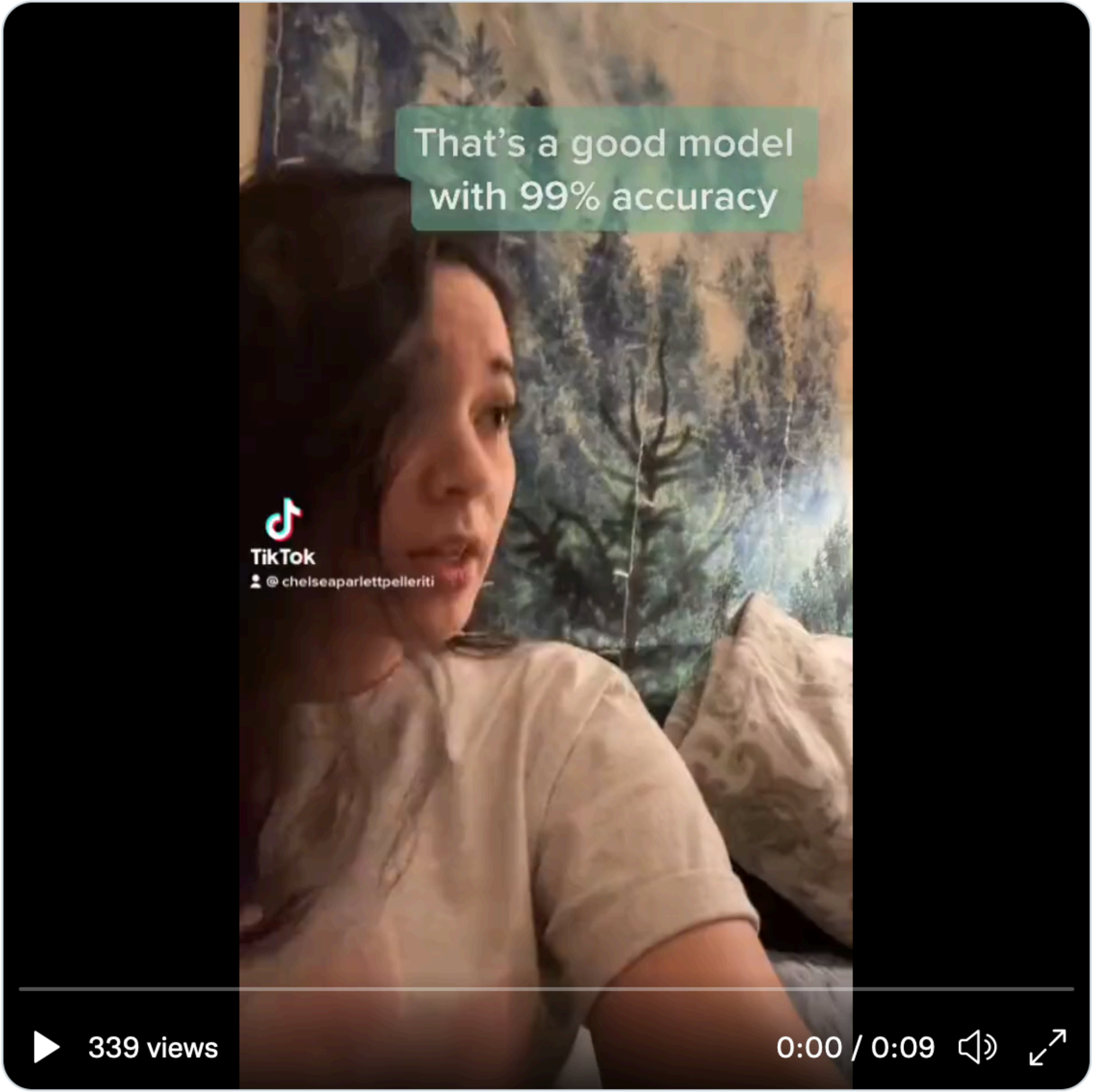
CS 232: Artificial Intelligence

Fall 2023

Prof. Carolyn Anderson
Wellesley College



https://www.tiktok.com/@chelseaparlettpelleriti/video/7072586373064248622?is_from_webapp=1&sender_device=pc&web_id=7159271848050869802

