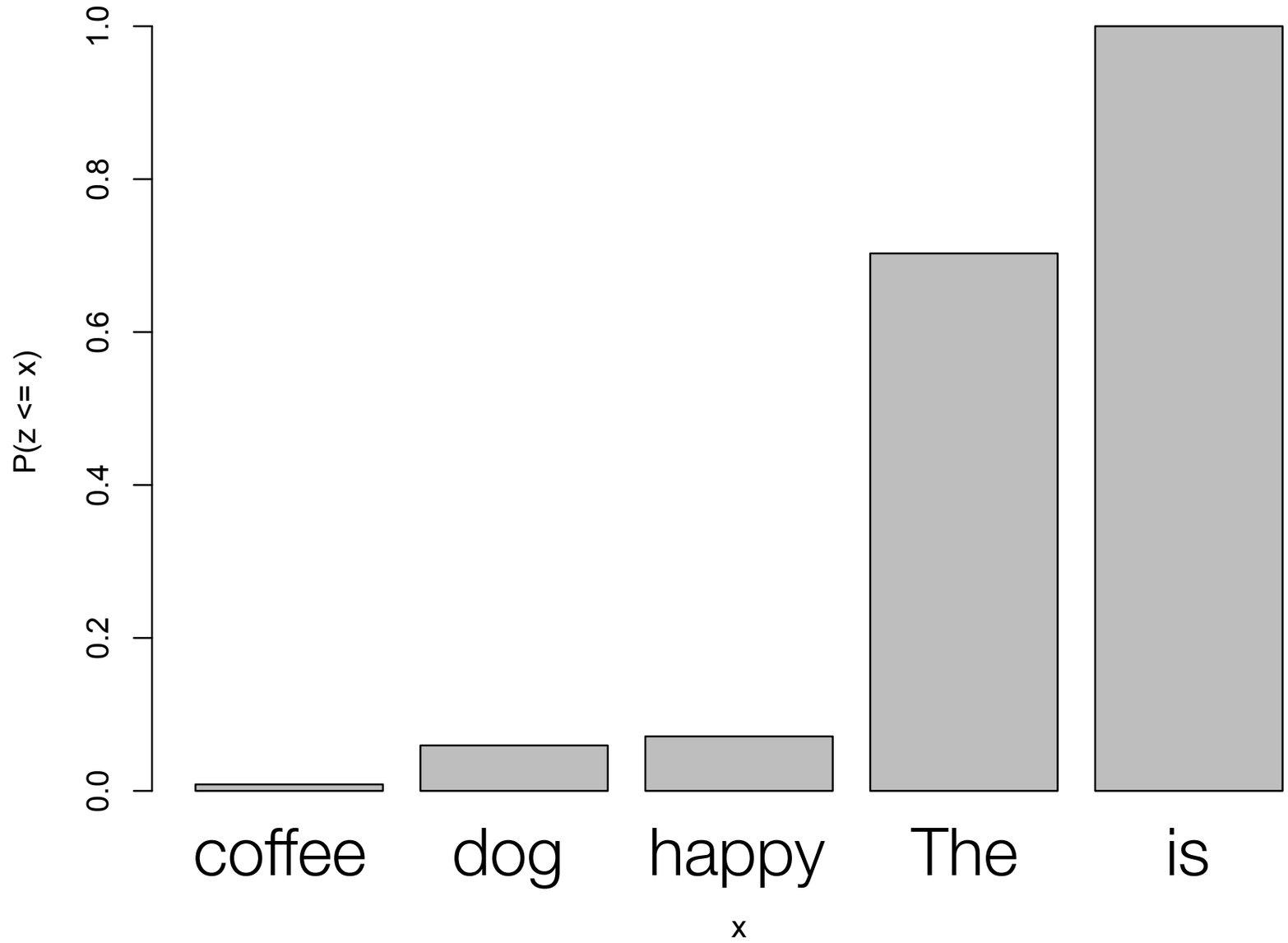
$P(z \leq x)$



Sampling from a Multinomial

Sample *p* uniformly in [0,1]

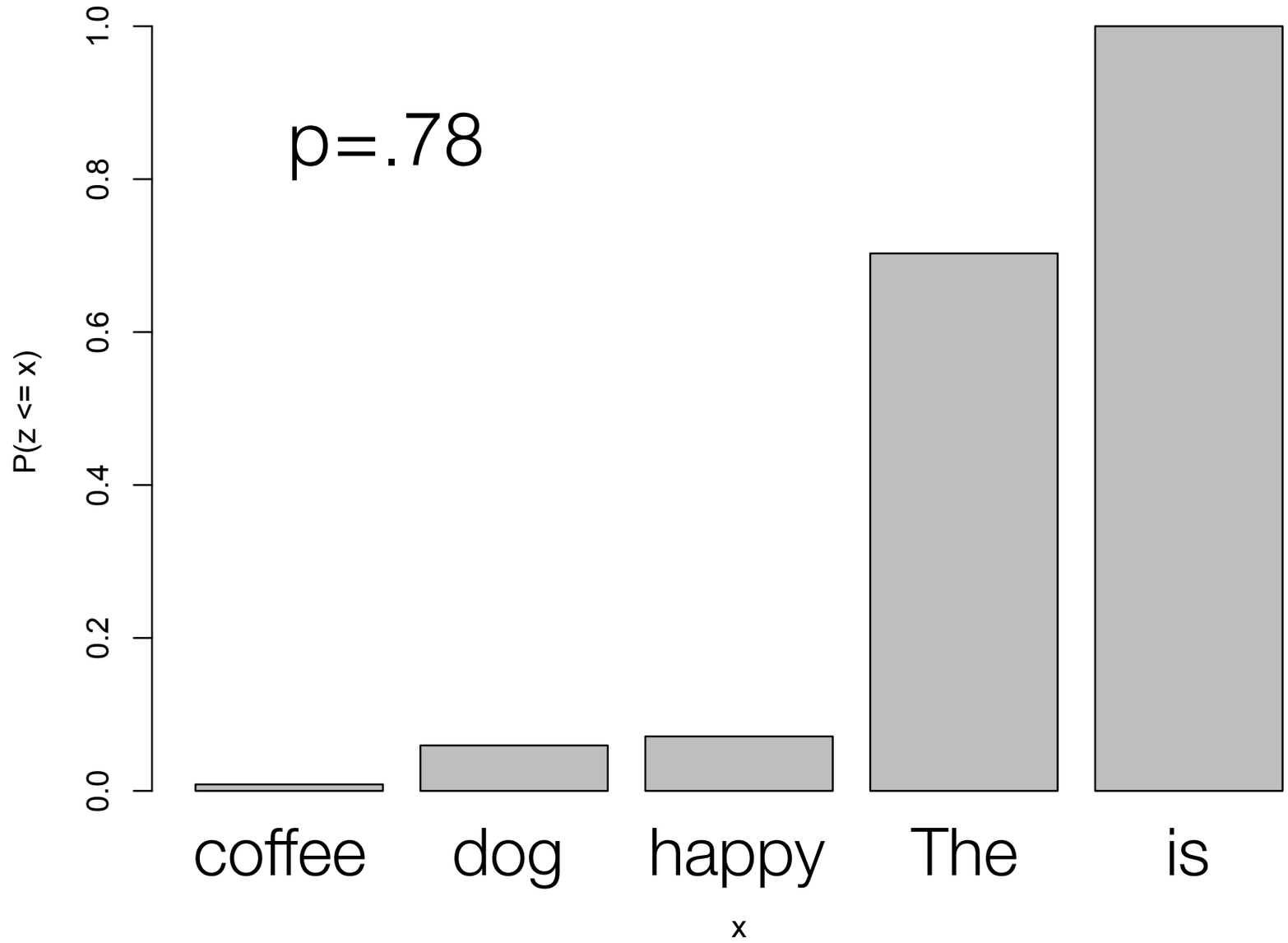
Find the point $CDF^{-1}(p)$



Sampling from a Multinomial

Sample *p* uniformly in [0,1]

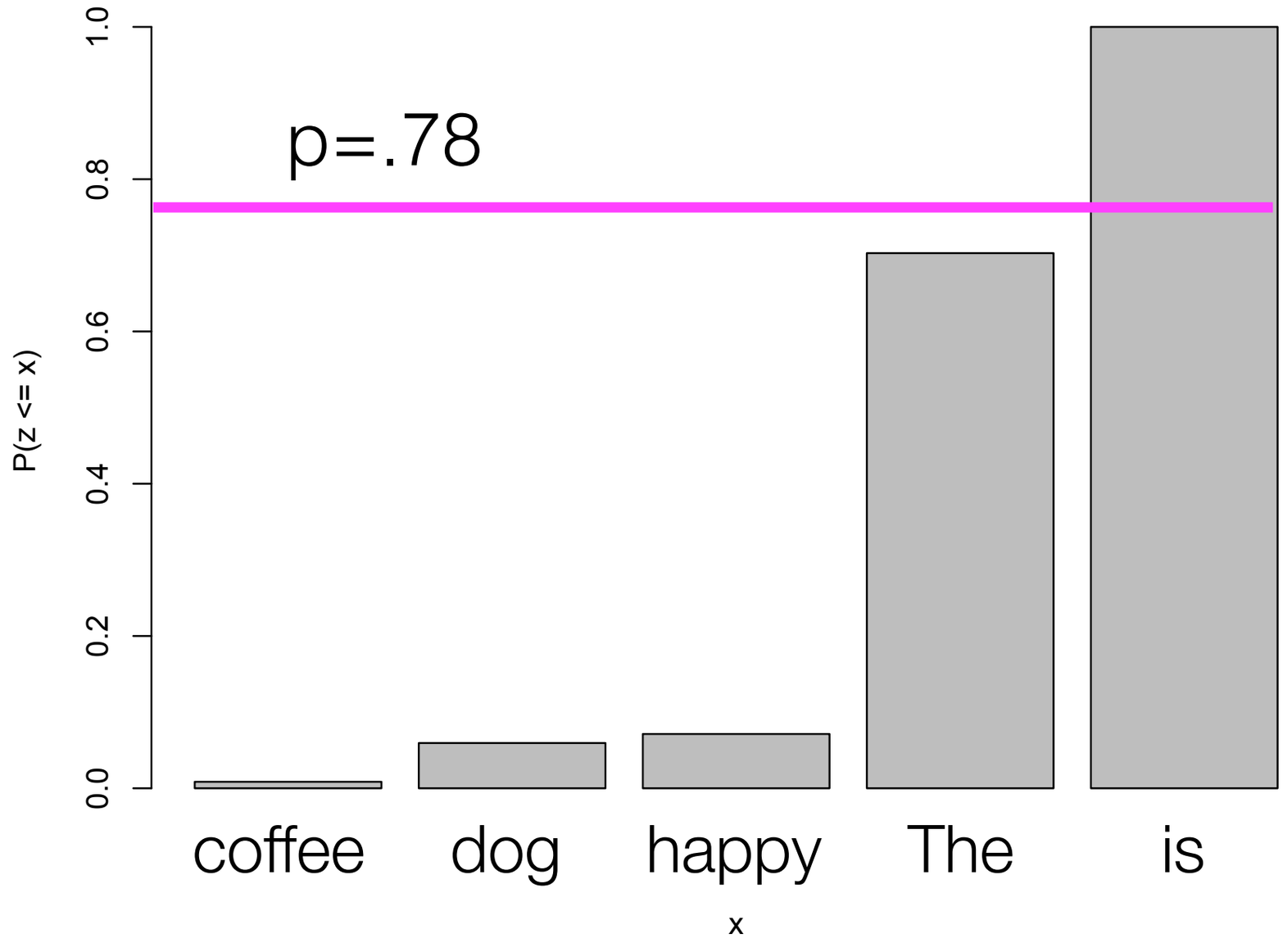
Find the point $CDF^{-1}(p)$



Sampling from a Multinomial

Sample *p* uniformly in [0,1]

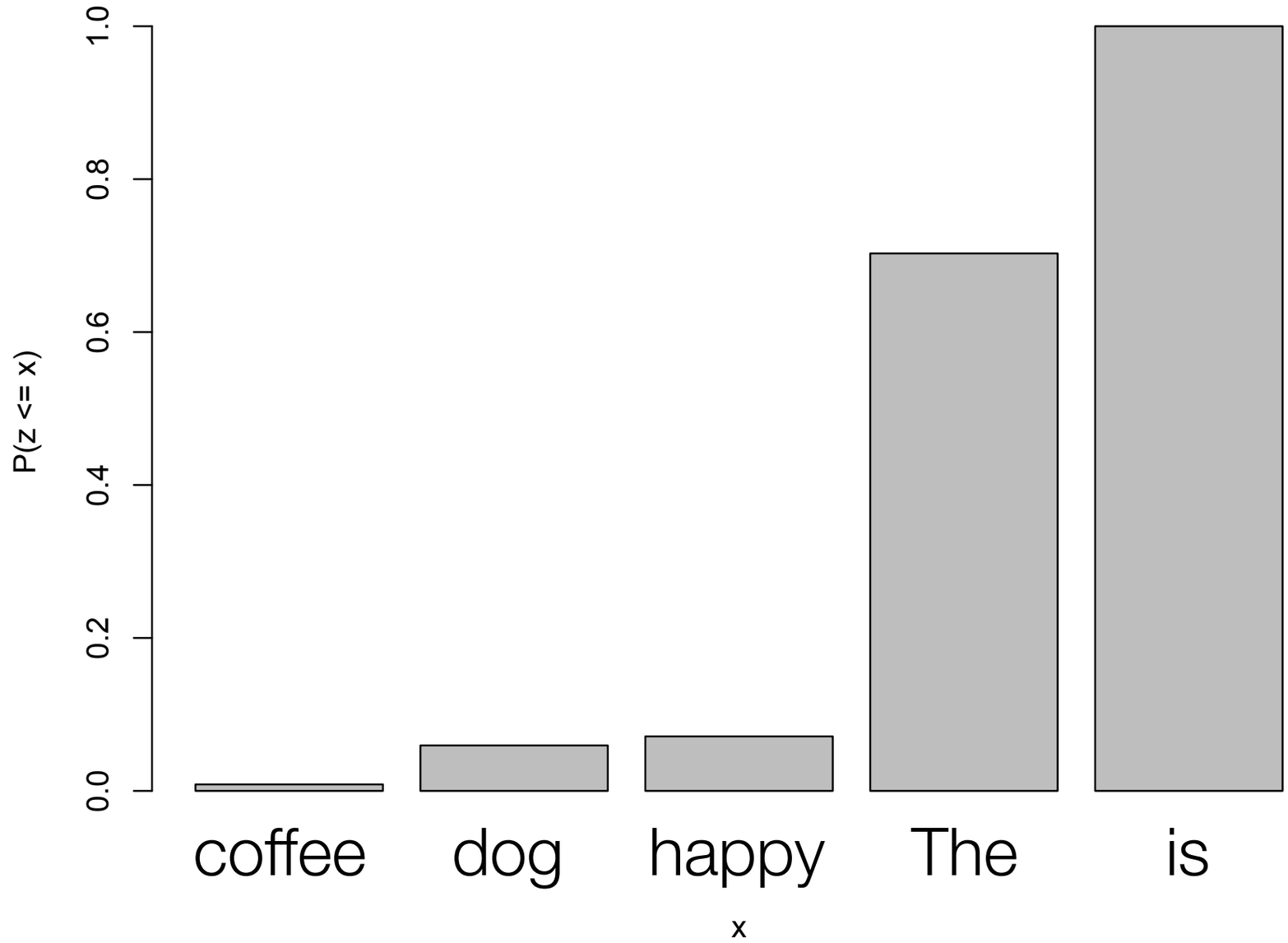
Find the point CDF^{-1}(p)



Sampling from a Multinomial

Sample *p* uniformly in [0,1]

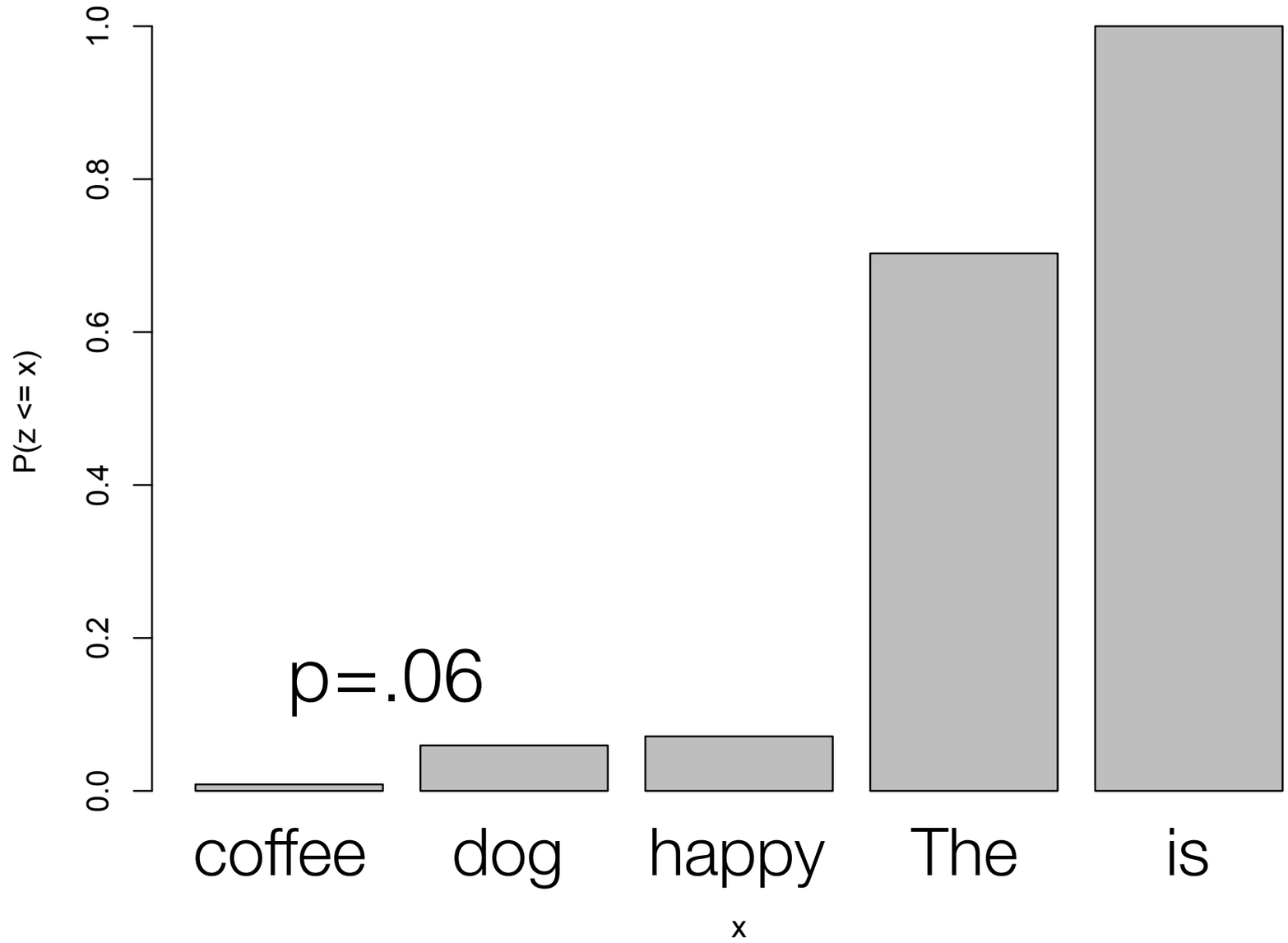
Find the point $CDF^{-1}(p)$



Sampling from a Multinomial

Sample *p* uniformly in [0,1]