Recap

AI Tasks

Search

Uninformed Search

Informed Search

Adversarial Games

Navigation

Learning Under Uncertainty

September

Classification

Regression

Sentiment Analysis

Neural Networks

Image Classification

Text Classification

October

Generation

Language Models

Image Generation

Chatbots

Finetuning

Prompt Engineering

November

Chiang (2023):
ChatGPT is a Blurry JPEG of the Web

[... H]allucinations are anything but surprising; if a compression algorithm is designed to reconstruct text after ninety-nine per cent of the original has been discarded, we should expect that significant portions of what it generates will be entirely fabricated.

If a large language model has compiled a vast number of correlations between economic terms—so many that it can offer plausible responses to a wide variety of questions—should we say that it actually understands economic theory?

Imagine what it would look like if ChatGPT were a lossless algorithm. If that were the case, it would always answer questions by providing a verbatim quote from a relevant Web page. We would probably regard the software as only a slight improvement over a conventional search engine, and be less impressed by it.[...] When we're dealing with sequences of words, lossy compression looks smarter than lossless compression.

There's a type of blurriness that is acceptable, which is the re-stating of information in different words. Then there's the blurriness of outright fabrication, which we consider unacceptable when we're looking for facts.

Some might say that the output of large language models doesn't look all that different from a human writer's first draft, but, again, I think this is a superficial resemblance. Your first draft isn't an unoriginal idea expressed clearly; it's an original idea expressed poorly, and it is accompanied by your amorphous dissatisfaction, your awareness of the distance between what it says and what you want it to say.

Indeed, a useful criterion for gauging a large language model's quality might be the willingness of a company to use the text that it generates as training material for a new model. If the output of ChatGPT isn't good enough for GPT-4, we might take that as an indicator that it's not good enough for us, either.

The Curse of Recursion: Training on Generated Data Makes Models Forget

Ilia Shumailov* University of Oxford
Zakhar Shumaylov* University of Cambridge
Yiren Zhao Imperial College London
Yarin Gal University of Oxford
Nicolas Papernot University of Toronto & Vector Institute
Ross Anderson University of Cambridge & University of Edinburgh

Abstract

Stable Diffusion revolutionised image creation from descriptiv
demonstrated astonishing performance across a variety of lang
language models to the general public. It is now clear that larg
stay, and will bring about drastic change in the whole ecosys
paper we consider what the future might hold. What will happ
much of the language found online? We find that use of mode
irreversible defects in the resulting models, where tails of the
We refer to this effect as *model collapse*[1] and show that it ca
Gaussian Mixture Models and LLMs. We build theoretical i

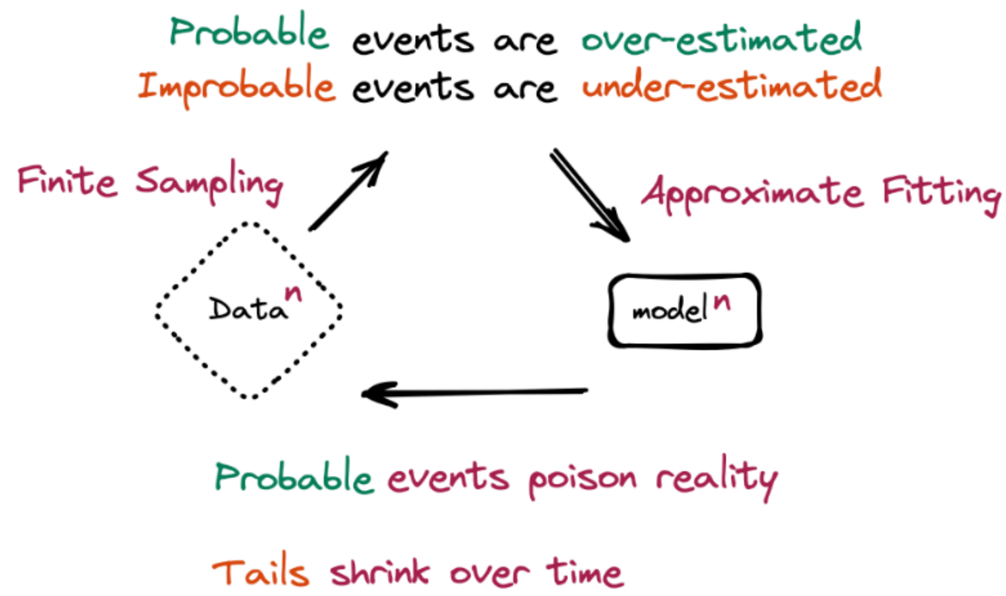


Figure 1: *Model Collapse* refers to a degenerative learning process where models start forgetting improbable events over time, as the model becomes poisoned with its own projection of reality.

Language Modeling

Language Model

A language model is a model that computes the probability of a sentence in a language:

$p(w_0, \ldots , w_n)$

A language model can also be used to compute the probability of the next word in a sentence:

$p(w_n \mid w_0, \ldots , w_{n-1})$

Sentence probability

How do we guess what the next word is?

I write this sitting in the kitchen…

Answer: try out different words and compare their likelihood.

p(I write this sitting in the kitchen sink)

versus

p(I write this sitting in the kitchen knife)

versus

p(I write this sitting in the kitchen chair)

The Chain Rule

Chain Rule of Probability:

$$P(X_1 \ldots X_n) = P(X_1)P(X_2 \mid X_1)P(X_3 \mid X_{1:2}) \ldots P(X_n \mid X_{1:n-1})$$

$$= \prod_{k=1}^{n} P(X_k \mid X_{1:k-1})$$

Applied to a sentence:

$$P(w_{1:n}) = P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_{1:2}) \ldots P(w_n \mid w_{1:n-1})$$

$$= \prod_{k=1}^{n} P(w_k \mid w_{1:k-1})$$

P("I write this sitting in the kitchen sink") = P("I")P("write" | "I")P("this" | "I write") . . . P("sink" | "I write this sitting in the kitchen")

The Chain Rule

Great, so now we have:

P(“I”)P(“write” | “I”)P("this" | "I write") . . . P("sink" | "I write this sitting in the kitchen")

versus

P(“I”)P(“write” | “I”)P("this" | "I write") . . . P("knife" | "I write this sitting in the kitchen")

versus

P(“I”)P(“write” | “I”)P("this" | "I write") . . . P("chair" | "I write this sitting in the kitchen")

The Chain Rule

But, since we never saw "I write this sitting in the kitchen sink", we still don't have a way to calculate

p("sink" | "I write this sitting in the kitchen")

What can we do?

Markov Assumption

For a very simple language model, we could **estimate** this conditional probability by looking at a smaller context window: a single word.

$p(\text{sink} \mid \text{kitchen})$

$p(\text{knife} \mid \text{kitchen})$

$p(\text{chair} \mid \text{kitchen})$

$p(\text{sink} \mid \text{I write this sitting in the kitchen})$

$\approx p(\text{sink} \mid \text{kitchen})$

Bigram language model

A **bigram language model** is a language model that makes a **Markov assumption**: the probability of a word is conditioned solely on the previous word.

$$P(w_n \mid w_{1:n-1}) \approx p(w_n \mid w_{n-1})$$

p("I write this sitting in the kitchen sink") ≈

p(I) p(write | I) p(this | write) p(sitting | this) p(in | sitting)

p(the | in) p(kitchen | the) p(sink | kitchen)

Bigram language model

In a bigram model, we have:

p("I")p("write" | "I")p("this" | "write")p("sitting" | "this")p("in" | "sitting")p("the" | "in")p("kitchen" | "the")p("sink" | "kitchen")

versus

p("I")p("write" | "I")p("this" | "write")p("sitting" | "this")p("in" | "sitting")p("the" | "in")p("kitchen" | "the")p("knife" | "kitchen")

versus

p("I")p("write" | "I")p("this" | "write")p("sitting" | "this")p("in" | "sitting")p("the" | "in")p("kitchen" | "the")p("chair" | "kitchen")

Maximum Likelihood Estimates: Bigram

Let’s get some counts! Let's use the Brown corpus to estimate the probabilities with a Maximum Likelihood Estimate.