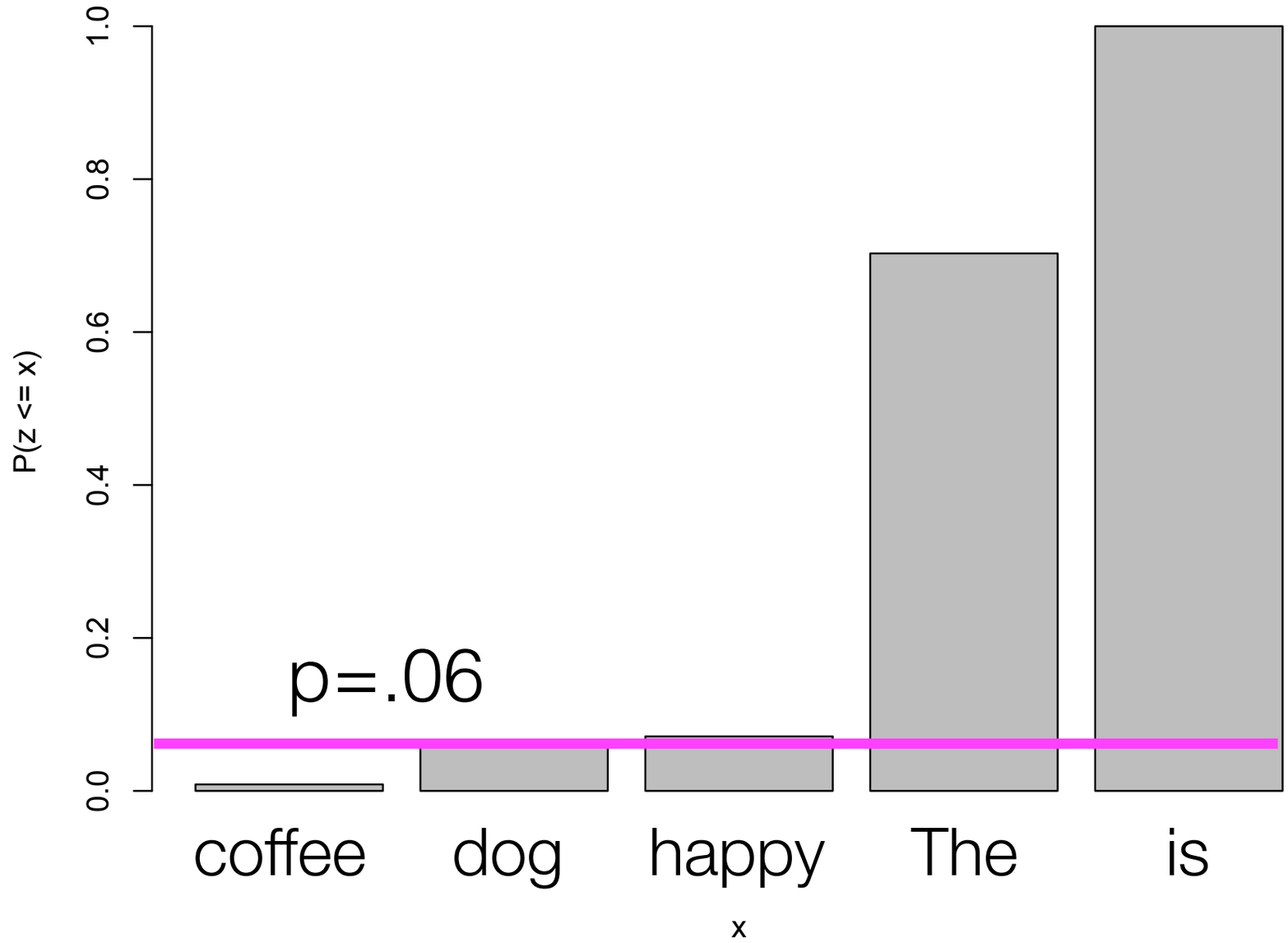
Find the point $CDF^{-1}(p)$



Sampling from a Multinomial

Sample *p* uniformly in [0,1]

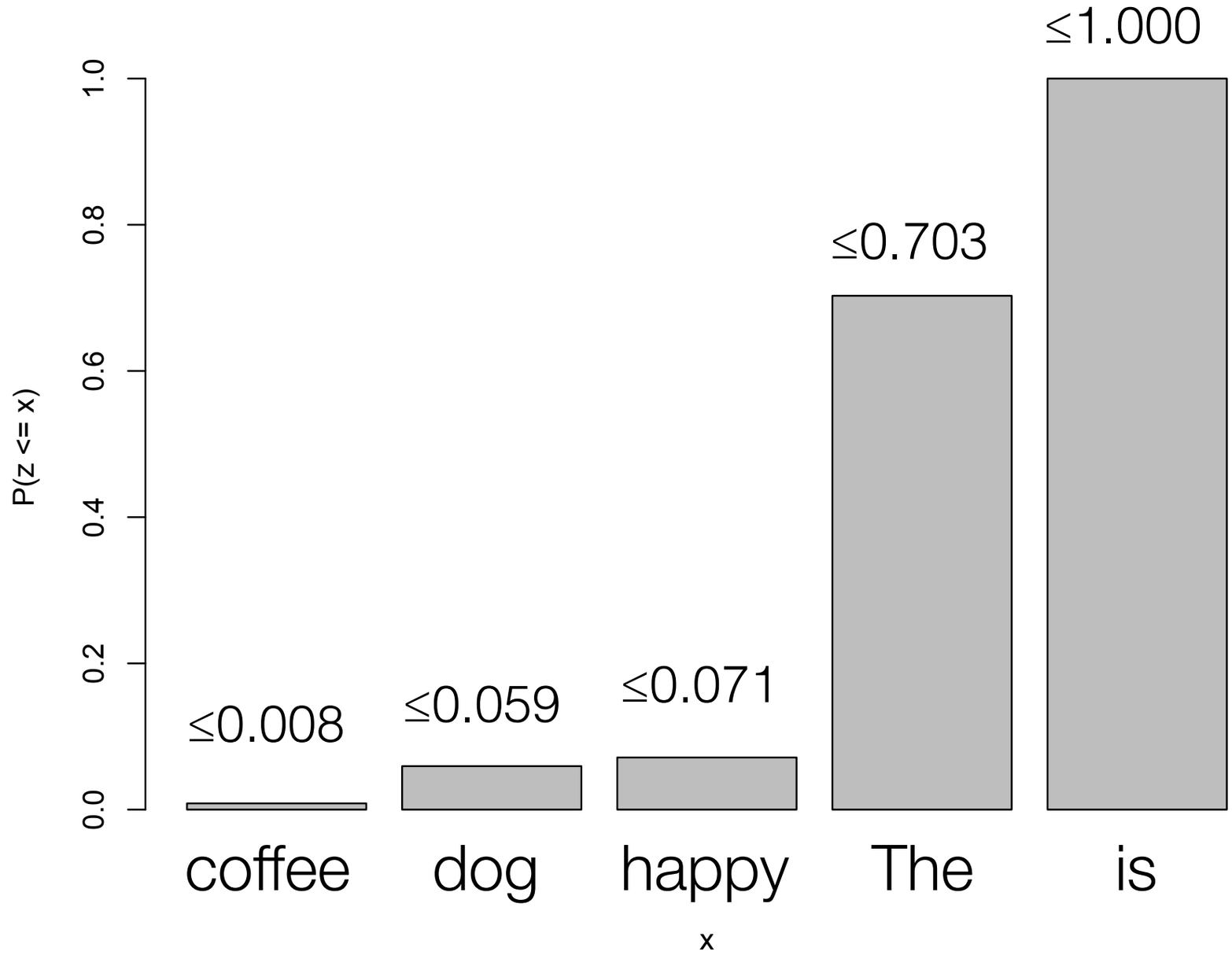
Find the point $CDF^{-1}(p)$



Sampling from a Multinomial

Sample *p* uniformly in [0,1]

Find the point $CDF^{-1}(p)$



Generating

- As we sample, the words we generate form the new context we condition on

context1	context2	generated word
START	START	The
START	The	dog
The	dog	walked
dog	walked	in

Sampling from a unigram model: p(w)

- the around, she They I blue talking “Don’t to and little come of

- on fallen used there. young people to Lázaro

- of the

- the of of never that ordered don't avoided to complaining.

- words do had men flung killed gift the one of but thing seen I plate Bradley was by small Kingmaker.

Sampling from a bigram Model: $p(w_i|w_{i-1})$

- “What the way to feel where we’re all those ancients called me one of the Council member, and smelled Tales of like a Korps peaks.”
- Tuna battle which sold or a monocle, I planned to help and distinctly.
- “I lay in the canoe ”
- She started to be able to the blundering collapsed.
- “Fine.”

Sampling from a trigram Model: $p(w_i|w_{i-1},w_{i-2})$

- “I’ll worry about it.”
- Avenue Great-Grandfather Edgeworth hasn’t gotten there.
- “If you know what. It was a photograph of seventeenth-century flourishin’ To their right hands to the fish who would not care at all. Looking at the clock, ticking away like electronic warnings about wonderfully SAT ON FIFTH
- Democratic Convention in rags soaked and my past life, I managed to wring your neck a boss won’t so David Pritchet giggled.
- He humped an argument but her bare He stood next to Larry, these days it will have no trouble Jay Grayer continued to peer around the Germans weren’t going to faint in the

Sampling from a 4gram Model: $p(w_i|w_{i-1},w_{i-2},w_{i-3})$

- Our visitor in an idiot sister shall be blotted out in bars and flirting with curly black hair right marble, wallpapered on screen credit.”

- You are much instant coffee ranges of hills.

- Madison might be stored here and tell everyone about was tight in her pained face was an old enemy, trading-posts of the outdoors watching Anyog extended On my lips moved feebly.

- said.

- “I’m in my mind, threw dirt in an inch,’ the Director.

N-gram models

- We can extend to 5-grams, 10-grams, etc
  - But only under certain conditions (why?)

N-gram models

- We can extend to 5-grams, 10-grams, etc
  - But only under certain conditions (why?)
- In general this is an insufficient model of language
  - because language has **long-distance dependencies**:

N-gram models

- We can extend to 5-grams, 10-grams, etc
  - But only under certain conditions (why?)
- 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."

N-gram models

- We can extend to 5-grams, 10-grams, etc
  - But only under certain conditions (why?)
- 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

Evaluation

How do we know if our language model learned anything?

Evaluation: How good is our model?

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

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

- 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"

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!

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

Evaluation

- The best evaluation metrics are external — how does a better language model influence the application you care about?

- Speech recognition (word error rate), machine translation (BLEU score), topic models (sensemaking)

Extrinsic evaluation of N-gram models

- Best evaluation for comparing models A and B

Extrinsic evaluation of N-gram models

- Best evaluation for comparing models A and B
  - Put each model in a task
    - spelling corrector, speech recognizer, MT system

Extrinsic evaluation of N-gram models

- Best evaluation for comparing models A and B
  - Put each model in a task
    - spelling corrector, speech recognizer, MT system
  - Run the task, get an accuracy for A and for B

Extrinsic evaluation of N-gram models

- Best evaluation for comparing models A and B
  - Put each model in a task
    - spelling corrector, speech recognizer, MT system
  - Run the task, get an accuracy for A and for B
    - How many misspelled words corrected properly

Extrinsic evaluation of N-gram models

- Best evaluation for comparing models A and B
  - Put each model in a task
    - spelling corrector, speech recognizer, MT system
  - Run the task, get an accuracy for A and for B
    - How many misspelled words corrected properly
    - How many words translated correctly

Extrinsic evaluation of N-gram models

- Best evaluation for comparing models A and B
  - Put each model in a task
    - spelling corrector, speech recognizer, MT system
  - Run the task, get an accuracy for A and for B
    - How many misspelled words corrected properly
    - How many words translated correctly
  - Compare accuracy for A and B

Intrinsic Evaluation

- A good language model should judge unseen real language to have high probability
- Perplexity = inverse probability of test data, averaged by word.
- To be reliable, the test data must be truly unseen, including knowledge of its vocabulary (could have unseen words!).

perplexity = $\sqrt[N]{\frac{1}{P(w_1, \ldots, w_n)}}$

Experiment design

	training	development	testing
size	80%	10%	10%
purpose	training models	model selection; hyperparameter tuning	evaluation; never look at it until the very end

Evaluation

$$\log P(w_1, \ldots, w_n) = \sum_i^N \log P(w_i)$$

$$\frac{1}{N} \sum_i^N \log P(w_i)$$

perplexity = $$\exp\left(-\frac{1}{N} \sum_i^N \log P(w_i)\right)$$

Perplexity

trigram model
(second-order markov)

$$\exp\left(-\frac{1}{N}\sum_{i}^{N}\log P(w_i \mid w_{i-2}, w_{i-1})\right)$$

Perplexity

Model	Unigram	Bigram	Trigram
Perplexity	962	170	109

SLP3 4.3

Addressing Data Sparisty

unigram

$$\prod_{i}^{n} P(w_i) \times P(STOP)$$

bigram

$$\prod_{i}^{n} P(w_i \mid w_{i-1}) \times P(STOP \mid w_n)$$

trigram

$$\prod_{i}^{n} P(w_i \mid w_{i-2}, w_{i-1}) \times P(STOP \mid w_{n-1}, w_n)$$

Addressing Data Sparisty

unigram

$$\prod_{i}^{n} P(w_i) \times P(STOP)$$

bigram

$$\prod_{i}^{n} P(w_i \mid w_{i-1}) \times P(STOP \mid w_n)$$

trigram

$$\prod_{i}^{n} P(w_i \mid w_{i-2}, w_{i-1}) \times P(STOP \mid w_{n-1}, w_n)$$

Could we just keep extending our n-grams?

Addressing Data Sparisty

unigram

$$\prod_{i}^{n} P(w_i) \times P(STOP)$$

bigram

$$\prod_{i}^{n} P(w_i \mid w_{i-1}) \times P(STOP \mid w_n)$$

trigram

$$\prod_{i}^{n} P(w_i \mid w_{i-2}, w_{i-1}) \times P(STOP \mid w_{n-1}, w_n)$$

Could we just keep extending our n-grams?

Maximum likelihood estimate

$$\frac{c(w_i)}{N}$$

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

$$\frac{c(w_{i-2}, w_{i-1}, w_i)}{c(w_{i-2}, w_{i-1})}$$

Smoothing

- When estimating a language model, we're relying on the data we've observed in a training corpus.

- Training data is a small (and biased) sample of the creativity of language.

Berkeley Restaurant Project

- can you tell me about any good cantonese restaurants close by
- mid priced that food is what i'm looking for
- tell me about chez pansies
- can you give me a listing of the kinds of food that are available
- i'm looking for a good place to eat breakfast
- when is caffe venezia open during the day

Data sparsity

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Figure 4.1 Bigram counts for eight of the words (out of $V = 1446$) in the Berkeley Restaurant Project corpus of 9332 sentences. Zero counts are in gray.

SLP3 4.1

$$\prod_{i}^{n} P(w_i \mid w_{i-1}) \times P(\text{STOP} \mid w_n)$$

- As in Naive Bayes, $P(w_i) = 0$ causes $P(w) = 0$. (Perplexity?)

Smoothing in NB

- One solution: add a little probability mass to every element.

maximum likelihood estimate

$$P(x_i \mid y) = \frac{n_{i,y}}{n_y}$$

$n_{i,y}$ = count of word i in class y
n_y = number of words in y
V = size of vocabulary

smoothed estimates

$$P(x_i \mid y) = \frac{n_{i,y} + \alpha}{n_y + V\alpha}$$

same α for all x_i

$$P(x_i \mid y) = \frac{n_{i,y} + \alpha_i}{n_y + \sum_{j=1}^{V} \alpha_j}$$

possibly different α for each x_i

Additive smoothing

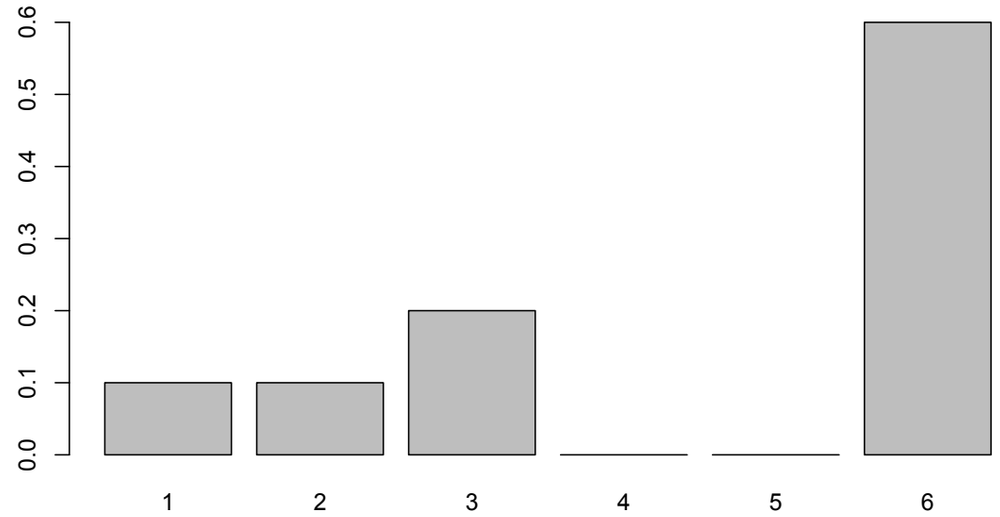
Laplace smoothing:
α = 1

$$P(w_i) = \frac{c(w_i) + \alpha}{N + V\alpha}$$

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

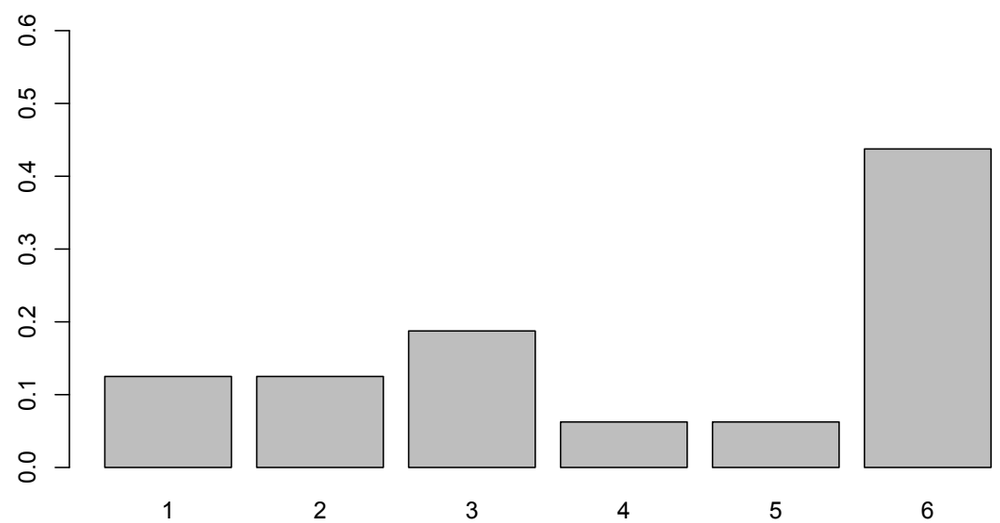
Smoothing

MLE



Smoothing is the re-allocation of probability mass

smoothing with α =1



Smoothing

- How can best re-allocate probability mass?

Stanley F. Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Center for Research in Computing Technology, Harvard University, 1998.

Interpolation

- As ngram order rises, we have the potential for higher precision but also higher variability in our estimates.

- A linear interpolation of any two language models p and q (with $\lambda \in [0,1]$) is also a valid language model.

$$\lambda p + (1 - \lambda) q$$

p = the web

q = political speeches

Interpolation

$$\lambda p + (1 - \lambda)q$$

p = Nihilist text

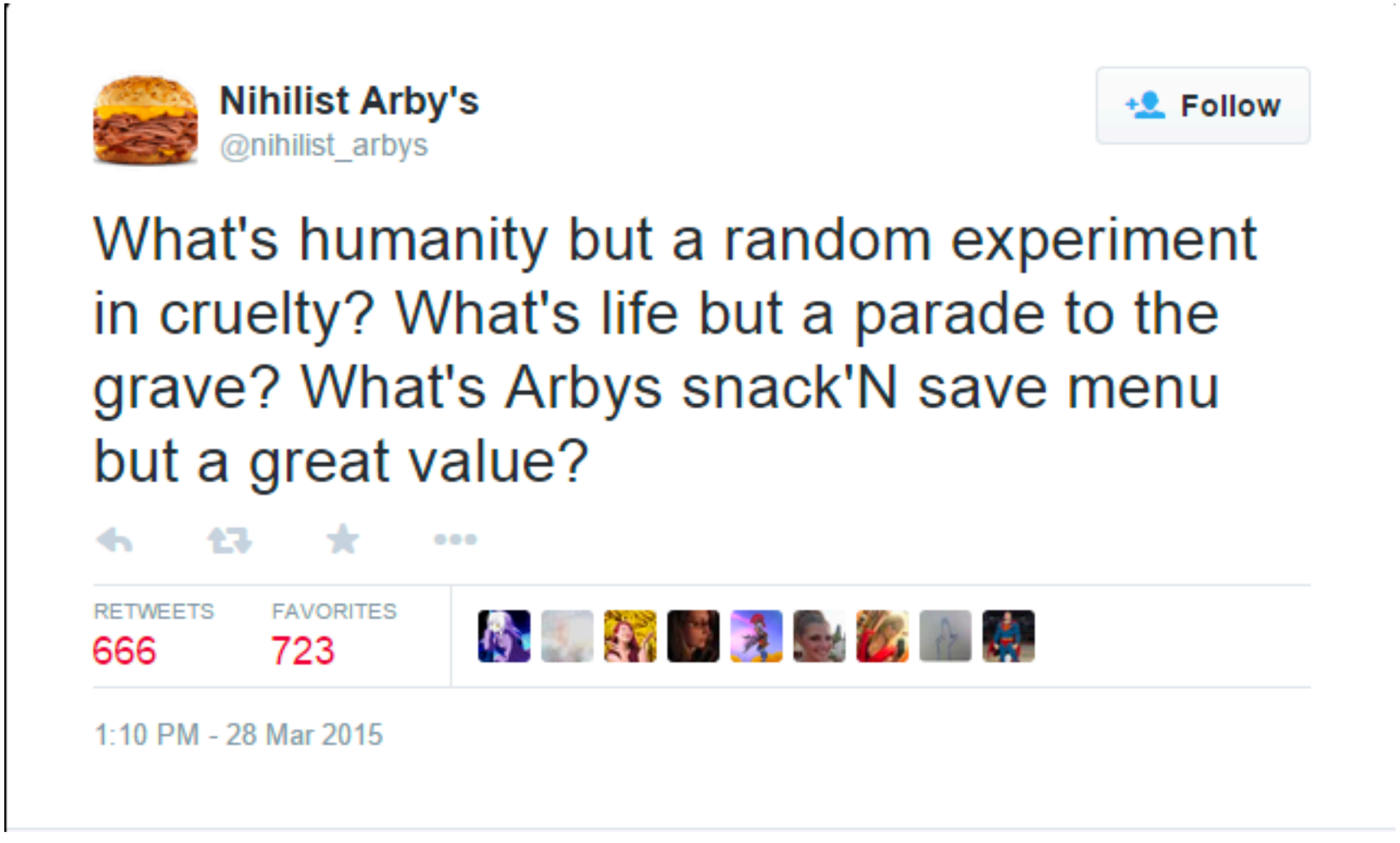
q = Arby's ads

Interpolation

$$\lambda p + (1 - \lambda) q$$

p = Nihilist text

q = Arby's ads



Interpolation

- We can use this fact to make higher-order language models more robust.

$$P(w_i \mid w_{i-2}, w_{i-1}) = \lambda_1 P(w_i \mid w_{i-2}, w_{i-1}) + \lambda_2 P(w_i \mid w_{i-1}) + \lambda_3 P(w_i)$$

$$\lambda_1 + \lambda_2 + \lambda_3 = 1$$

Interpolation

- How do we pick the best values of λ?
  - Grid search over development corpus
  - Expectation-Maximization algorithm (treat as missing parameters to be estimated to maximize the probability of the data we see).

Kneser-Ney smoothing

Kneser-Ney smoothing

- Intuition: When backing off to a lower-order ngram, maybe the overall ngram frequency is not our best guess.

Kneser-Ney smoothing

- Intuition: When backing off to a lower-order ngram, maybe the overall ngram frequency is not our best guess.

I can’t see without my reading _____

P(“Francisco”) > P(“glasses”)

Kneser-Ney smoothing

- Intuition: When backing off to a lower-order ngram, maybe the overall ngram frequency is not our best guess.

I can't see without my reading _____

P("Francisco") > P("glasses")

- *Francisco* is more frequent, but shows up in fewer unique bigrams ("San Francisco") — so we shouldn't expect it in new contexts; *glasses*, however, does show up in many different bigrams

Kneser-Ney smoothing

- Intuition: estimate how likely a word is to show up in a new continuation?
- How many different bigram types does a word type *w* show up in (normalized by all bigram types that are seen)

continuation probability: of all bigram types in training data, how many is w the suffix for?

$$\frac{|v \in \mathcal{V} : c(v, w) > 0|}{|v', w' \in \mathcal{V} : c(v', w') > 0|}$$

$$P_{KN}(v) = \frac{|v \in \mathcal{V} : c(v,w) > 0|}{|v', w' \in \mathcal{V} : c(v',w') > 0|}$$

$P_{KN}(v)$ is the continuation probability for the unigram v

This is computed using the frequency with which it appears as the suffix in distinct bigram types

Kneser-Ney smoothing

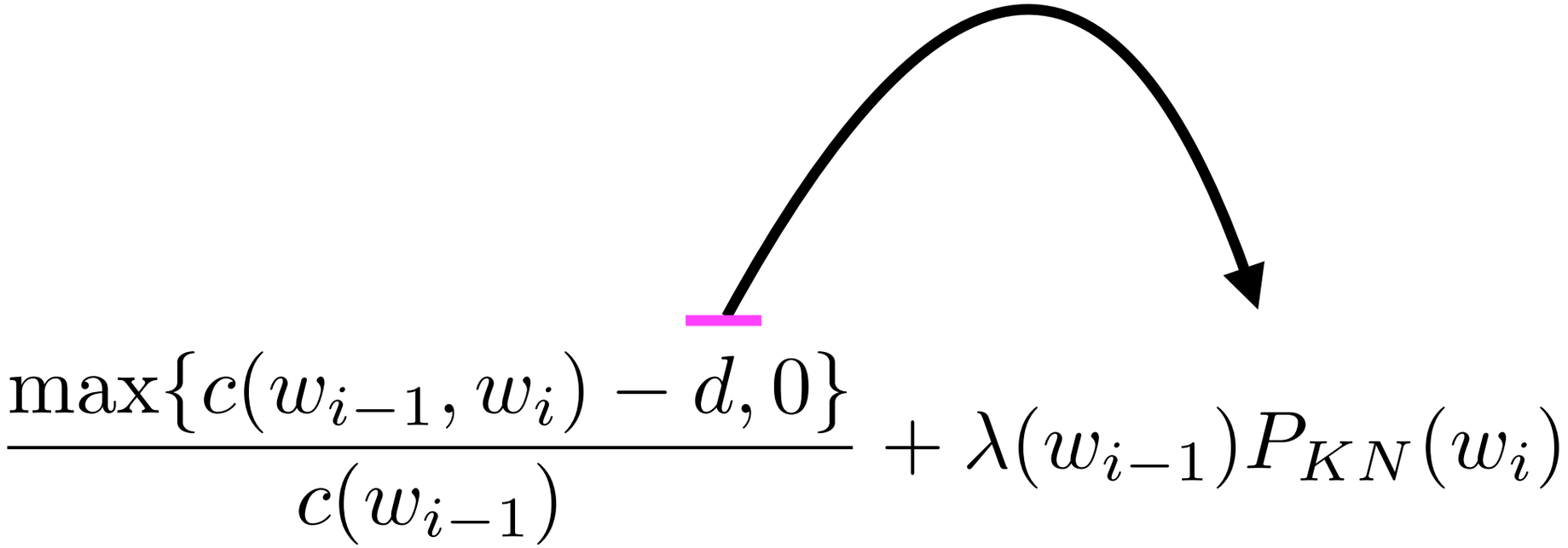
discounted mass

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

discounted bigram probability

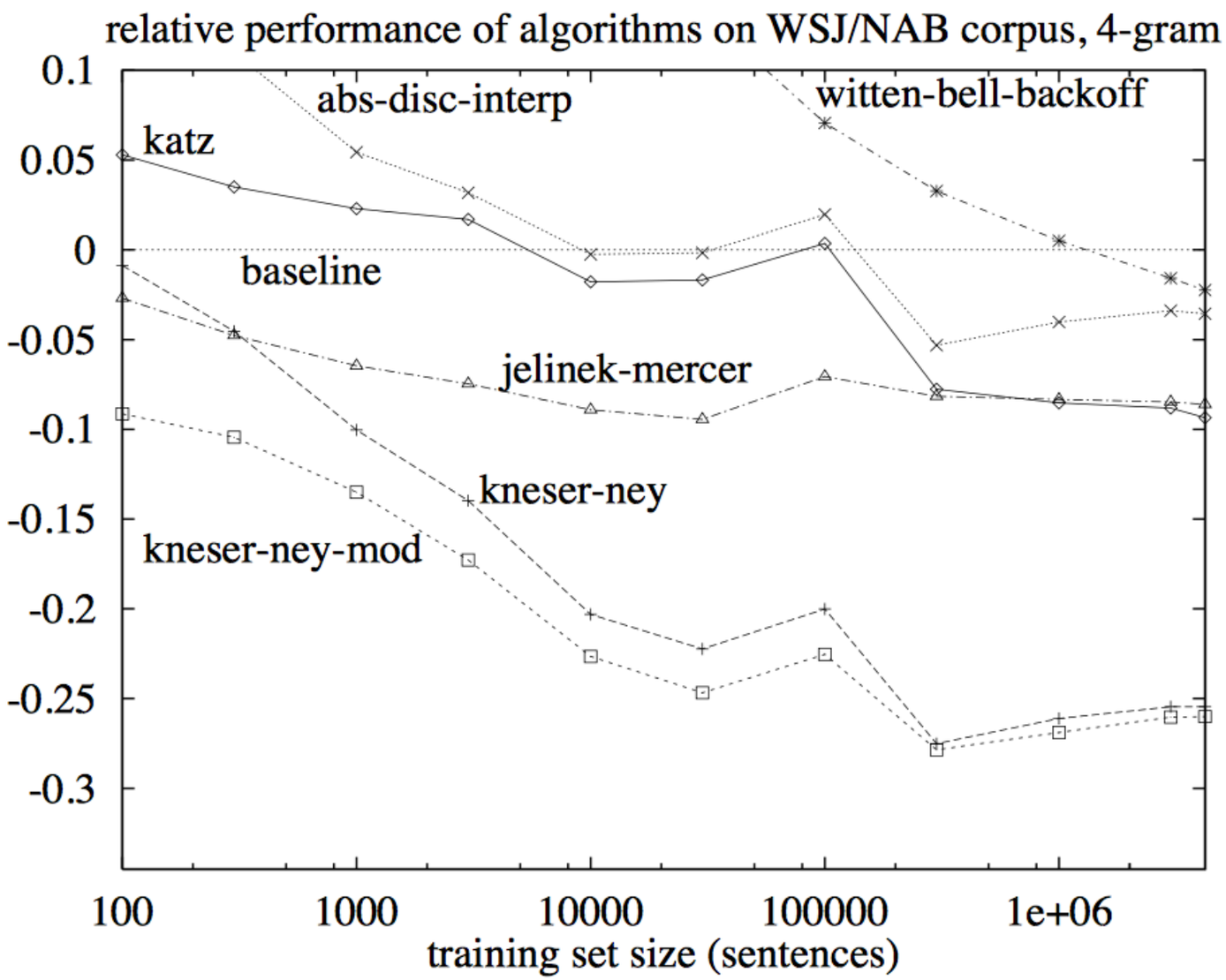
continuition probability

we'll move all of the mass we subtracted here over to this side



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

and distribute it according to the continuation probability



Stanley F. Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Center for Research in Computing Technology, Harvard University, 1998.

“Stupid backoff”

if full sequence observed

$$S(w_i \mid w_{i-k+1}, \ldots, w_{i-1}) = \frac{c(w_{i-k+1}, \ldots, w_i)}{c(w_{i-k+1}, \ldots, w_{i-1})}$$

No discounting here, just back off to lower order ngram if the higher order is not observed.

Cheap to calculate; works almost as well as KN when there is a lot of data

otherwise

$$= \lambda S(w_i \mid w_{i-k+2}, \ldots, w_{i-1})$$

Brants et al. (2007), “Large Language Models in Machine Translation”

Unknown words: Open versus closed vocabulary tasks

Unknown words: Open versus closed vocabulary tasks

- If we know all the words in advanced, the vocabulary V is fixed.
  -> Closed vocabulary task

Unknown words: Open versus closed vocabulary tasks

- If we know all the words in advanced, the vocabulary V is fixed.
  -> Closed vocabulary task
- Often we don't know this, Out Of Vocabulary = OOV words ->
  Open vocabulary task

Unknown words: Open versus closed vocabulary tasks

- If we know all the words in advanced, the 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>

Unknown words: Open versus closed vocabulary tasks

- If we know all the words in advanced, the 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

Unknown words: Open versus closed vocabulary tasks

- If we know all the words in advanced, the 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

Unknown words: Open versus closed vocabulary tasks

- If we know all the words in advanced, the 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>

Unknown words: Open versus closed vocabulary tasks

- If we know all the words in advanced, the 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

Unknown words: Open versus closed vocabulary tasks

- If we know all the words in advanced, the 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 testing time

Unknown words: Open versus closed vocabulary tasks

- If we know all the words in advanced, the 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 testing time
  - If text input: Use UNK probabilities for any word not in training

Advanced UNKing

I went ____ with my friends

Advanced UNKing

I went ____ with my friends

spelunking

Advanced UNKing

I went ____ with my friends

spelunking

quiddiching

Advanced UNKing

I went ____ with my friends

spelunking

quiddiching

-ing

Advanced UNKing

I went ____ with my friends

spelunking

quiddiching

-ing

- Words have regular morphology that we expect, even when we don't know what the word means
- Most systems don't use a single UNK token; UNK+suffix or prefix+UNK

Advanced UNKing

I went ____ with my friends

spelunking

quiddiching

-ing

- Words have regular morphology that we expect, even when we don't know what the word means
- Most systems don't use a single UNK token; UNK+suffix or prefix+UNK

```
def unkify(ws):
  uk = 'unk'
  sz = len(ws)-1
  if ws[0].isupper():
    uk = 'c' + uk
  if ws[0].isdigit() and ws[sz].isdigit():
    uk = uk + 'n'
  elif sz <= 2:
    pass
  elif ws[sz-2:sz+1] == 'ing':
    uk = uk + 'ing'
  elif ws[sz-1:sz+1] == 'ed':
    uk = uk + 'ed'
  elif ws[sz-1:sz+1] == 'ly':
    uk = uk + 'ly'
  elif ws[sz] == 's':
    uk = uk + 's'
  elif ws[sz-2:sz+1] == 'est':
    uk = uk + 'est'
  elif ws[sz-1:sz+1] == 'er':
    uk = uk + 'ER'
  elif ws[sz-2:sz+1] == 'ion':
    uk = uk + 'ion'
  elif ws[sz-2:sz+1] == 'ory':
    uk = uk + 'ory'
  elif ws[0:2] == 'un':
    uk = 'un' + uk
  elif ws[sz-1:sz+1] == 'al':
    uk = uk + 'al'
  else:
```

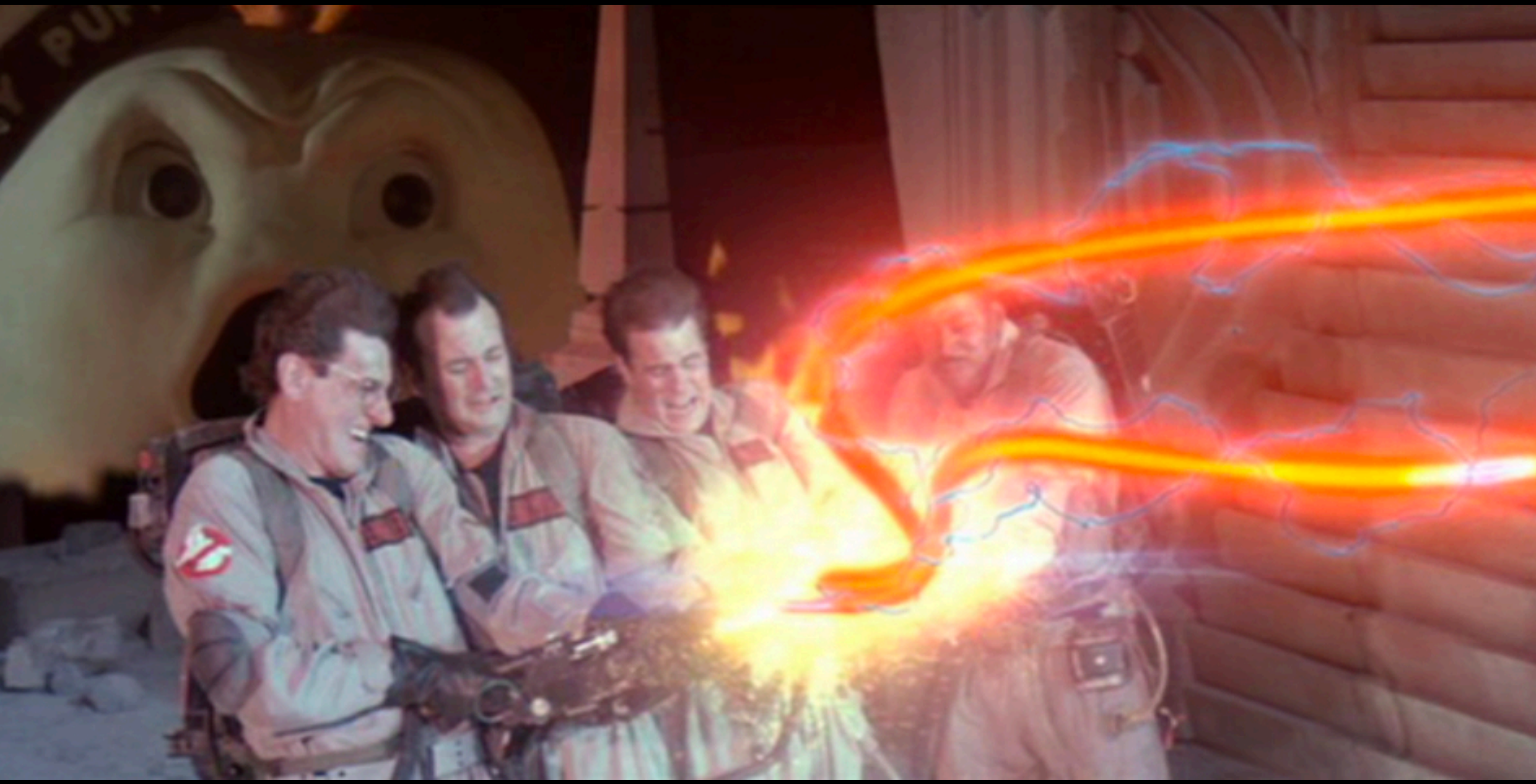


Image credit: Ghostbusters

Language Model

- Vocabulary $\mathscr{V}$ is a finite set of discrete symbols (e.g., words, characters); V = | $\mathscr{V}$ |

- $\mathscr{V}^+$ is the infinite set of sequences of symbols from $\mathscr{V}$; each sequence ends with STOP

- x $\in \mathscr{V}^+$

Language Model

- Vocabulary $\mathscr{V}$ is a finite set of discrete symbols (e.g., words, characters); V = | $\mathscr{V}$ |

- $\mathscr{V}^+$ is the infinite set of sequences of symbols from $\mathscr{V}$; each sequence ends with STOP

- x ∈ $\mathscr{V}^+$

Language modeling is the task of estimating P(w)

Language Model

arma virumque cano

Language Model

- arms man and I sing

arma virumque cano

Language Model

arma virumque cano

- arms man and I sing
- Arms, and the man I sing [Dryden]

Language Model

arma virumque cano

- arms man and I sing
- Arms, and the man I sing [Dryden]
- I sing of arms and a man

Language Model

arma virumque cano

- arms man and I sing
- Arms, and the man I sing [Dryden]
- I sing of arms and a man

- When we have choices to make about different ways to say something, a language models gives us a formal way of operationalizing their fluency
  (in terms of how likely they are exist in the language)

Markov assumption

bigram model
(first-order markov)

$$\prod_{i}^{n} P(w_i \mid w_{i-1}) \times P(\text{STOP} \mid w_n)$$

trigram model
(second-order markov)

$$\prod_{i}^{n} P(w_i \mid w_{i-2}, w_{i-1})$$

$$\times P(\text{STOP} \mid w_{n-1}, w_n)$$

Language Modeling as Classification

A mapping *h* from input data x (drawn from instance space $\mathcal{X}$) to a label (or labels) y from some enumerable output space $\mathcal{Y}$

Language Modeling as Classification

A mapping *h* from input data x (drawn from instance space $\mathcal{X}$) to a label (or labels) y from some enumerable output space $\mathcal{Y}$

$\mathcal{Y}$ = {the, of, a, dog, iphone, …}

Language Modeling as Classification

A mapping *h* from input data x (drawn from instance space $\mathcal{X}$) to a label (or labels) y from some enumerable output space $\mathcal{Y}$

$\mathcal{Y}$ = {the, of, a, dog, iphone, …}

x = (context words)
y = word

Logistic regression

$$P(y = 1 \mid x, \beta) = \frac{1}{1 + \exp\left(-\sum_{i=1}^{F} x_i \beta_i\right)}$$

output space $\mathcal{Y} = \{0, 1\}$

x = feature vector

Feature	Value
the	0
and	0
bravest	0
love	0
loved	0
genius	0
not	0
fruit	1
BIAS	1

β = coefficients

Feature	β
the	0.01
and	0.03
bravest	1.4
love	3.1
loved	1.2
genius	0.5
not	-3.0
fruit	-0.8
BIAS	-0.1

Logistic regression

$$P(Y = 1 \mid X = x; \beta) = \frac{\exp(x^\top \beta)}{1 + \exp(x^\top \beta)}$$

$$= \frac{\exp(x^\top \beta)}{\exp(x^\top 0) + \exp(x^\top \beta)}$$

Multinomial Logistic Regression

$$P(Y = y \mid X = x; \beta) = \frac{\exp(x^\top \beta_y)}{\sum_{y' \in \mathcal{Y}} \exp(x^\top \beta_{y'})}$$

output space $\mathcal{Y} = \{1, \ldots, K\}$

x = feature vector

Feature	Value
the	0
and	0
bravest	0
love	0
loved	0
genius	0
not	0
fruit	1
BIAS	1

β = coefficients

Feature	β_1	β_2	β_3	β_4	β_5
the	1.33	-0.80	-0.54	0.87	0
and	1.21	-1.73	-1.57	-0.13	0
bravest	0.96	-0.05	0.24	0.81	0
love	1.49	0.53	1.01	0.64	0
loved	-0.52	-0.02	2.21	-2.53	0
genius	0.98	0.77	1.53	-0.95	0
not	-0.96	2.14	-0.71	0.43	0
fruit	0.59	-0.76	0.93	0.03	0
BIAS	-1.92	-0.70	0.94	-0.63	0

Language Model

- We can use multi class logistic regression for language modeling by treating the vocabulary as the output space (SLP3 c4)

$$\mathcal{Y} = \mathcal{V}$$

Language Model

- We can use multi class logistic regression for language modeling by treating the vocabulary as the output space (SLP3 c4)

$$\mathcal{Y} = \mathcal{V}$$

Output classes

Vocabulary

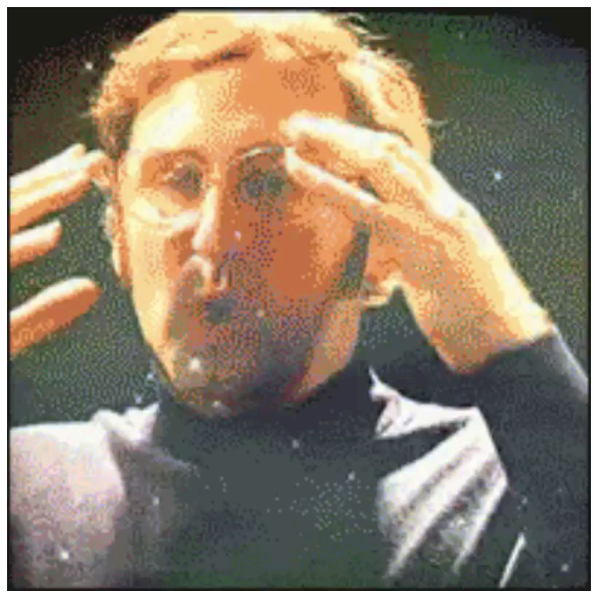
Language Model

- We can use multi class logistic regression for language modeling by treating the vocabulary as the output space (SLP3 c4)

$$\mathcal{Y} = \mathcal{V}$$

Output classes

Vocabulary



Unigram LM

- A unigram language model here would have just one feature: a bias term.