$p(\text{sink} \mid \text{kitchen}) = 4/138$

$p(\text{knife} \mid \text{kitchen}) = 2/138$

$p(\text{chair} \mid \text{kitchen}) = 1/138$

"I write this sitting in the kitchen sink" is a winner!

Generalizing to n-grams

We can improve our context by looking at larger window sizes.

Our bigram model predicts "knife" is a better completion than "chair" in our context, because kitchen knives are more frequent than kitchen chairs.

If we considered more context, we might be able to capture that "in the kitchen knife" is not a good completion because it is rare to be in knives.

Generalizing the n-gram model

Generic n-gram model:

$$P(w_n \mid w_{1:n-1}) \approx P(w_n \mid w_{n-N+1:n-1})$$

where N is the context window

Language Models for Language Generation

Language Generation

So far we have used language models to predict the next word in a sequence and estimate the probability of a sentence.

How do we **generate** sentences?

Language Generation

We sample words according to their estimated probabilities:

Bigram Model

P(english | want) = .0011

P(chinese | want) = .0065

P(to | want) = .66

P(eat | to) = .28

P(food | to) = 0

P(want | spend) = 0

P (i | <s>) = .25

Example adapted from Jurafsky & Martin, ed. 3

Language Generation

- Choose a random bigram (<s>, w) according to its probability.
- Then choose a random bigram (w, x) according to its probability.
- Repeat until we choose </s>.

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

eat Chinese
eat English
eat cake

Example adapted from Jurafsky & Martin, ed.

Evaluation

Slides adapted from Jurafsky & Martin, ed. 3

Evaluation: How good is our model?

Does our language model prefer good sentences to bad ones?

Does it assign higher probability to “real” or “frequently observed” sentences than “ungrammatical” or “rarely observed” sentences?

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.**

summing over all words in test text

$$PP(W) = \exp\left(-\frac{1}{N}\sum_{i}^{N}\log p(w_i \mid w_{<i})\right)$$

probability of the actual text according to our model

numerical stability

normalize by length

Minimizing perplexity is the same as maximizing probability

Lower perplexity = better model

Training 38 million words, test 1.5 million words, WSJ

N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

Neural Language Models

Problems with n-gram Language Models

Sparsity Problem 1

<u>Problem:</u> What if *"students opened their w_j"* never occurred in data? Then w_j has probability 0!

$$p(w_j \mid \text{students opened their}) = \frac{\text{count(students opened their } w_j\text{)}}{\text{count(students opened their)}}$$

Problems with n-gram Language Models

Sparsity Problem 1

Problem: What if *"students opened their* w_j*"* never occurred in data? Then w_j has probability 0!

→

(Partial) Solution: Add small δ to count for every $w_j \in V$. This is called *smoothing*.

$$p(w_j \mid \text{students opened their}) = \frac{\text{count(students opened their } w_j)}{\text{count(students opened their)}}$$

Problems with n-gram Language Models

Storage: Need to store count for all possible *n*-grams. So model size is O(exp(*n*)).

$$P(\boldsymbol{w}_j|\text{students opened their}) = \frac{\text{count}(\text{students opened their } \boldsymbol{w}_j)}{\text{count}(\text{students opened their})}$$

Increasing *n* makes model size huge!

Slides adapted from Mohit Iyyer

another issue:

- We treat all words / prefixes independently of each other!

students opened their ___

pupils opened their ___

scholars opened their ___

undergraduates opened their ___

students turned the pages of their ___

students attentively perused their ___

…

Shouldn't we *share information* across these semantically-similar prefixes?

Slides adapted from Mohit Iyyer

How Do We Represent Text?

To feed text into a neural network, we need to turn it into numbers. In our regression classifier, we did this by **hand-crafting features**. Now we're going to use neural networks to **learn representations** for us.

Word Embeddings

We represent text using word vectors.

Idea: a word meaning is based on its **distance** from other word meanings.

Each word = a vector (not just "good" or "w_{45}")

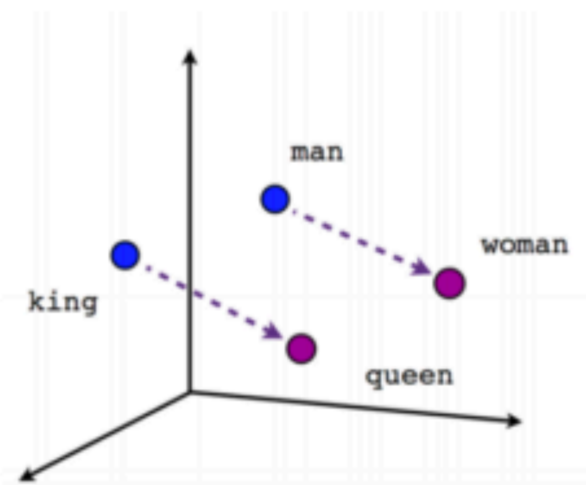
Similar words are "**nearby in semantic space**"

We build this space automatically by seeing which words are **nearby in text**

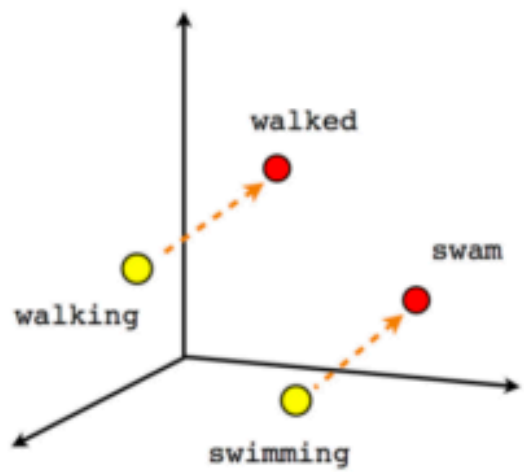
Word Embeddings

- represent words with low-dimensional vectors called **embeddings** (Mikolov et al., NIPS 2013)

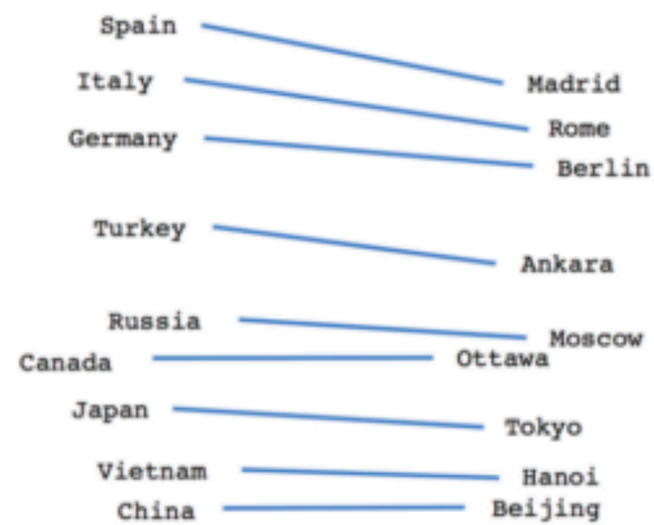
king =
[0.23, 1.3, -0.3, 0.43]