Feature	β_{the}	β_{of}	β_{a}	β_{dog}	β_{iphone}
BIAS	-1.92	-0.70	0.94	-0.63	0

Bigram LM

Feature	Value	β_{the}	β_{of}	β_{a}	β_{dog}	β_{iphone}
w_{i-1}=the	1	1.33	-0.80	-0.54	0.87	0
w_{i-1}=and	0	1.21	-1.73	-1.57	-0.13	0
w_{i-1}=brave	0	0.96	-0.05	0.24	0.81	0
w_{i-1}=love	0	1.49	0.53	1.01	0.64	0
w_{i-1}=loved	0	-0.52	-0.02	2.21	-2.53	0
w_{i-1}=geniu	0	0.98	0.77	1.53	-0.95	0
w_{i-1}=not	0	-0.96	2.14	-0.71	0.43	0
w_{i-1}=fruit	0	0.59	-0.76	0.93	0.03	0
BIAS	1	-1.92	-0.70	0.94	-0.63	0

$$P(w_i = dog \mid w_{i-1} = the)$$

Feature	Value	β_{the}	β_{of}	β_{a}	β_{dog}	β_{iphone}
w_{i-1}=the	1	1.33	-0.80	-0.54	0.87	0
w_{i-1}=and	0	1.21	-1.73	-1.57	-0.13	0
w_{i-1}=brave	0	0.96	-0.05	0.24	0.81	0
w_{i-1}=love	0	1.49	0.53	1.01	0.64	0
w_{i-1}=loved	0	-0.52	-0.02	2.21	-2.53	0
w_{i-1}=geniu	0	0.98	0.77	1.53	-0.95	0
w_{i-1}=not	0	-0.96	2.14	-0.71	0.43	0
w_{i-1}=fruit	0	0.59	-0.76	0.93	0.03	0
BIAS	1	-1.92	-0.70	0.94	-0.63	0

Trigram LM

$$P(w_i = dog \mid w_{i-2} = and, w_{i-1} = the)$$

Feature	Value
w_{i-2}=the ^ w_{i-1}=the	0
w_{i-2}=and ^ w_{i-1}=the	1
w_{i-2}=bravest ^ w_{i-1}=the	0
w_{i-2}=love ^ w_{i-1}=the	0
w_{i-2}=loved ^ w_{i-1}=the	0
w_{i-2}=genius ^ w_{i-1}=the	0
w_{i-2}=not ^ w_{i-1}=the	0
w_{i-2}=fruit ^ w_{i-1}=the	0
BIAS	1

Parameters

Parameters

- How many parameters do we have with a log-linear trigram language model?

Parameters

- How many parameters do we have with a log-linear trigram language model?

- How many do we have with a generative trigram LM?

Smoothing

$$P(w_i = dog \mid w_{i-2} = and, w_{i-1} = the)$$

Feature	Value
w_{i-2}=the ^ w_{i-1}=the	0
w_{i-2}=and ^ w_{i-1}=the	1
w_{i-2}=bravest ^ w_{i-1}=the	0
w_{i-2}=love ^ w_{i-1}=the	0
w_{i-1}=the	1
w_{i-1}=and	0
w_{i-1}=bravest	0
w_{i-1}=love	0
BIAS	1

Smoothing

$$P(w_i = dog \mid w_{i-2} = and, w_{i-1} = the)$$

second-order features

Feature	Value
w_{i-2}=the ^ w_{i-1}=the	0
w_{i-2}=and ^ w_{i-1}=the	1
w_{i-2}=bravest ^ w_{i-1}=the	0
w_{i-2}=love ^ w_{i-1}=the	0
w_{i-1}=the	1
w_{i-1}=and	0
w_{i-1}=bravest	0
w_{i-1}=love	0
BIAS	1

Smoothing

$$P(w_i = dog \mid w_{i-2} = and, w_{i-1} = the)$$

second-order features

first-order features

Feature	Value
w_{i-2}=the ^ w_{i-1}=the	0
w_{i-2}=and ^ w_{i-1}=the	1
w_{i-2}=bravest ^ w_{i-1}=the	0
w_{i-2}=love ^ w_{i-1}=the	0
w_{i-1}=the	1
w_{i-1}=and	0
w_{i-1}=bravest	0
w_{i-1}=love	0
BIAS	1

L2 regularization

$$\ell(\beta) = \underbrace{\sum_{i=1}^{N} \log P(y_i \mid x_i, \beta)}_{\text{we want this to be high}} - \underbrace{\eta \sum_{j=1}^{F} \beta_j^2}_{\text{but we want this to be small}}$$

- We can do this by changing the function we're trying to optimize by adding a penalty for having values of β that are high
- This is equivalent to saying that each β element is drawn from a Normal distribution centered on 0.
- η controls how much of a penalty to pay for coefficients that are far from 0 (optimize on development data)

L1 regularization

$$\ell(\beta) = \underbrace{\sum_{i=1}^{N} \log P(y_i \mid x_i, \beta)}_{\text{we want this to be high}} - \underbrace{\eta \sum_{j=1}^{F} |\beta_j|}_{\text{but we want this to be small}}$$

- L1 regularization encourages coefficients to be exactly 0.
- η again controls how much of a penalty to pay for coefficients that are far from 0 (optimize on development data)

Richer representations

- Log-linear models give us the flexibility of encoding richer representations of the context we are conditioning on.

- We can reason about any observations from the entire history and not just the local context.

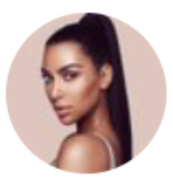
“JACKSONVILLE, Fla. — Stressed and exhausted families across the Southeast were assessing the damage from Hurricane Irma on Tuesday, even as flooding from the storm continued to plague some areas, like Jacksonville, and the worst of its wallop was being revealed in others, like the Florida Keys.

Officials in Florida, Georgia and South Carolina tried to prepare residents for the hardships of recovery from the _____”

https://www.nytimes.com/2017/09/12/us/irma-jacksonville-florida.html

“Hillary Clinton seemed to add Benghazi to her already-long list of culprits to blame for her upset loss to Donald _____”

http://www.foxnews.com/politics/2017/09/13/clinton-laments-how-benghazi-tragedy-hurt-her-politically.html

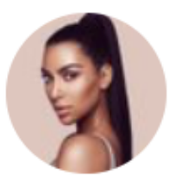


Kim Kardashian West

@KimKardashian

Follow

OMG I had to take the whole family to go see The Fault In Our Stars ✨ This movie _____

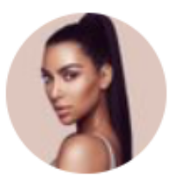


Kim Kardashian West

@KimKardashian

Follow

OMG I had to take the whole family to go see The Fault In Our Stars ✨ This movie OMG



Kim Kardashian West

@KimKardashian

Follow

OMG I had to take the whole family to go see The Fault In Our Stars ✨ This movie OMG it's literally my new Notebook 💫

“Hillary Clinton seemed to add Benghazi to her already-long list of culprits to blame for her upset loss to Donald _____”

feature classes	example
ngrams (w_{i-1}, w_{i-2}:w_{i-1}, w_{i-3}:w_{i-1})	w_{i-2}=“to”, w_i=“donald”
gappy ngrams	w_1=“hillary” and w_{i-1}=“donald”
spelling, capitalizaiton	w_{i-1} is capitalized and w_i is capitalized
class/gazetteer membership	w_{i-1} in list of names and w_i in list of names

Classes

bigram	count
black car	100
blue car	37
red car	0

bigram	count
<COLOR> car	137

Goldberg 2017

Classes

bigram	count
black car	100
blue car	37
red car	0

bigram	count
<COLOR> car	137

How do we get these classes? We'll see how to learn them in Week 8!

Goldberg 2017

Tradeoffs

- Richer representations = more parameters, higher likelihood of overfitting

- Much slower to train than estimating the parameters of a generative model

$$P(Y = y \mid X = x; \beta) = \frac{\exp(x^\top \beta_y)}{\sum_{y' \in \mathcal{Y}} \exp(x^\top \beta_{y'})}$$

http://www.greece-is.com/wp-content/uploads/2016/07/Aristotle-Freiburg.jpg

Neural Language Models

Important aside on neural networks

Important aside on neural networks

- We're going to see two types of neural networks: A feed-forward network and a recurrent neural network

Important aside on neural networks

- We're going to see two types of neural networks: A feed-forward network and a recurrent neural network
- We'll go over these types of networks in different applications / topics for the next few weeks

Important aside on neural networks

- We're going to see two types of neural networks: A feed-forward network and a recurrent neural network
- We'll go over these types of networks in different applications / topics for the next few weeks
- Please ask questions!

Count-based Language Models keep improving with more data

Count-based Language Models keep improving with more data

- But the number of n-grams also goes up with more data (the long tail of n-grams)

Count-based Language Models keep improving with more data

- But the number of n-grams also goes up with more data (the long tail of n-grams)
- Gigantic RAM requirements!

Count-based Language Models keep improving with more data

- But the number of n-grams also goes up with more data (the long tail of n-grams)
  - Gigantic RAM requirements!

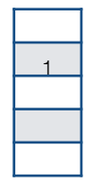
- Recent state of the art: Scalable Modified Kneser-Ney Language Model Estimation by Heafield et al.:

  > "Using one machine with 140 GB RAM for 2.8 days, we built an unpruned model on 126 billion tokens"

Feed-Forward Neural LM

Simple feed-forward multilayer perceptron
(e.g., one hidden layer)

input x = vector concatenation of a conditioning context of fixed size k

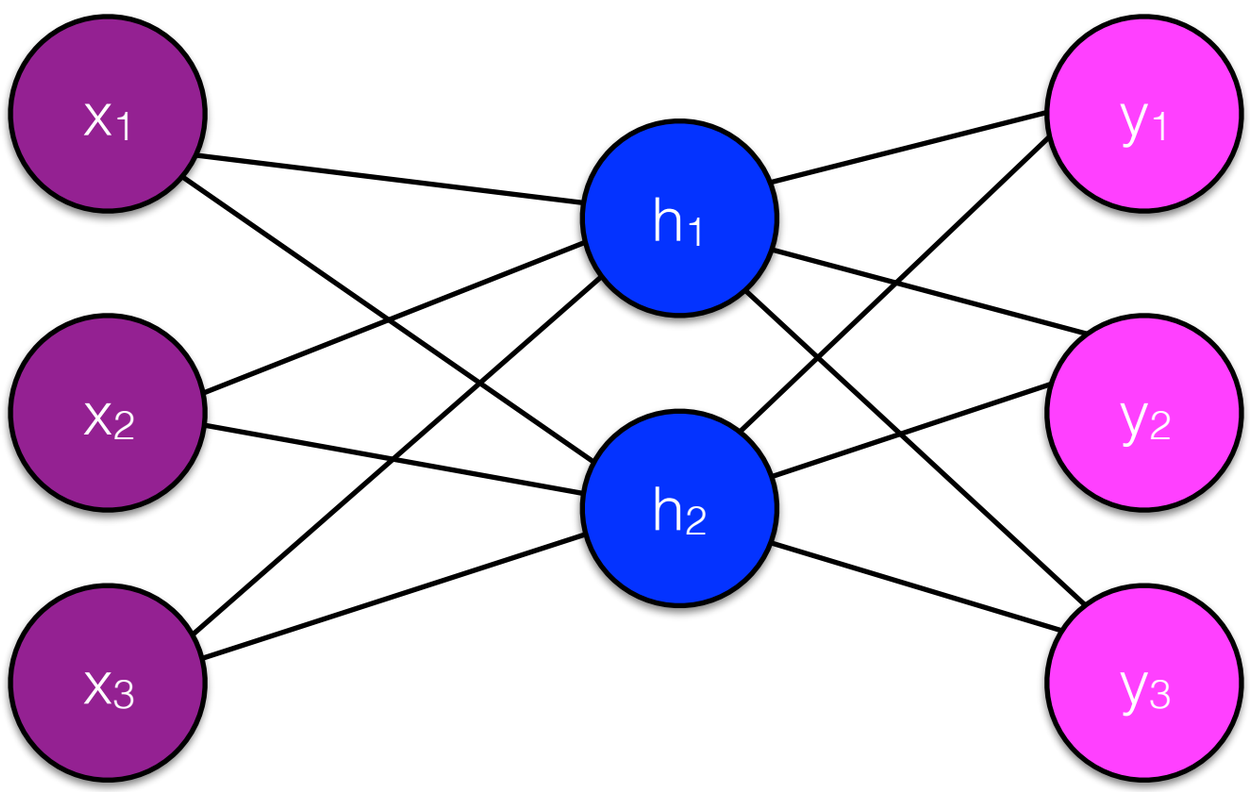


$$x = [v(w_1); \ldots; v(w_k)]$$

Bengio et al. 2003

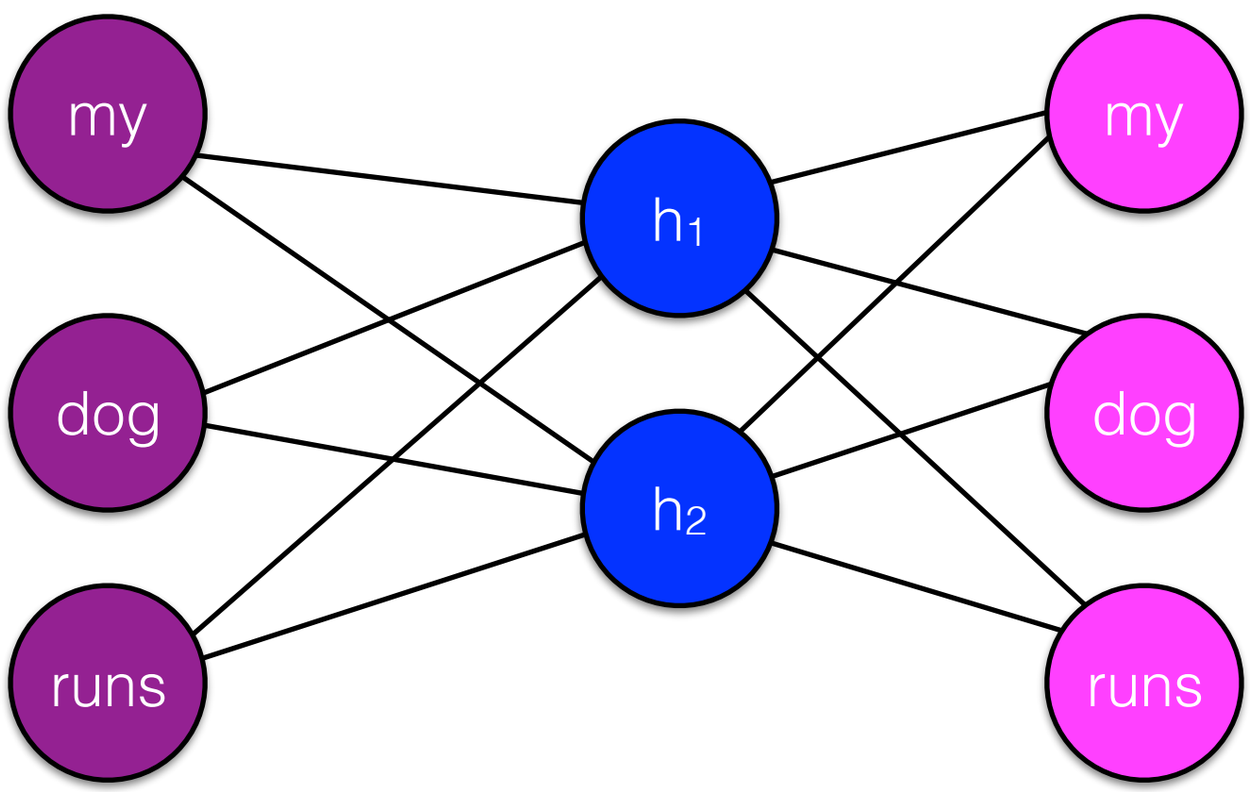
Let's try a bigram language model

$\mathscr{V}$ = {dog, my, runs}



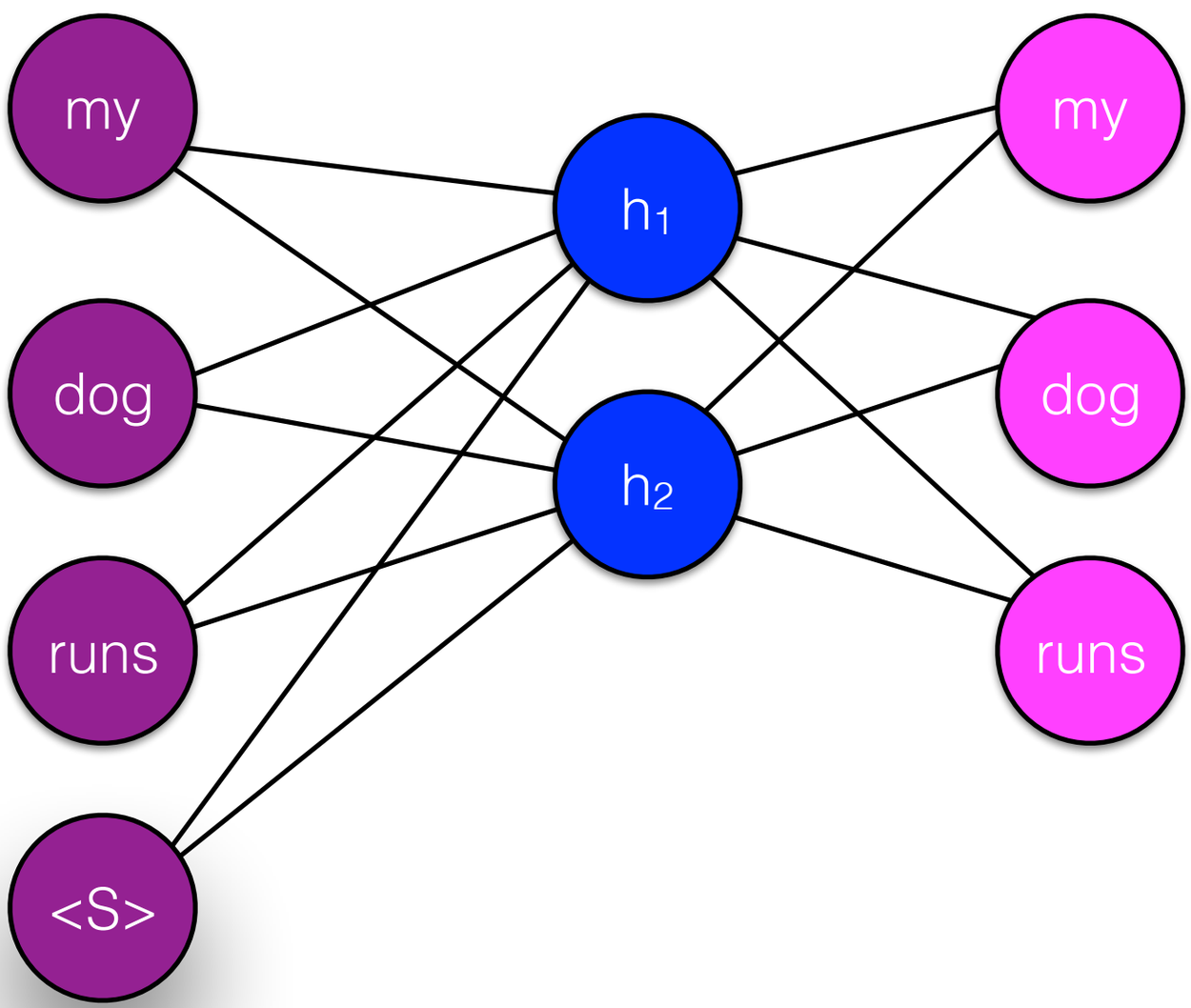
Let's try a bigram language model

$\mathscr{V}$ = {dog, my, runs}



Let's try a bigram language model

$\mathscr{V}$ = {dog, my, runs}



Can't forget the start token for predicting what word is first!

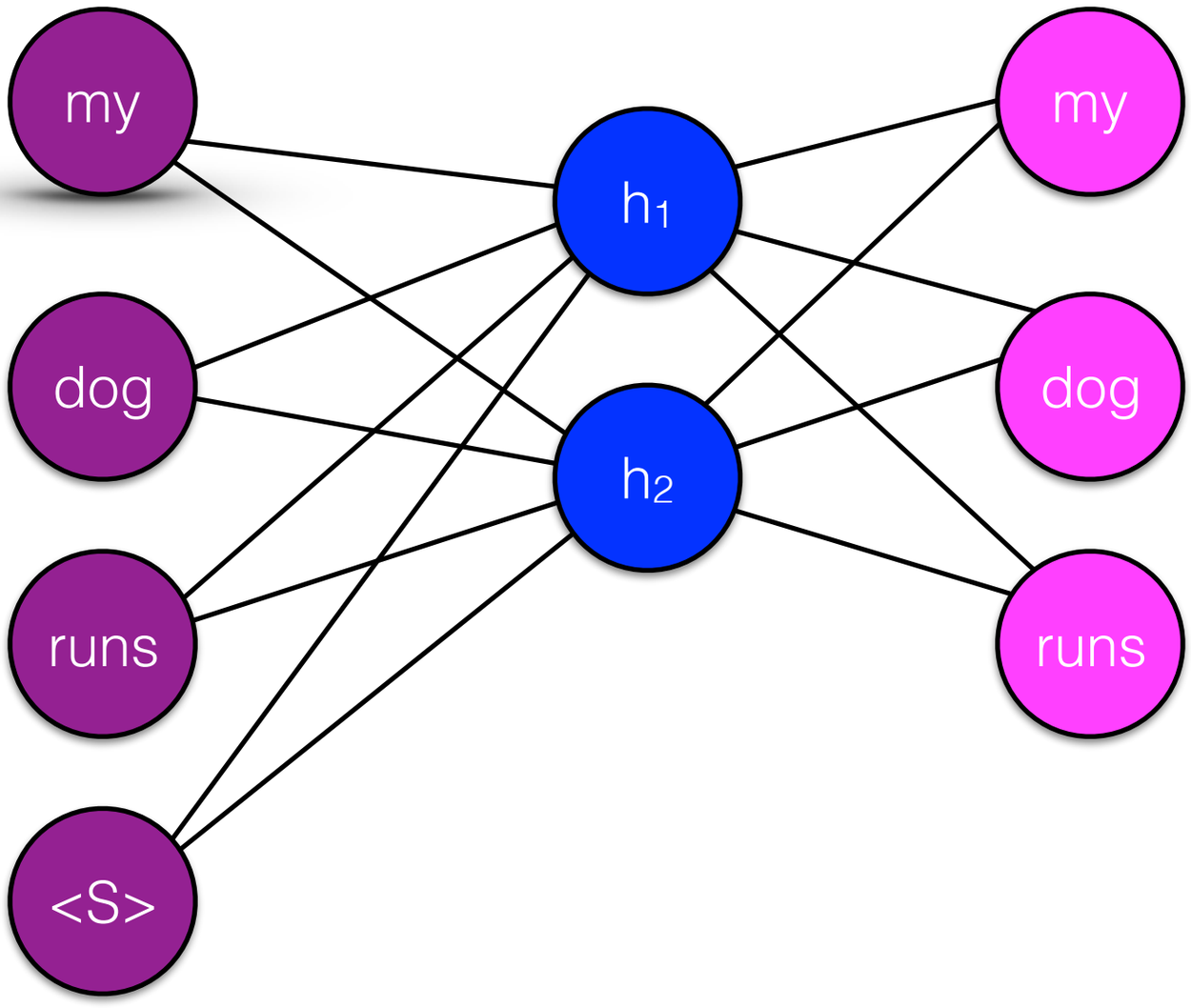
Let's try a bigram language model

$$\mathscr{V} = \{\text{dog, my, runs}\}$$

Input

Prediction

Let's say we have "my" as the current word



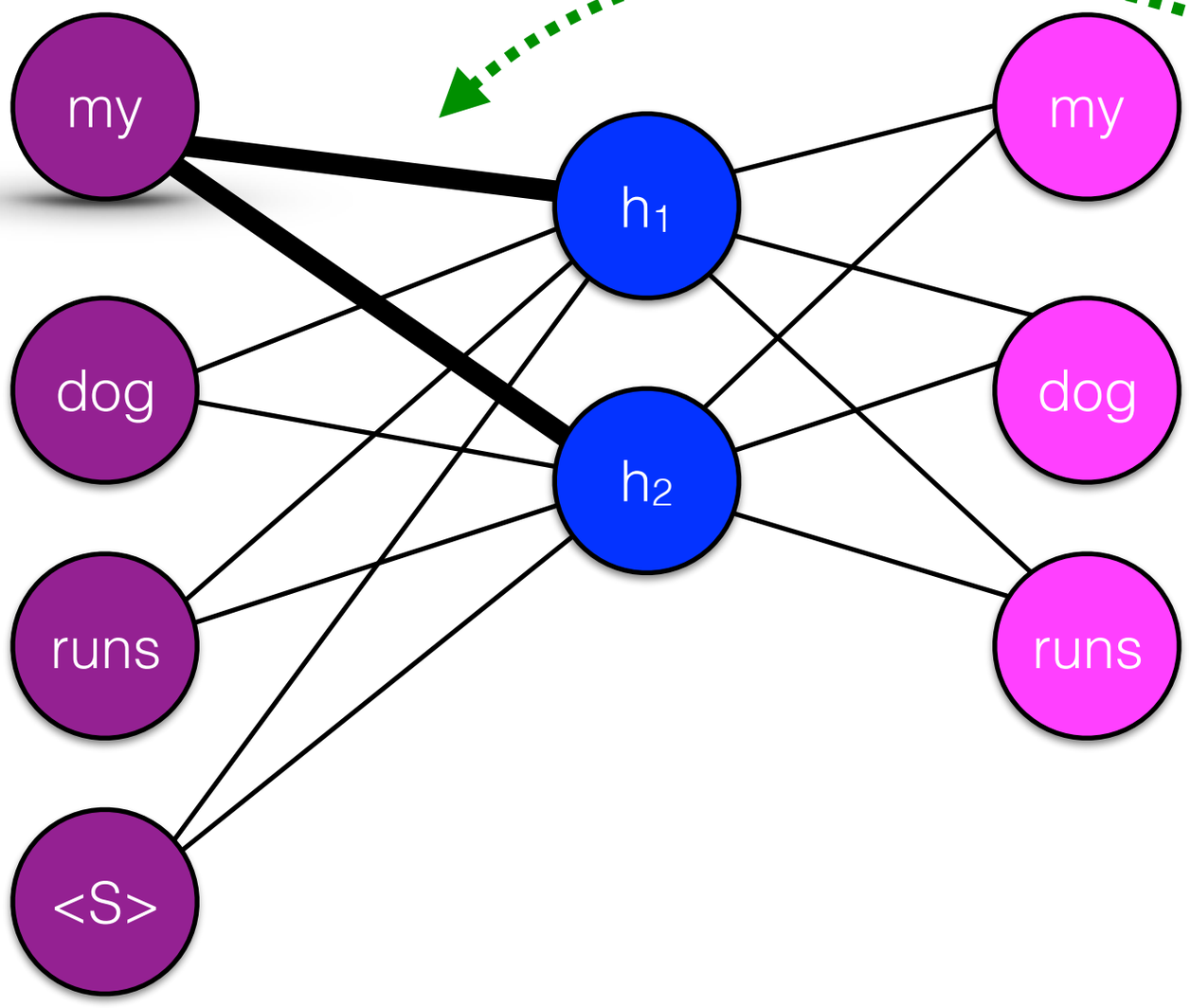
Let's try a bigram language model

$\mathscr{V}$ = {dog, my, runs}

Input

Prediction

Let's say we have "my" as the current word

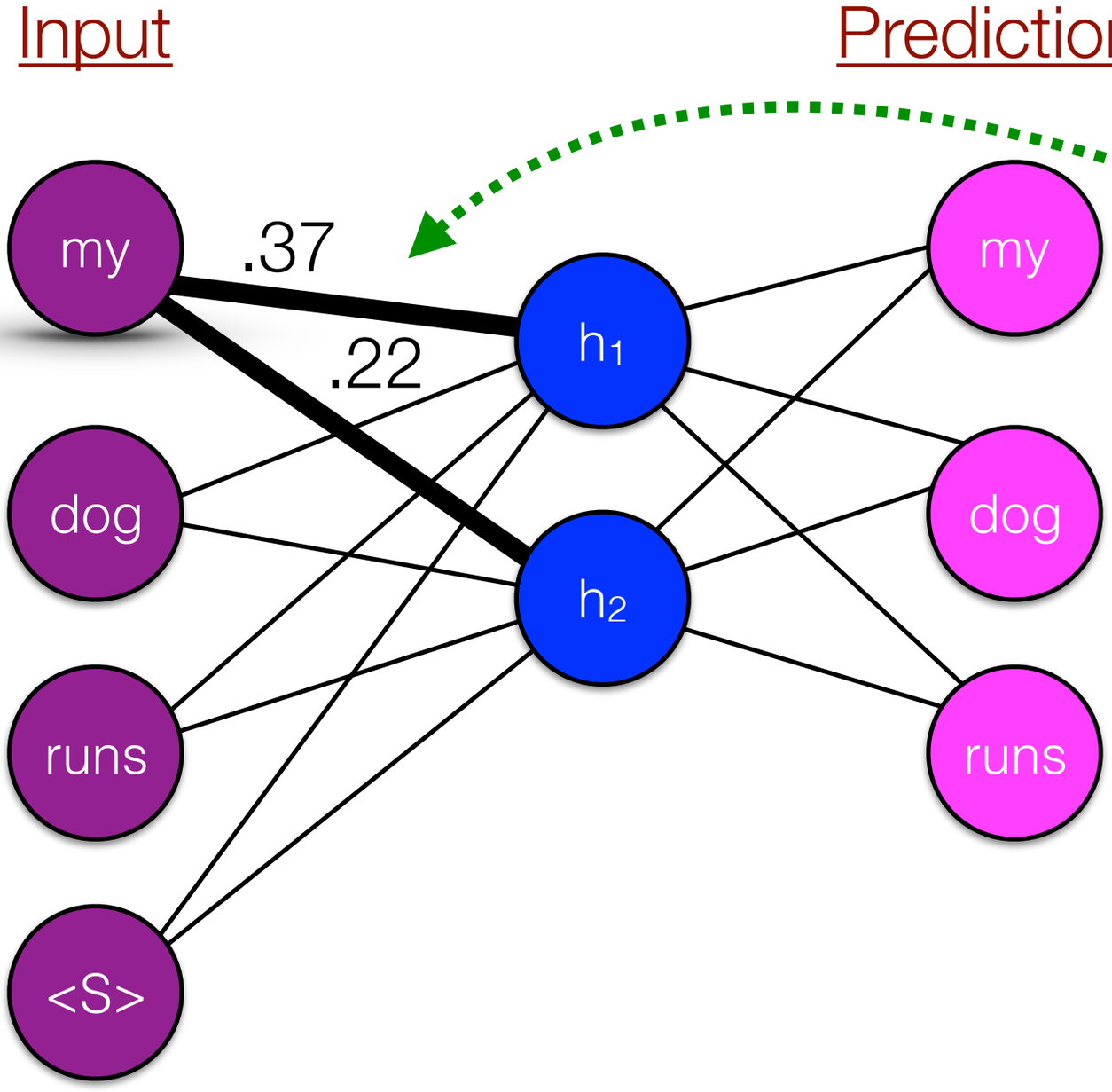


These weights determine how much to activate the hidden layer

Let's try a bigram language model

$\mathscr{V}$ = {dog, my, runs}

Let's say we have "my" as the current word



These weights determine how much to activate the hidden layer

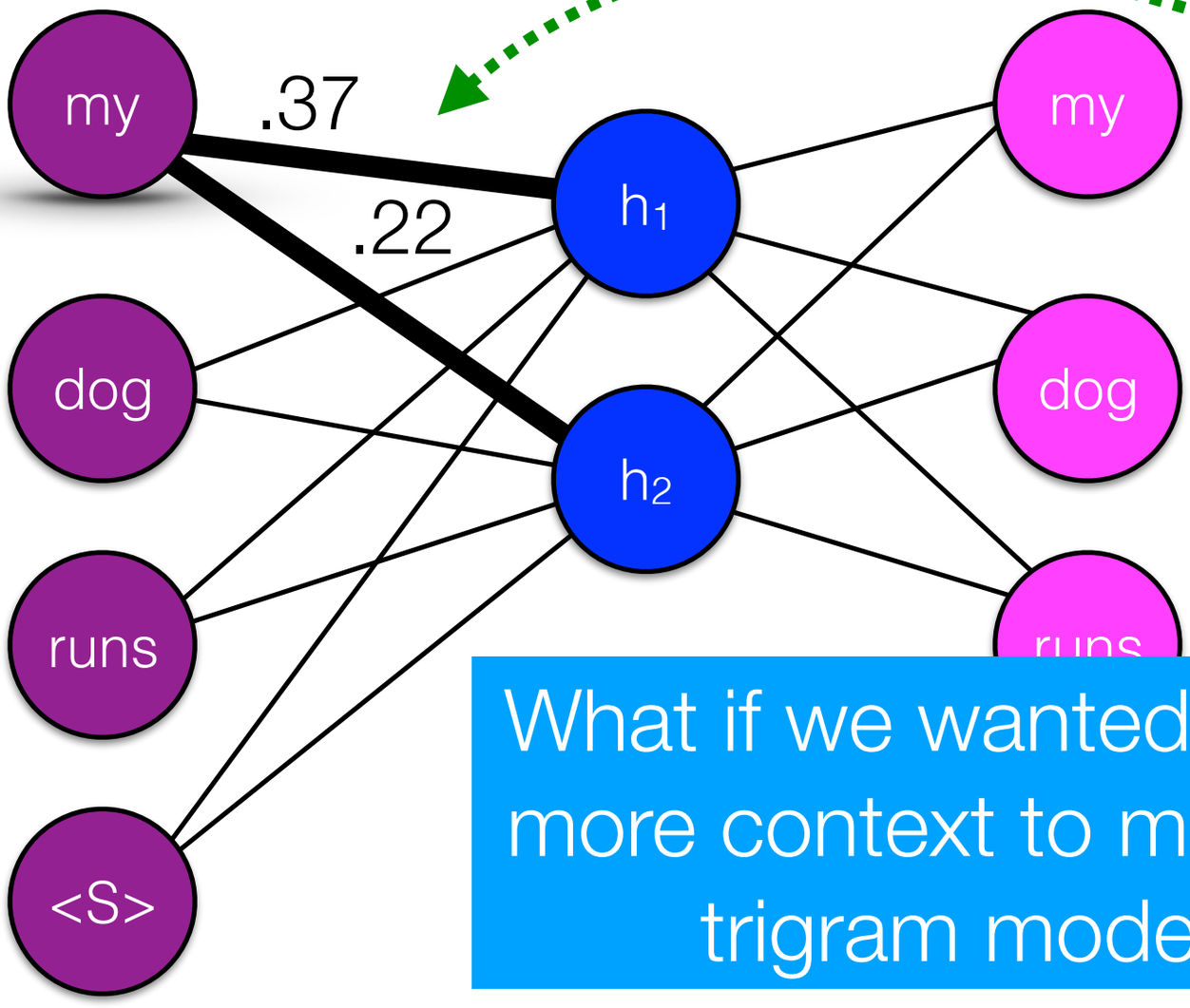
Let's try a bigram language model

$\mathscr{V}$ = {dog, my, runs}

Input

Prediction

Let's say we have "my" as the current word



These weights determine how much to activate the hidden layer

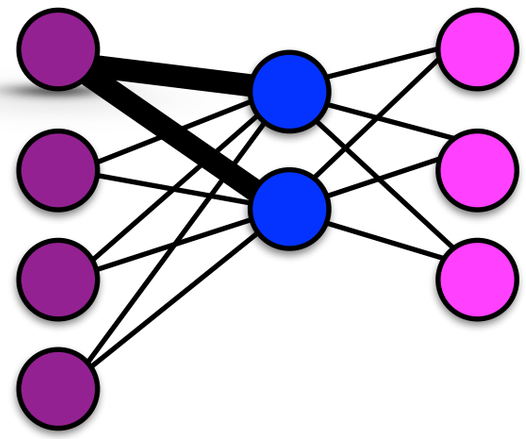
What if we wanted to add more context to make it a trigram model?

Let's try a trigram language model

$$\mathscr{V} = \{\text{dog, my, runs}\}$$

Input Prediction

Let's say we have "my dog" as the current words

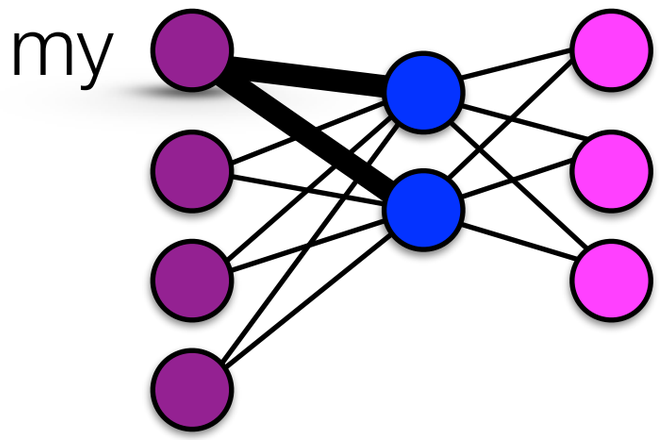


Let's try a trigram language model

$$\mathscr{V} = \{\text{dog, my, runs}\}$$

Input Prediction

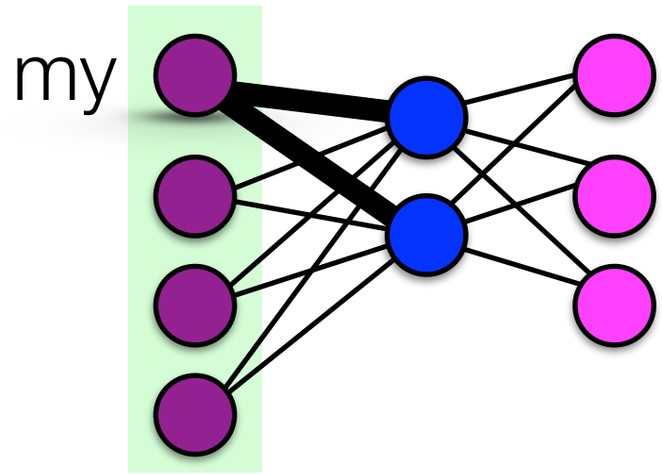
Let's say we have "my dog" as the current words



Let's try a trigram language model

$$\mathscr{V} = \{\text{dog, my, runs}\}$$

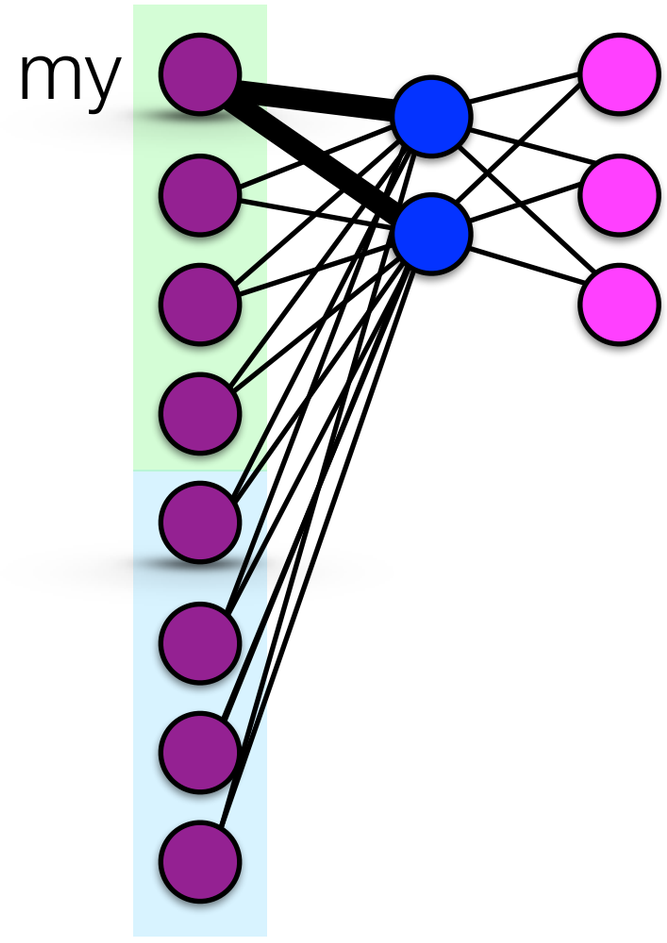
Let's say we have "my dog" as the current words