Male-Female

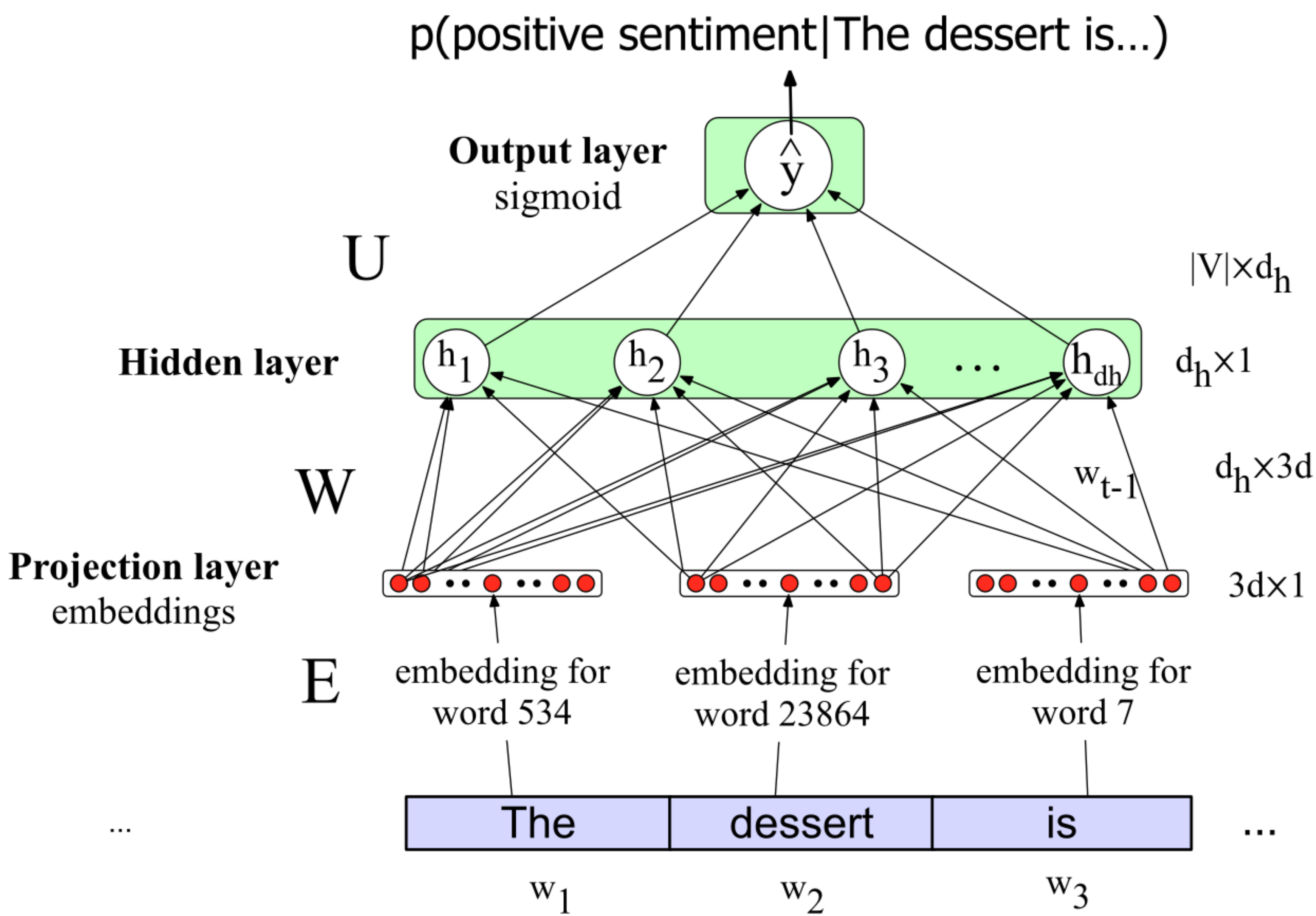


Verb tense



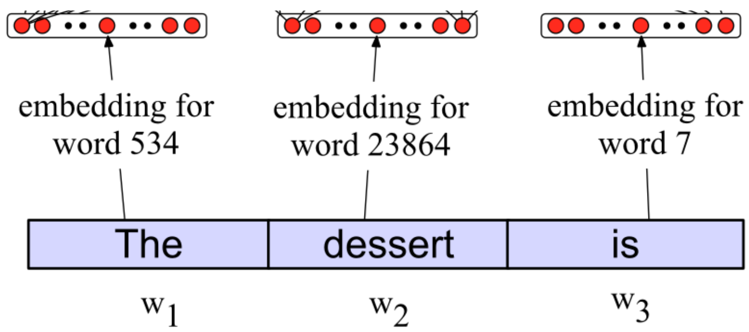
Country-Capital

Neural Net Classification with embeddings as input features!



Issue: texts come in different sizes

This assumes a fixed size length (3)!



Some simple solutions (more sophisticated solutions later)

1. Make the input the length of the longest review
   - If shorter then pad with zero embeddings
   - Truncate if you get longer reviews at test time
2. Create a single "sentence embedding" (the same dimensionality as a word) to represent all the words
   - Take the mean of all the word embeddings
   - Take the element-wise max of all the word embeddings
     - For each dimension, pick the max value from all words