Let's try a trigram language model

$$\mathscr{V} = \{\text{dog, my, runs}\}$$

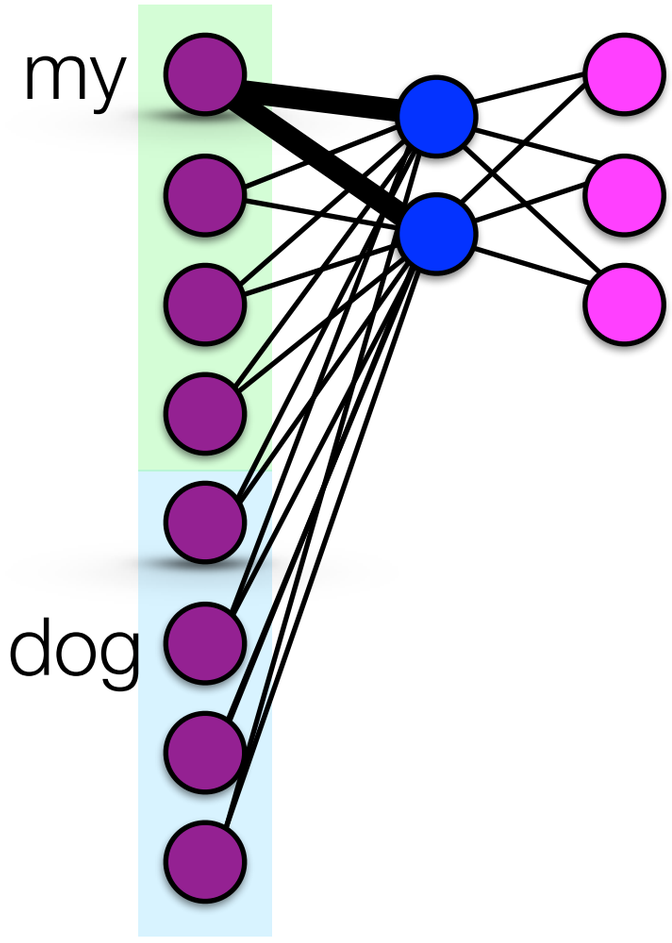
Let's say we have "my dog" as the current words



Let's try a trigram language model

$$\mathscr{V} = \{\text{dog, my, runs}\}$$

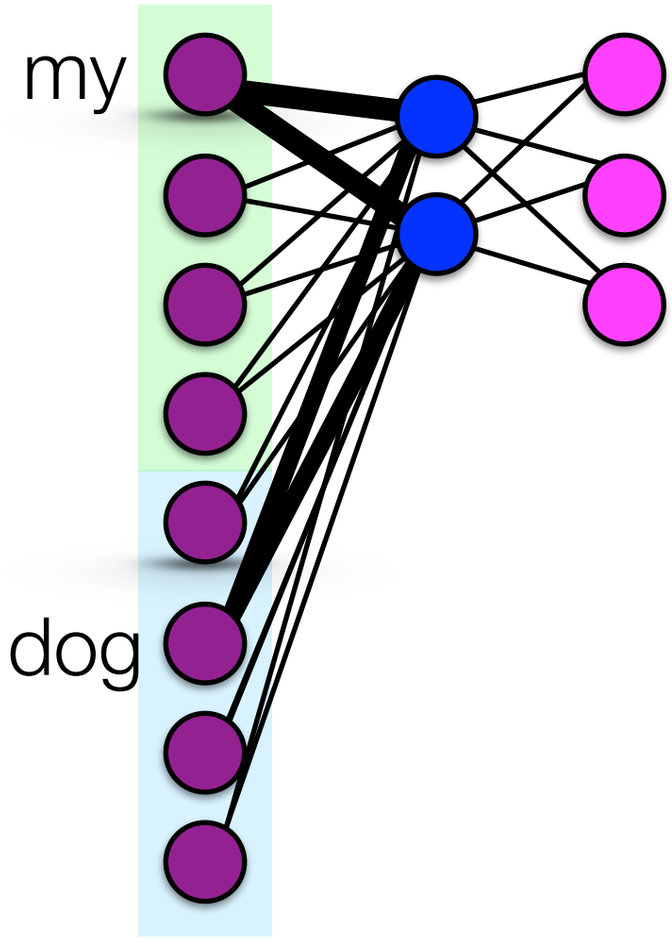
Let's say we have "my dog" as the current words



Let's try a trigram language model

$$\mathscr{V} = \{\text{dog, my, runs}\}$$

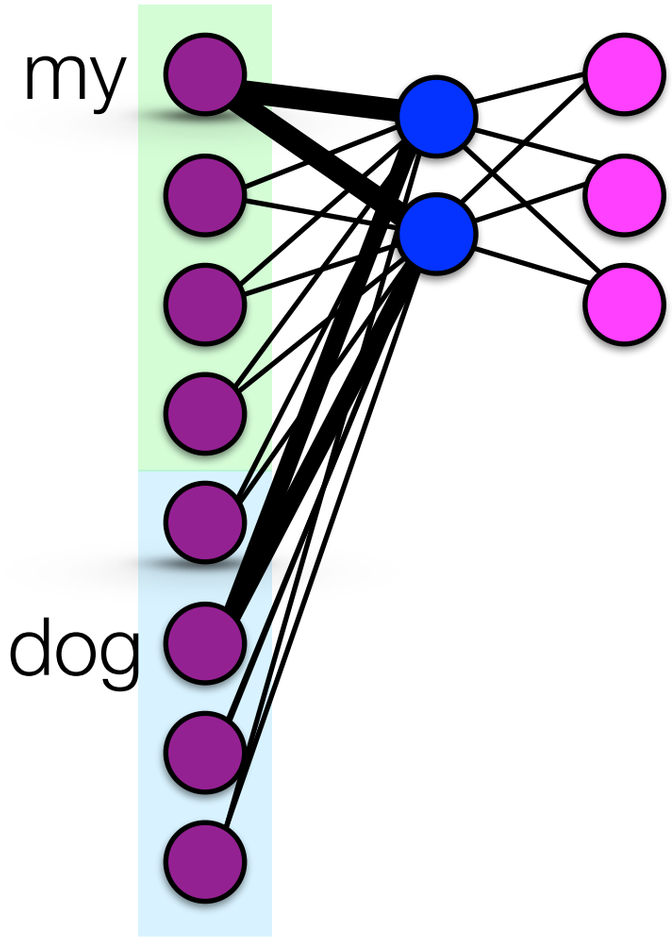
Let's say we have "my dog" as the current words



Let's try a trigram language model

$$\mathscr{V} = \{\text{dog, my, runs}\}$$

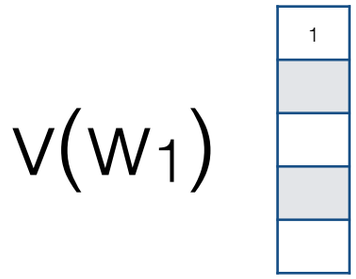
Let's say we have "my dog" as the current words

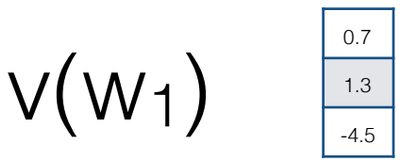


Q: Why do we need two start tokens?

$$x = [v(w_1); \ldots; v(w_k)]$$

w_1 = tried
w_2 = to
w_3 = prepare
w_4 = residents





one-hot encoding

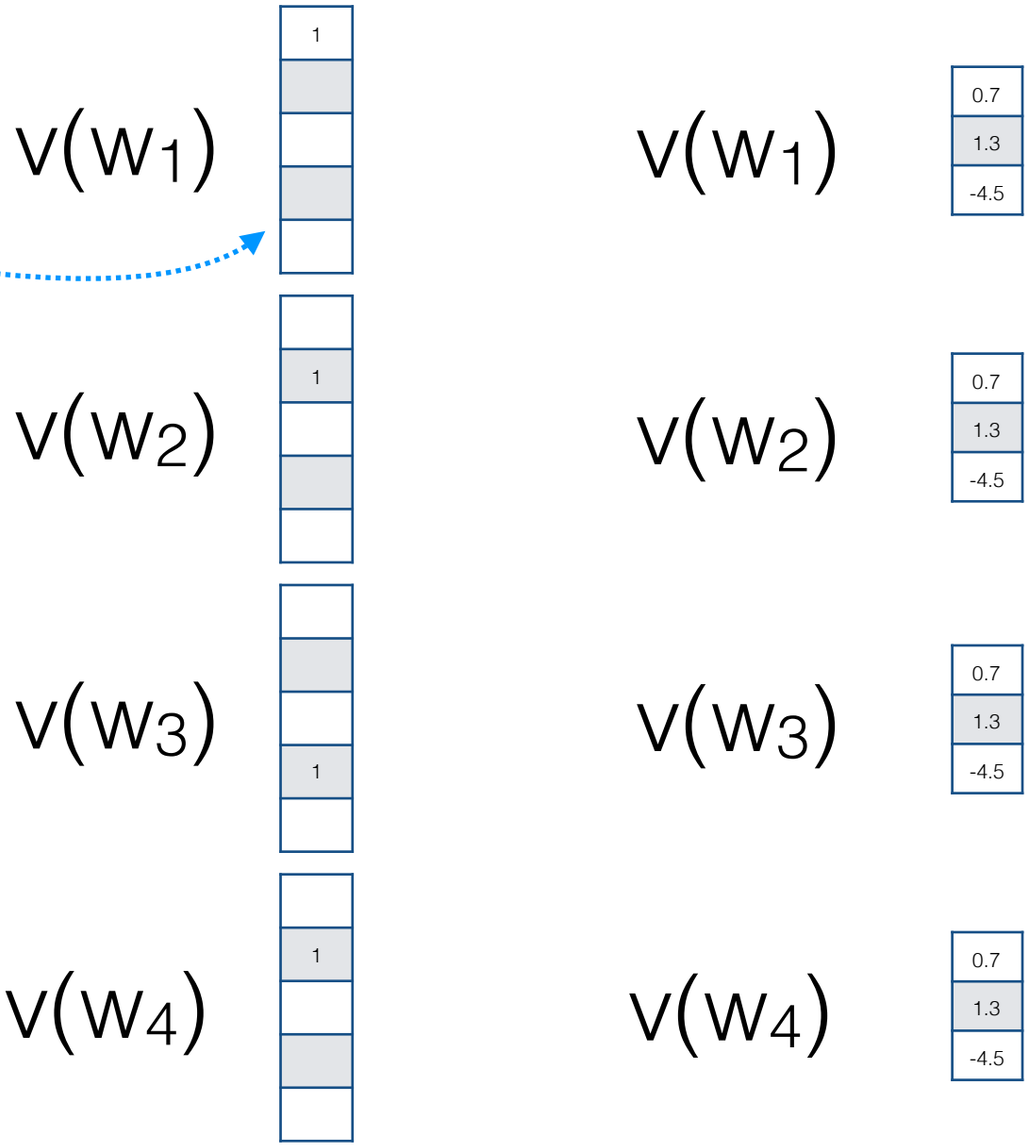
distributed representation

Bengio et al. 2003

This vector is the length $|\mathscr{V}|$ and everything is zero except for the dimension for w_1, which is 1

$$x = [v(w_1); \ldots; v(w_k)]$$

w_1 = tried
w_2 = to
w_3 = prepare
w_4 = residents



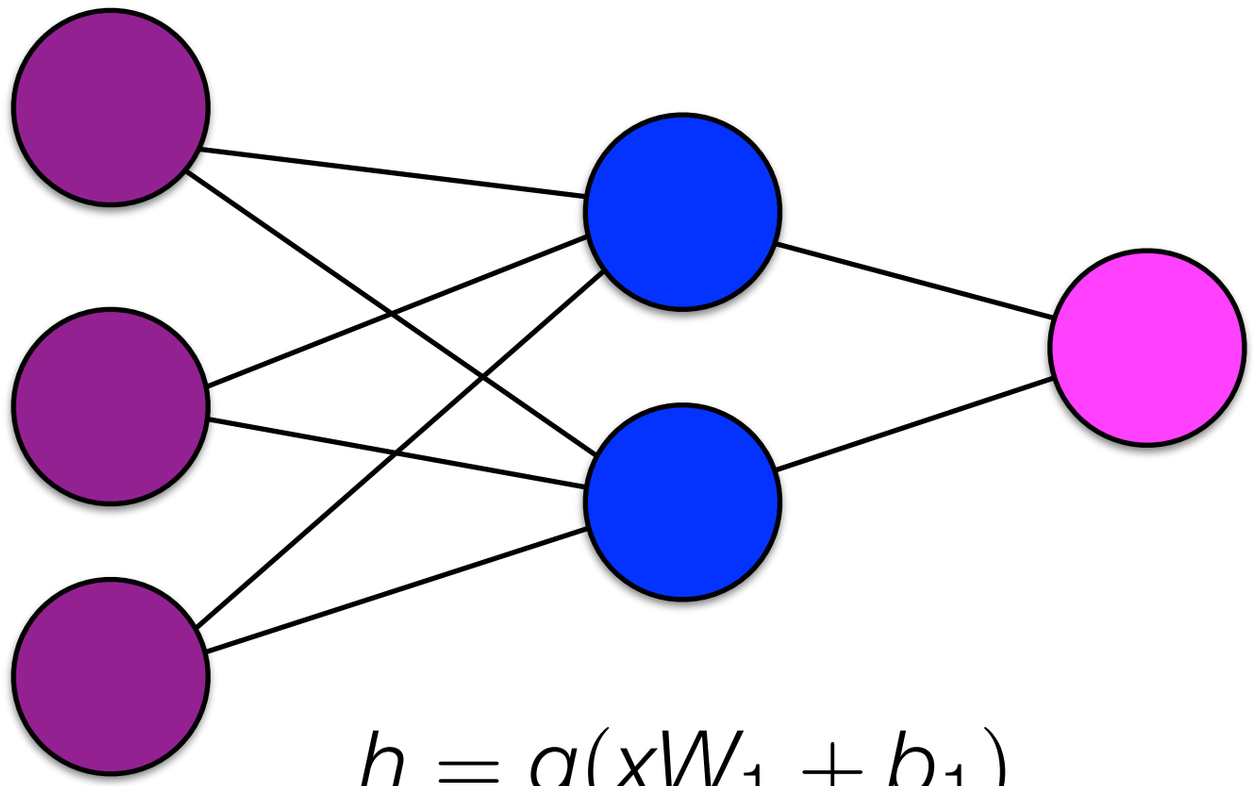
one-hot encoding

distributed representation

Bengio et al. 2003

$$W_1 \in \mathbb{R}^{kD \times H} \qquad W_2 \in \mathbb{R}^{H \times V}$$

$$b_1 \in \mathbb{R}^{H} \qquad b_2 \in \mathbb{R}^{V}$$

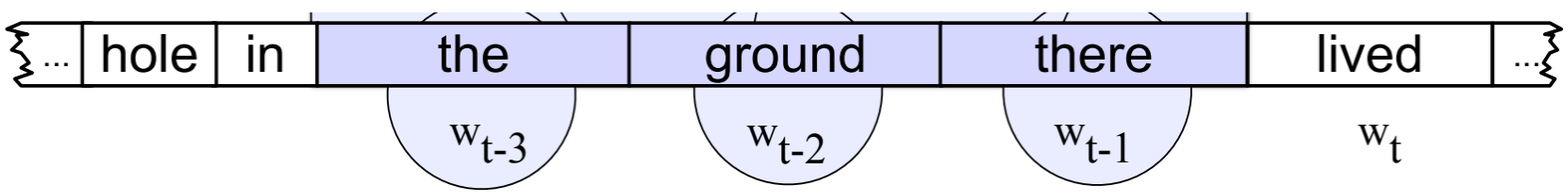


$$h = g(xW_1 + b_1)$$

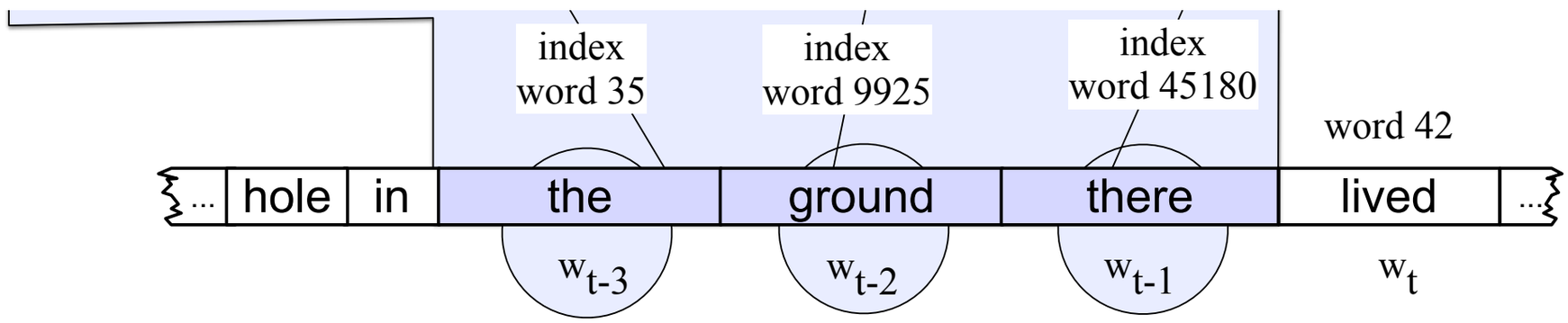
$$x = [v(w_1); \ldots; v(w_k)] \qquad \hat{y} = \text{softmax}(hW_2 + b_2)$$

Bengio et al. 2003

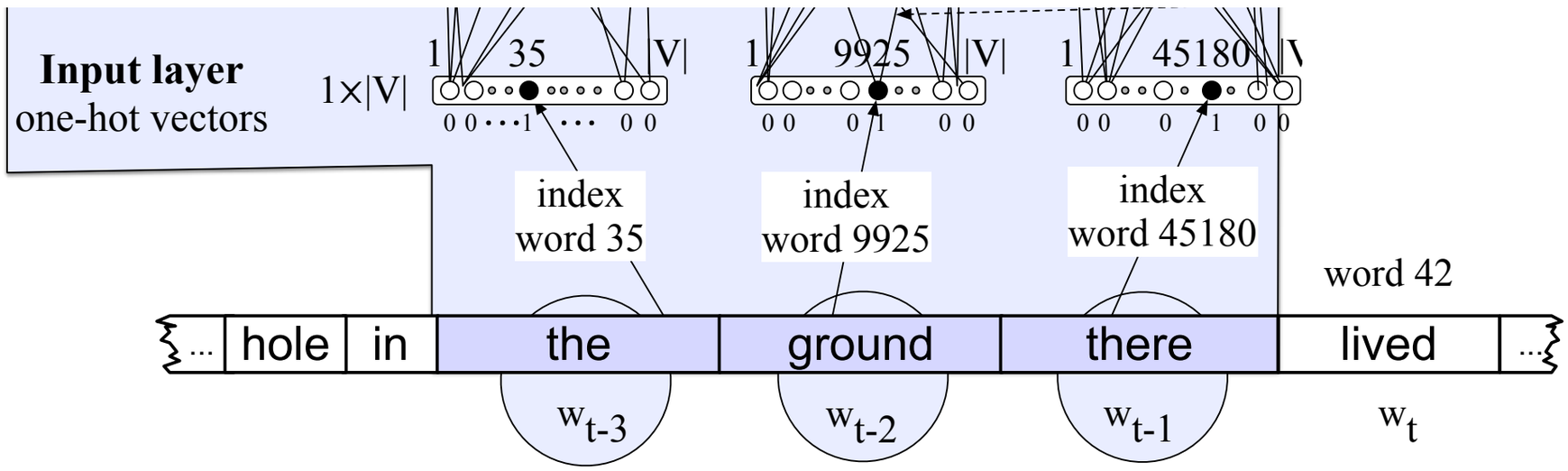
A slightly more complicated neural network language model



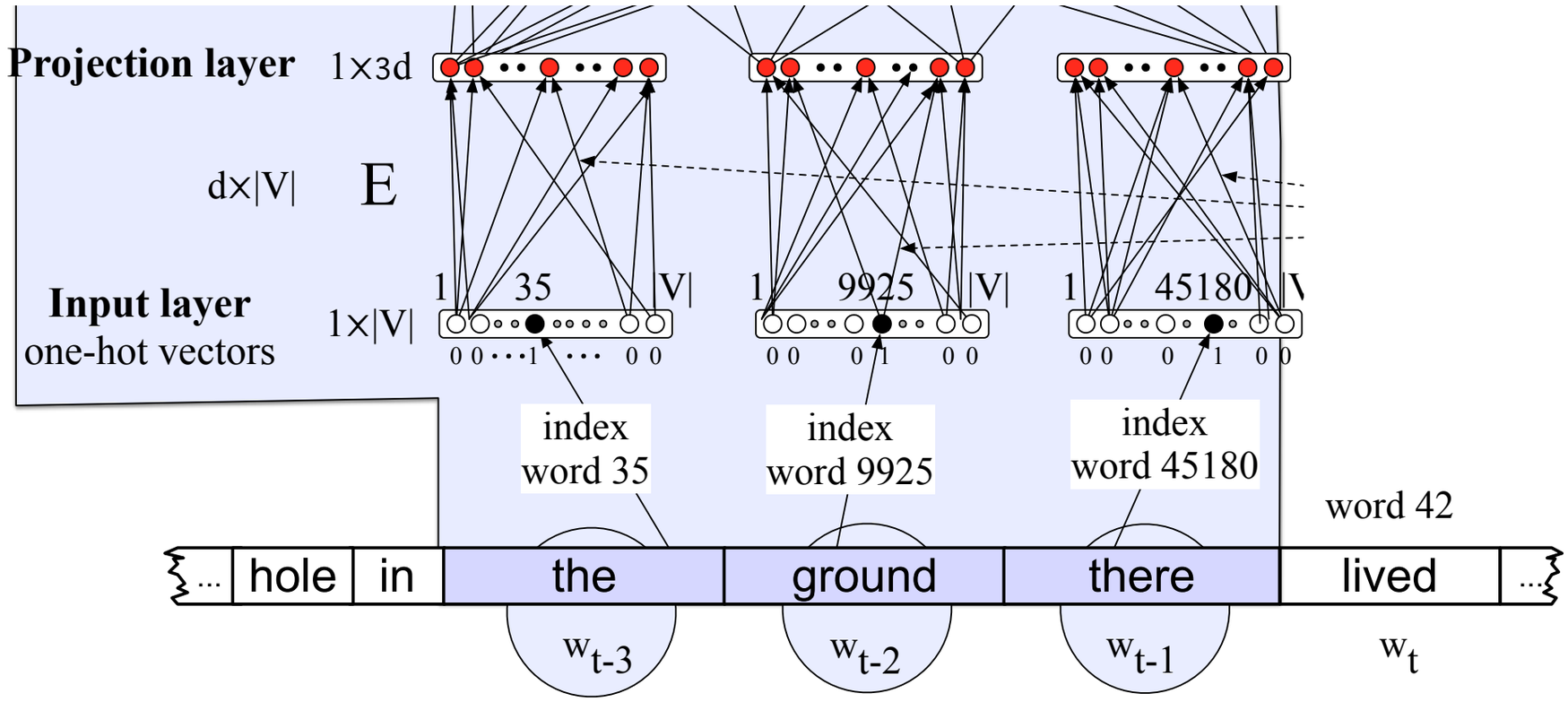
A slightly more complicated neural network language model



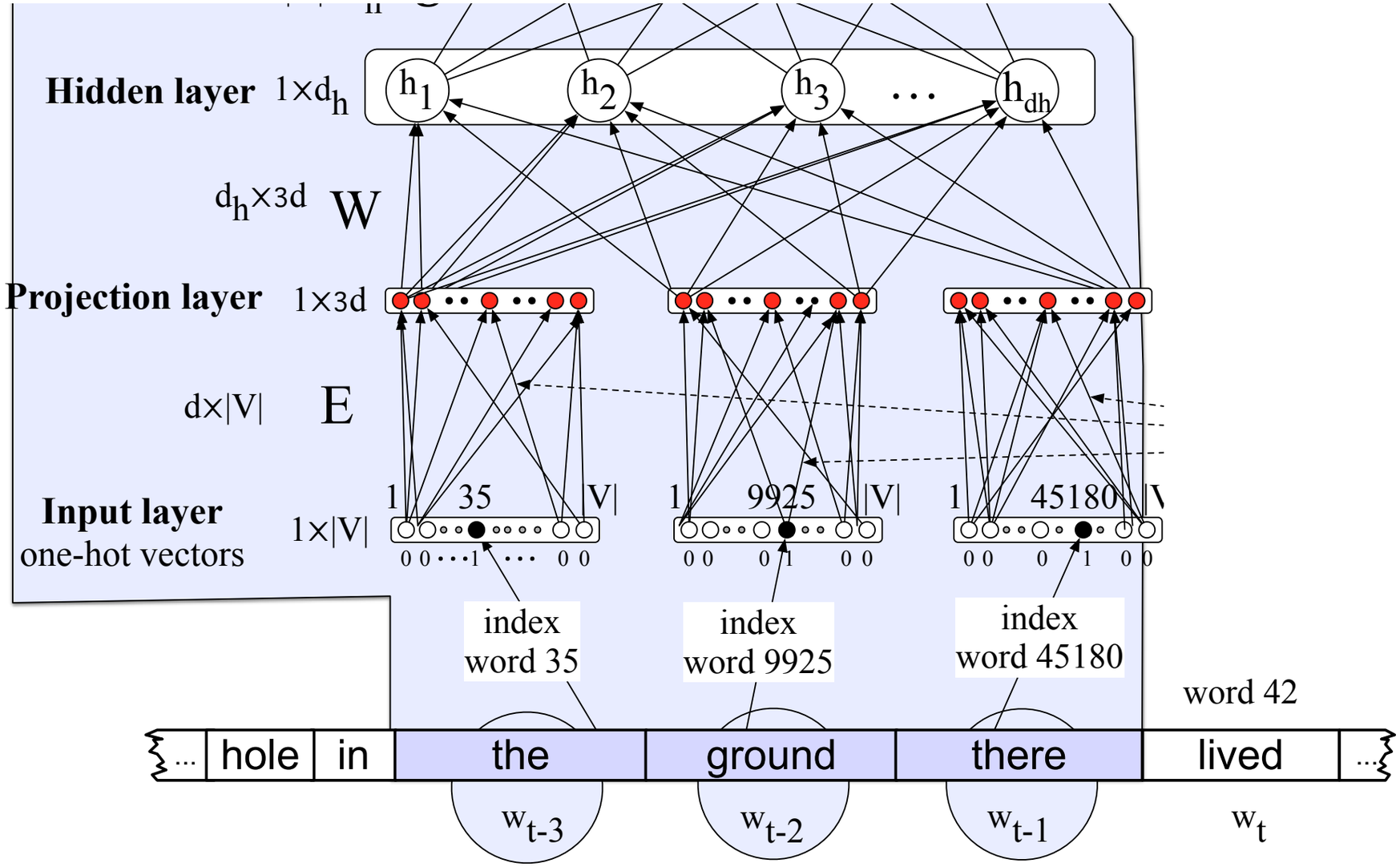
A slightly more complicated neural network language model



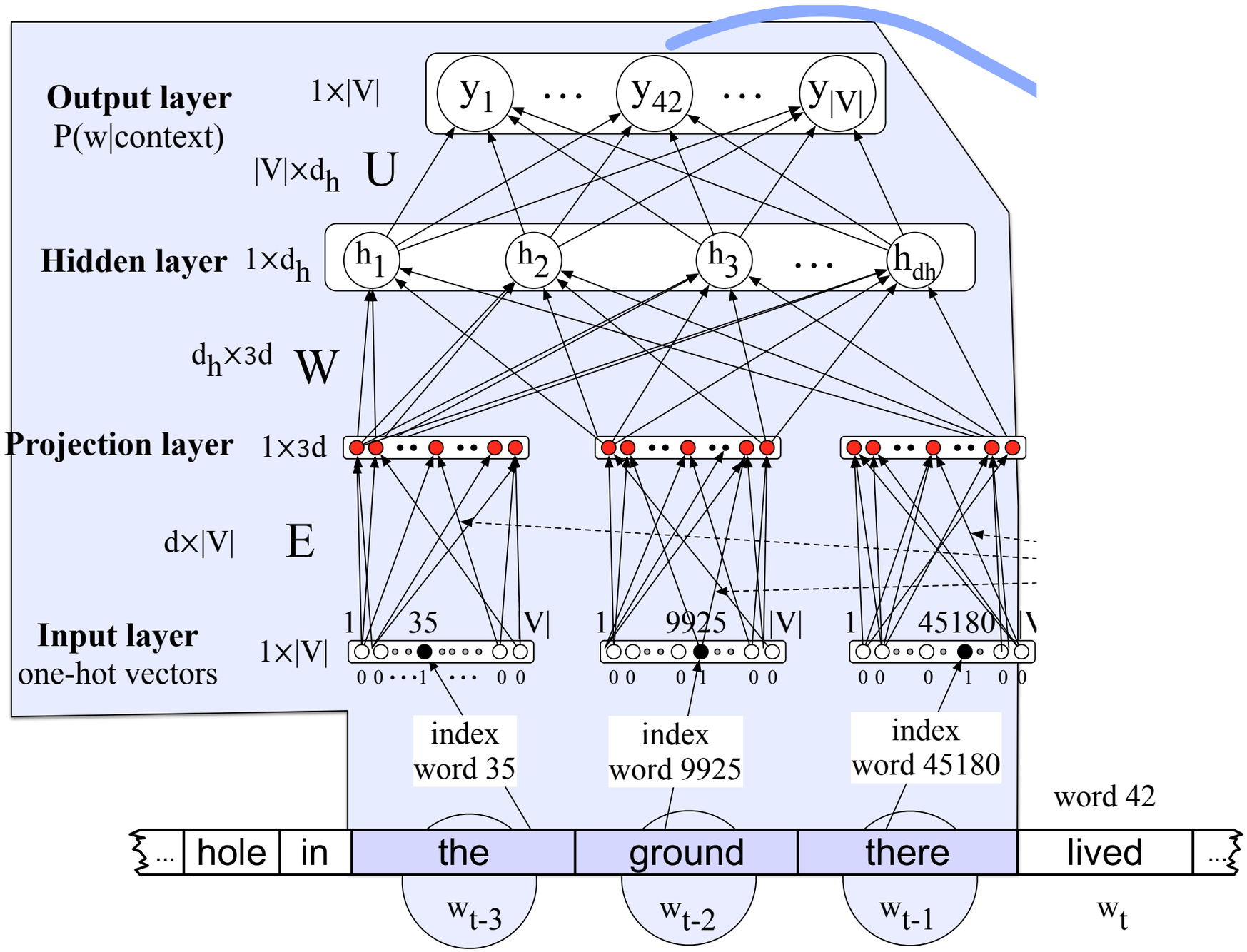
A slightly more complicated neural network language model



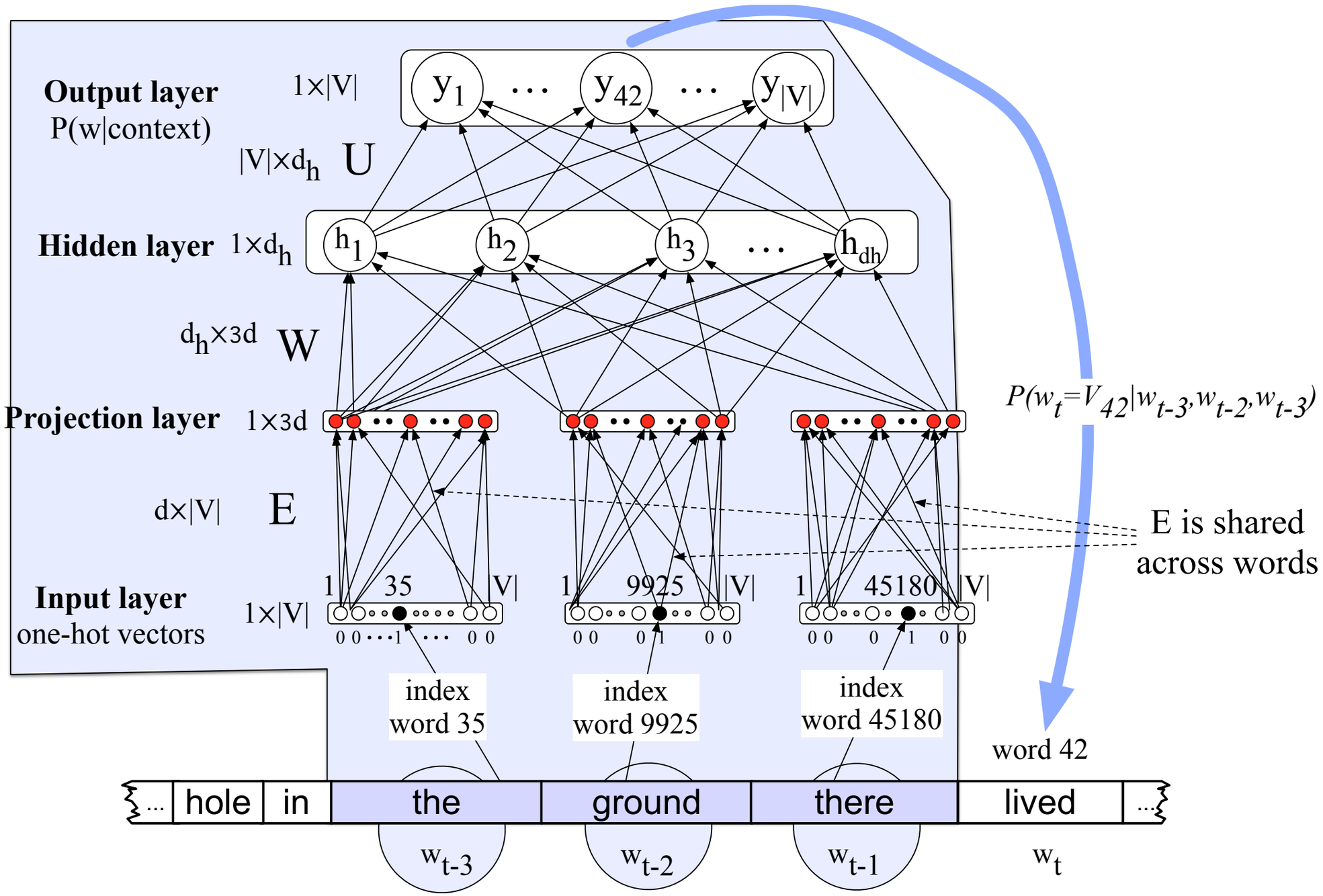
A slightly more complicated neural network language model



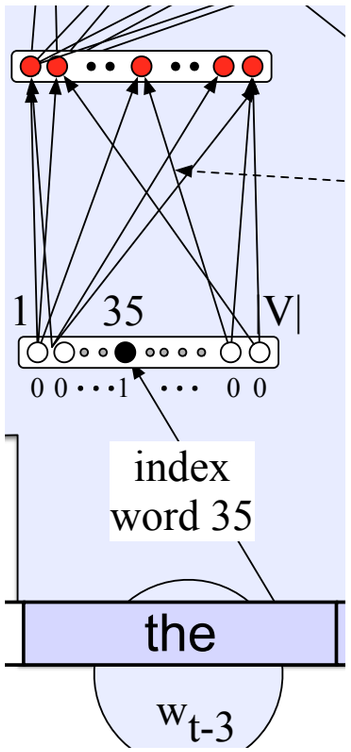
A slightly more complicated neural network language model

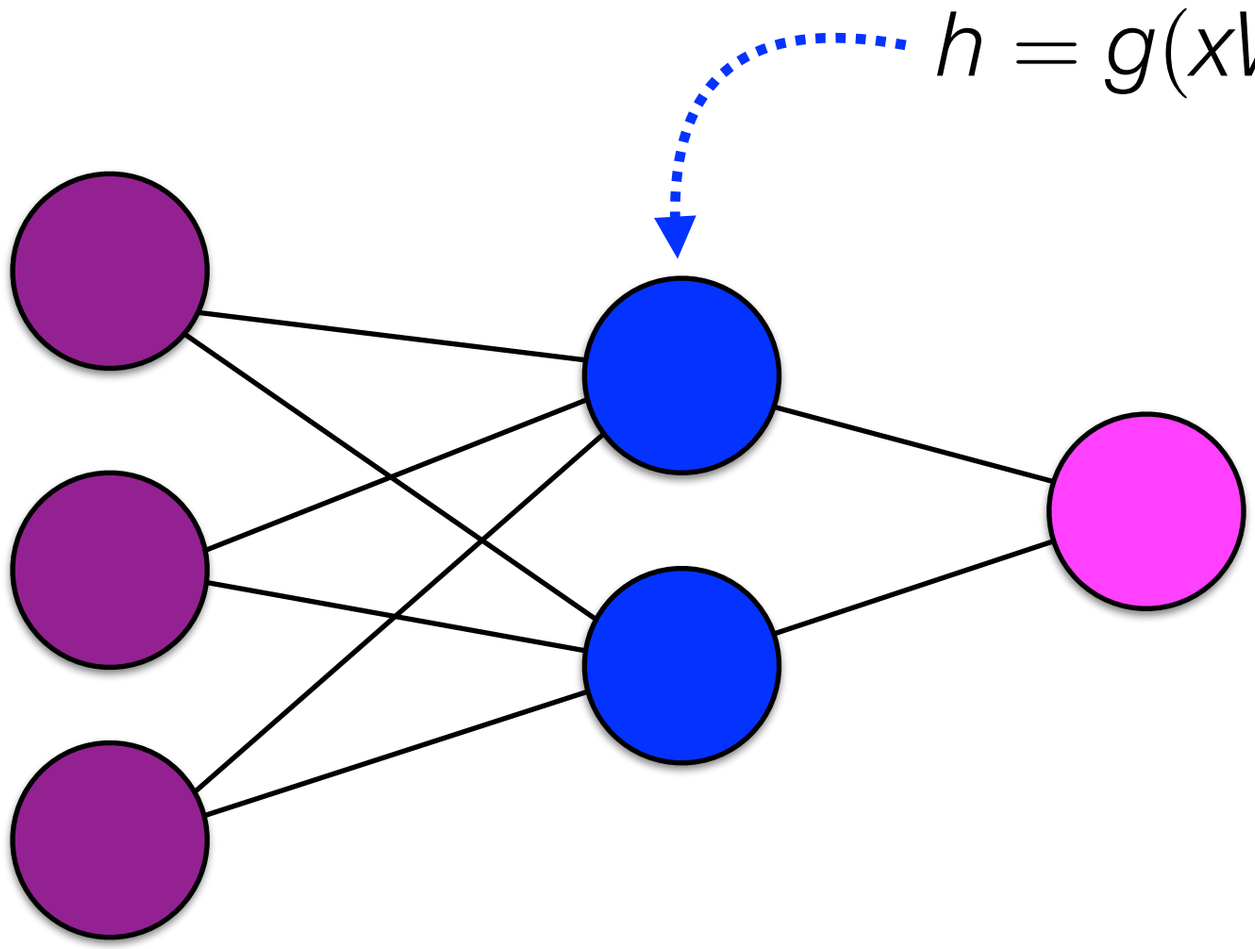


A slightly more complicated neural network language model

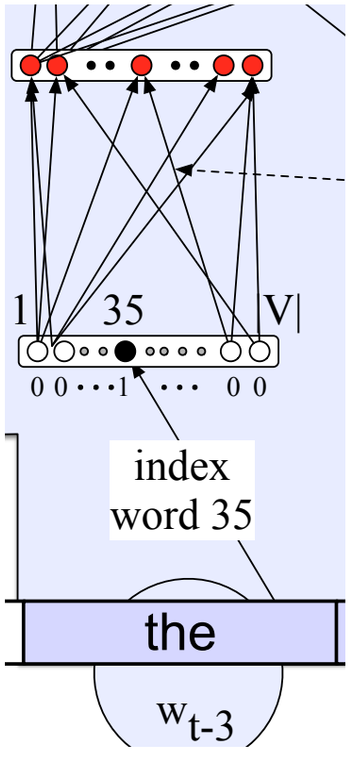


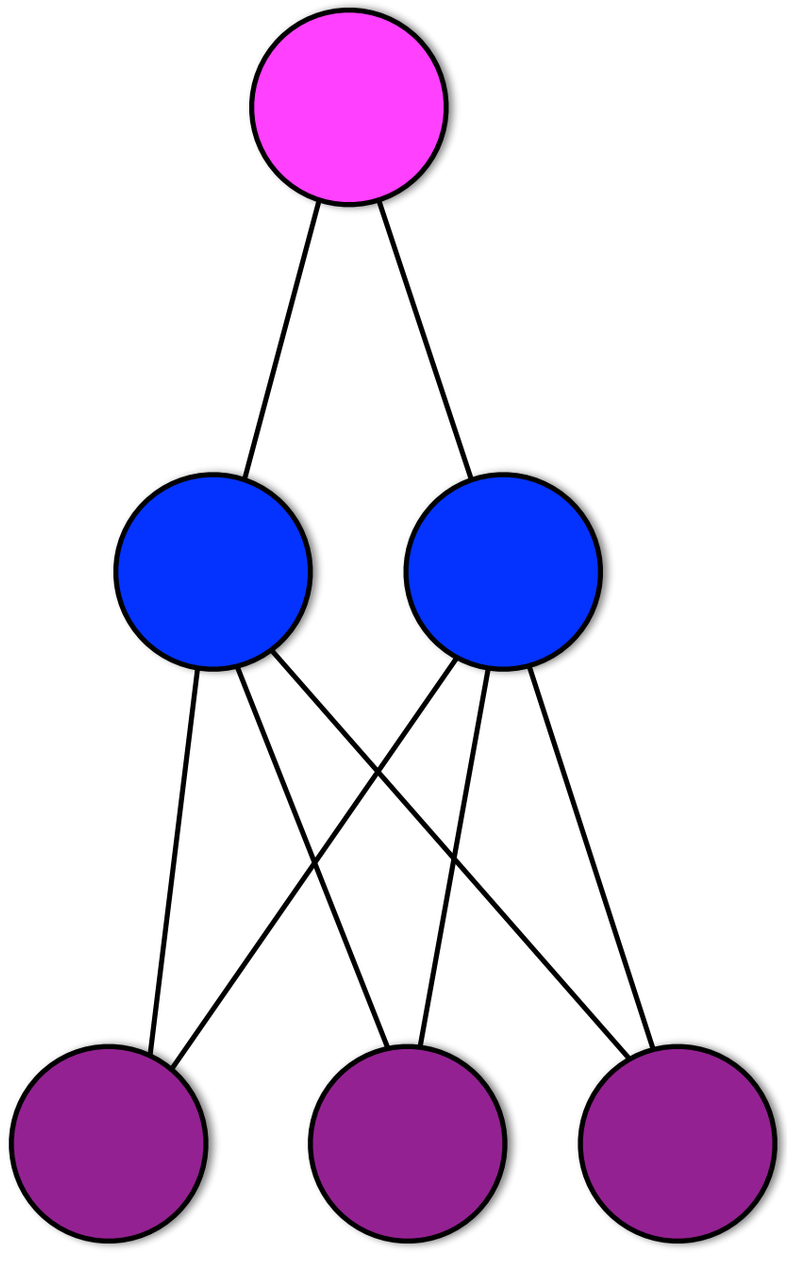
A closer look at that network



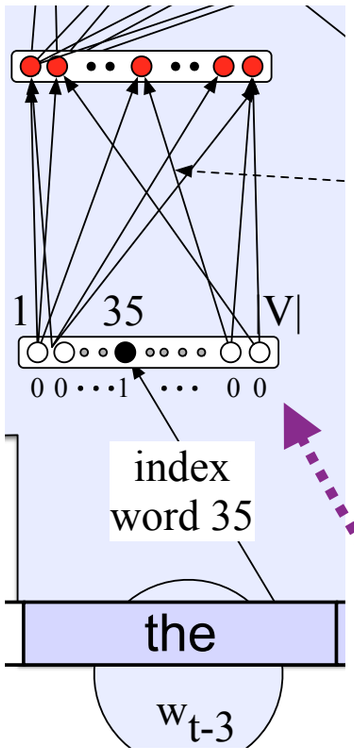


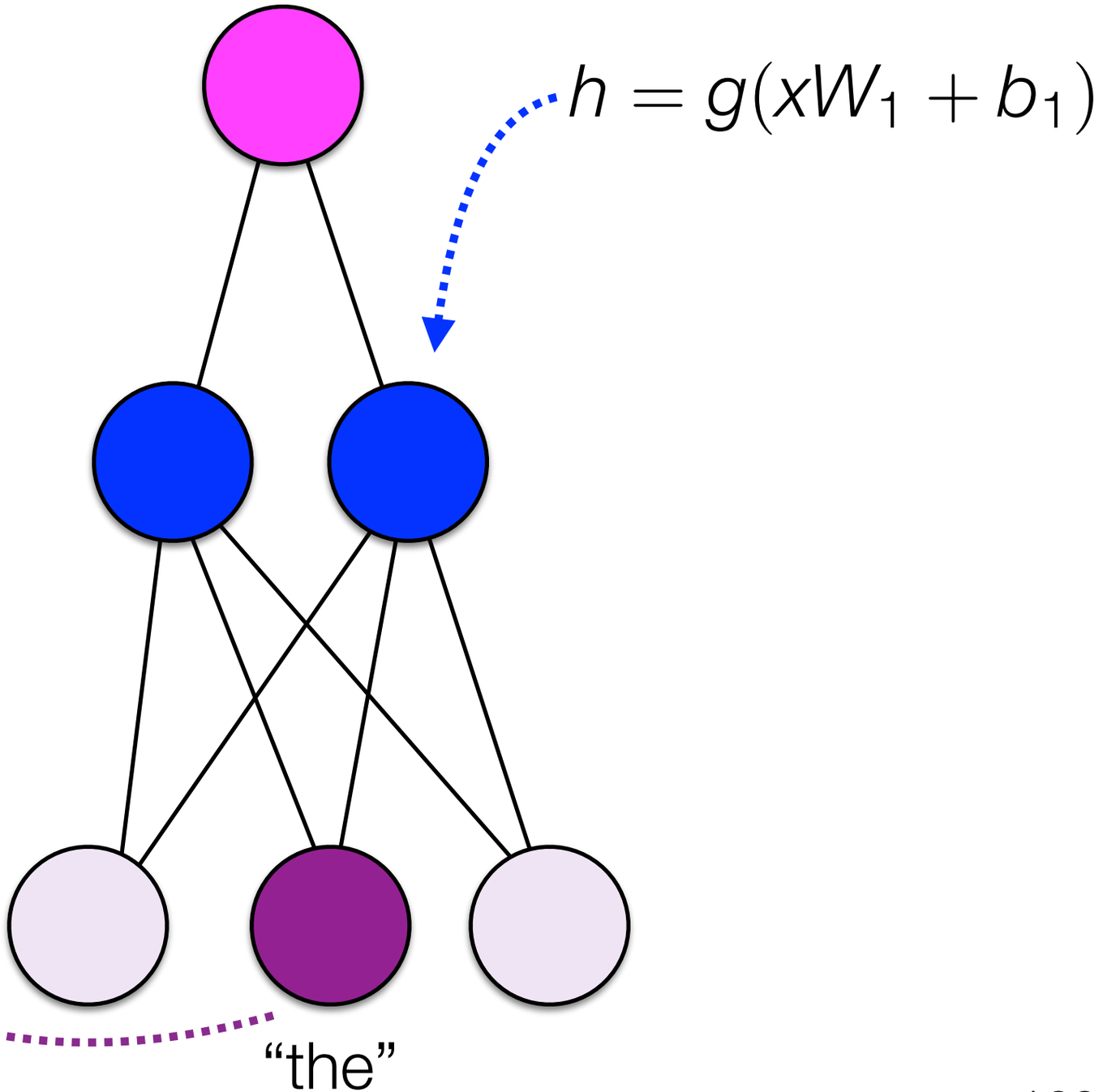
A closer look at that network





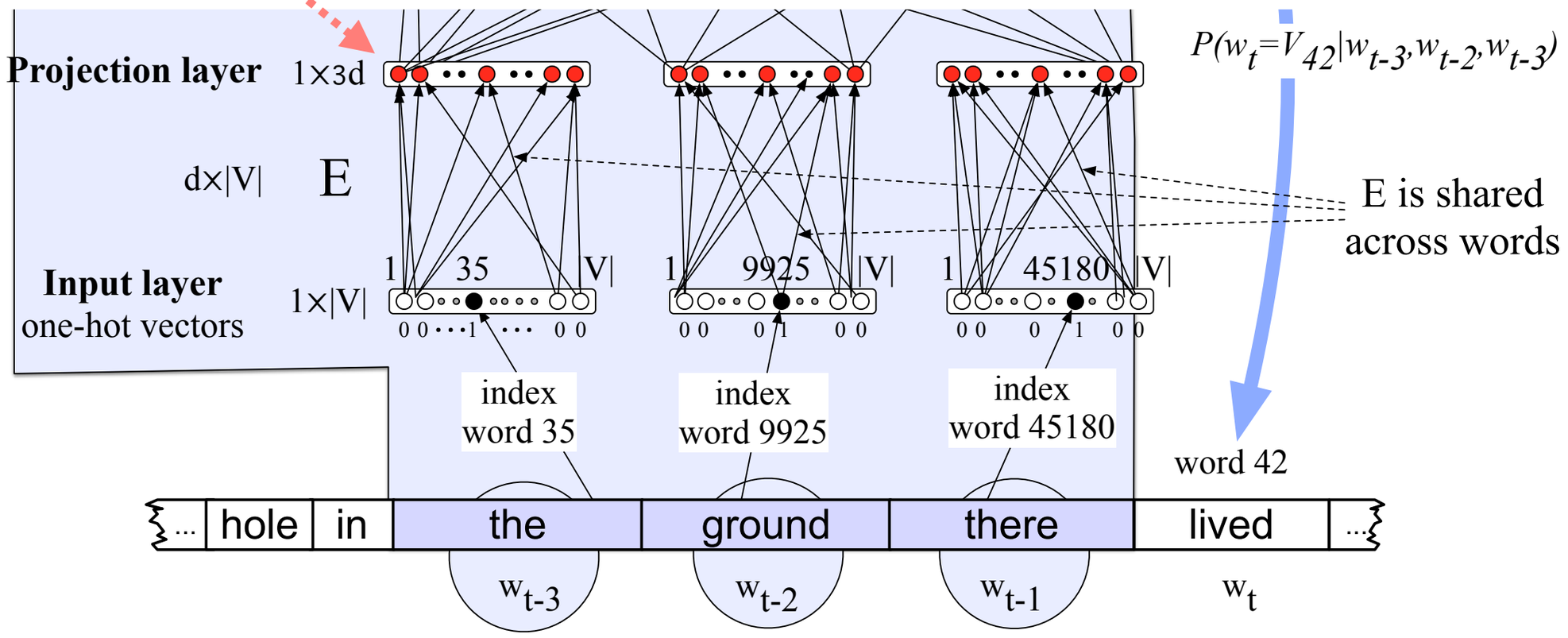
A closer look at that network





A closer look at that network

The projection layer is also known as an embedding layer. It's a distributed vector representation of the word



Softmax

$$P(Y = y \mid X = x; \beta) = \frac{\exp(x^\top \beta_y)}{\sum_{y' \in \mathcal{Y}} \exp(x^\top \beta_{y'})}$$

The softmax function is the general case of the sigmoid function for multiple classes. We already saw this equation on slide 87 for multi-class logistic regression!

$$\hat{y} = \frac{1}{1 + \exp\left(-\sum_{i=1}^{F} x_i \beta_i\right)}$$

Softmax

$$P(Y = y \mid X = x; \beta) = \frac{\exp(x^\top \beta_y)}{\sum_{y' \in \mathcal{Y}} \exp(x^\top \beta_{y'})}$$

The softmax function is the general case of the sigmoid function for multiple classes. We already saw this equation on slide 87 for multi-class logistic regression!

$$\hat{y} = \frac{1}{1 + \exp\left(-\sum_{i=1}^{F} x_i \beta_i\right)} = \sigma\left(\sum_{i=1}^{F} x_i \beta_i\right)$$

Neural LM

conditioning context

tried to prepare residents for the hardships of recovery from the

y

Neural LM

conditioning context

tried to prepare residents for the hardships of recovery from the

y

Neural LM

conditioning context

tried to prepare residents for the hardships of recovery from the

y

Neural LM

conditioning context

tried to prepare residents for the hardships of recovery from the

y

Neural LM

conditioning context

tried to prepare residents for the hardships of recovery from the

y

Image credit: Inception

Character LM

- Vocabulary $\mathscr{V}$ is a finite set of discrete characters
- When the output space is small, you're putting a lot of the burden on the structure of the model
- Encode long-range dependencies (suffixes depend on prefixes, word boundaries etc.)

Character LM

```
/*
 * Increment the size file of the new incorrect UI_FILTER group information
 * of the size generatively.
 */
static int indicate_policy(void)
{
  int error;
  if (fd == MARN_EPT) {
    /*
     * The kernel blank will coeld it to userspace.
     */
    if (ss->segment < mem_total)
      unblock_graph_and_set_blocked();
    else
      ret = 1;
    goto bail;
  }
  segaddr = in_SB(in.addr);
  selector = seg / 16;
  setup_works = true;
```

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Character LM

```
PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:
Well, your wit is in the care of side and that.
```

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Drawbacks

- Very expensive to train (especially for large vocabulary, though tricks exists — cf. hierarchical softmax)

- Backpropagation through long histories leads to vanishing gradients (cf. LSTMs in a few weeks).

- But they consistently have some of the strongest performances in perplexity evaluations.

Model	Perplexity			Entropy reduction over baseline	
	individual	+KN5	+KN5+cache	KN5	KN5+cache
3-gram, Good-Turing smoothing (GT3)	165.2	-	-	-	-
5-gram, Good-Turing smoothing (GT5)	162.3	-	-	-	-
3-gram, Kneser-Ney smoothing (KN3)	148.3	-	-	-	-
5-gram, Kneser-Ney smoothing (KN5)	**141.2**	-	-	-	-
5-gram, Kneser-Ney smoothing + cache	**125.7**	-	-	-	-
PAQ8o10t	131.1	-	-	-	-
Maximum entropy 5-gram model	142.1	138.7	124.5	0.4%	0.2%
Random clusterings LM	170.1	126.3	115.6	2.3%	1.7%
Random forest LM	131.9	131.3	117.5	1.5%	1.4%
Structured LM	146.1	125.5	114.4	2.4%	1.9%
Within and across sentence boundary LM	116.6	110.0	108.7	5.0%	3.0%
Log-bilinear LM	144.5	115.2	105.8	4.1%	3.6%
Feedforward neural network LM [50]	140.2	116.7	106.6	3.8%	3.4%
Feedforward neural network LM [40]	141.8	114.8	105.2	4.2%	3.7%
Syntactical neural network LM	131.3	110.0	101.5	5.0%	4.4%
Recurrent neural network LM	124.7	105.7	97.5	5.8%	5.3%
Dynamically evaluated RNNLM	123.2	102.7	98.0	6.4%	5.1%
Combination of static RNNLMs	102.1	95.5	**89.4**	7.9%	7.0%
Combination of dynamic RNNLMs	101.0	**92.9**	90.0	8.5%	6.9%

Mikolov 2010

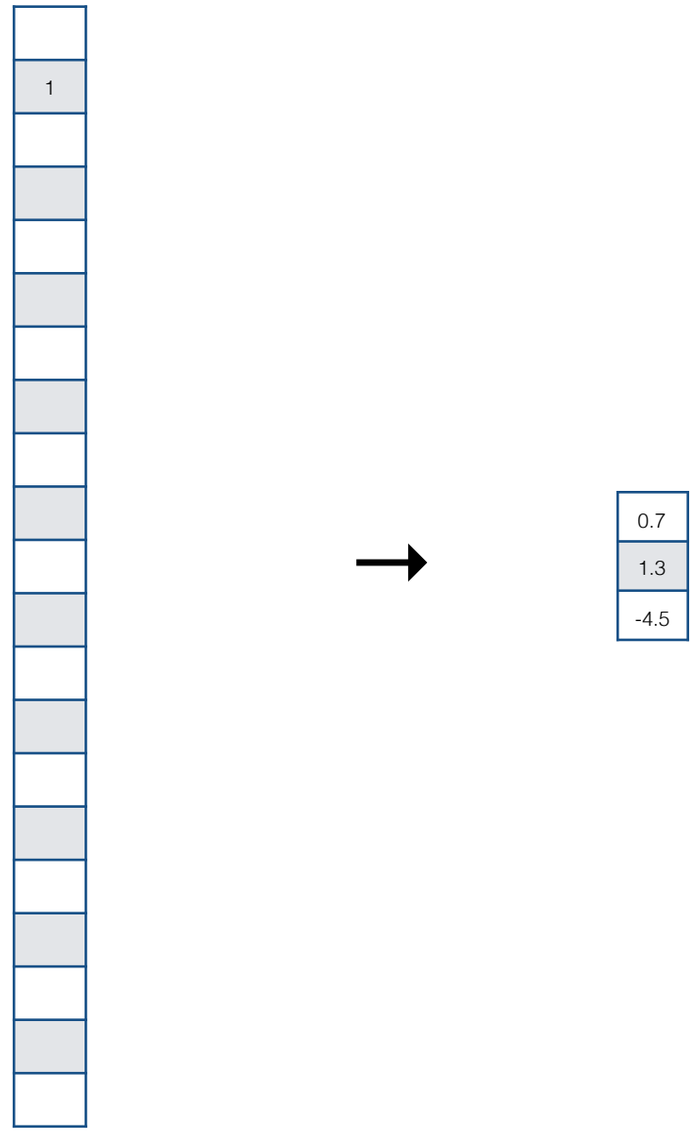
Model	Size	D	Valid	Test
Medium LSTM, Zaremba (2014)	10M	2	86.2	82.7
Large LSTM, Zaremba (2014)	24M	2	82.2	78.4
VD LSTM, Press (2016)	51M	2	75.8	73.2
VD LSTM, Inan (2016)	9M	2	77.1	73.9
VD LSTM, Inan (2016)	28M	2	72.5	69.0
VD RHN, Zilly (2016)	24M	10	67.9	65.4
NAS, Zoph (2016)	25M	-	-	64.0
NAS, Zoph (2016)	54M	-	-	62.4
LSTM	10M	1	61.8	59.6
LSTM		2	63.0	60.8
LSTM		4	62.4	60.1
RHN		5	66.0	63.5
LSTM	24M	1	61.4	59.5
LSTM		2	62.1	59.6
LSTM		4	60.9	58.3
RHN		5	64.8	62.2

Melis, Dyer and Blunsom 2017

Distributed representations

- Some of the greatest power in neural network approaches is in the representation of words (and contexts) as low-dimensional vectors

- We'll talk much more about that next week!



Let's build a count-based character LM

```
from collections import *

def normalize(word2count):
    s = float(sum(word2count()))
    return [(c,cnt/s) for c,cnt in word2count.items()]

def train_char_lm(fname, order=4):
    data = open(fname).read()
    lm = defaultdict(Counter)
    pad = "~" * order
    data = pad + data
    for i in range(len(data)-order):
        history, char = data[i:i+order], data[i+order]
        lm[history][char]+=1
    outlm = { hist:normalize(chars) \
                    for hist, w2count in lm.items() }
    return outlm
```

Courtesy of Yoav Goldberg

Let's build a count-based character LM

```
from collections import *

def normalize(word2count):
    s = float(sum(word2count()))
    return [(c,cnt/s) for c,cnt in word2count.items()]

def train_char_lm(fname, order=4):
    data = open(fname).read()
    lm = defaultdict(Counter)
    pad = "~" * order
    data = pad + data
    for i in range(len(data)-order):
        history, char = data[i:i+order], data[i+order]
        lm[history][char]+=1
    outlm = { hist:normalize(chars) \
                    for hist, w2count in lm.items() }
    return outlm
```

What does this data structure represent/contain?

Courtesy of Yoav Goldberg

Let's build a count-based character LM

```
from collections import *

def normalize(word2count):
    s = float(sum(word2count()))
    return [(c,cnt/s) for c,cnt in word2count.items()]

def train_char_lm(fname, order=4):
    data = open(fname).read()
    lm = defaultdict(Counter)
    pad = "~" * order
    data = pad + data
    for i in range(len(data)-order):
        history, char = data[i:i+order], data[i+order]
        lm[history][char]+=1
    outlm = { hist:normalize(chars) \
                for hist, w2count in lm.items() }
    return outlm
```

What does this data structure represent/contain?

What does this data structure represent/contain?

Courtesy of Yoav Goldberg

Demo time!

Grab the .ipynb file from canvas

- Try different length language models!
- Try different data:
    - Religious texts, Wikipedia articles, books by your favorite author,
- Try adding smoothing!
- Try switching it to a word-based language model

https://news.nationalgeographic.com/content/dam/news/2016/09/08/humpback-whales/01humpbackwhales.jpg

Summary

- Language modeling is a fundamental NLP task used in machine translation, spelling correction, speech recognition, etc.
- Traditional models use n-gram counts and smoothing
- Feed-forward take into account word similarity to generalize better
- Recurrent models can potentially learn to exploit long-range interactions and generally perform better than feed-forward models
- Neural models dramatically reduced perplexity

You should feel comfortable:

- Calculate the probability of a sentence given a trained model
- Estimating (e.g., trigram) language model
- Evaluating perplexity on held-out data
- Sampling a sentence from a trained model
- Explaining the parts of feed-forward and recurrent neural networks

Tools

- SRILM
  http://www.speech.sri.com/projects/srilm/

- KenLM
  https://kheafield.com/code/kenlm/

- Berkeley LM
  https://code.google.com/archive/p/berkeleylm/

Thinking forward to Homework 2

- You'll be implementing a neural language model

- And learning how to construct low-dimensional distributed representations of word meaning

Reminder: Homework 1

Due next week