ChatGPT prompt “minimalist landscape painting of a deep underwater scene with a blue tang fish in the bottom right corner”
Roadmap

Machine Learning Concepts

- Perceptron

- Fully Connected Neural Networks

- Convolutional Neural Networks
New data type: sequences

- Audio

- DNA

- Stock market

- Weather

What is the data property here that we could leverage?
Natural Language

“language that has developed naturally in use”
Natural Language

“language that has developed naturally in use”

Compare to constructed or formal language

- code: `for i in range(50):`
- math: 52 + 94 = 147
- logic: $A \land B \rightarrow C$ (if A and B, then C)
Natural Language

In this class: sequence of words

“They went to the grocery store and bought bread, peanut butter, and jam.”
Natural Language: Prediction tasks?

Example of prediction?

Input: X

I do not want sour cream in my burrito

Function: f

Output: Y

No quiero crema agria en mi burrito
Natural Language: Prediction tasks?

Example of classification?

Input: X

“The story telling was erratic and, at times, slow”

“Loved the diverse cast of this movie”

Output: Y

“Good review?”

Function: f
"They went to the grocery store and bought... bread?
milk?
rock?"

**Generating artificial sentences:** Here each word is a discrete unit; predicting the next part of the sequence means predicting words
Language models

Definition: Probability distribution over strings in a language.

Exponentially-many strings means each string has very low probability

Relative probabilities are meaningful:

\[ P(\text{"they went to the store"}) \gg P(\text{"butter dancing rock"}) \]
Language models logic: leverage sentence structure

\[ P(\text{any sequence}) \text{ is determined by } P(\text{the words in the sequence}). \]

Said differently, we can represent a sequence as \( w_1, w_2, \ldots w_n \), and

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

\[ P(\text{“they went to the store”}) = P(\text{“they”}) \times P(\text{“went”}|\text{“they”}) \times P(\text{“to”}|\text{“they went”}) \times \ldots \]

“The probability of a sentence is the product of the probabilities of each word given the previous words”

This is an application of the \textit{chain rule for probabilities}
Language models: weird & cool!

Model trained on the King James Bible, Structure and Interpretation of Computer Programs, and some of Eric S. Raymond's writings:

• The righteous shall inherit the land, and leave it for an inheritance unto the children of Gad according to the number of steps that is linear in b.

• 25:12 And thou shalt put into the heart of today’s IBM mainframe operating systems.

(King James Programming)

https://kingjamesprogramming.tumblr.com/
Language models: the math

At each step, we look at a probability distribution for what the next word might be.

They went to the grocery store and bought ..
"They went to the grocery store and bought bread, peanut butter, and jam."

They went to the grocery store and bought bread, peanut butter, and jam.

- Consistent casing
- Strip punctuation
- One word is one token
- Split on spaces

Aside: Tokenization itself can be challenging...

- A lot easier in English than other languages (e.g. Chinese)
  - Chinese is character-based; words & phrases have different character lengths
  - No spaces
Language models: the math

At each step, we look at a probability distribution for what the next word might be.

They went to the grocery store and bought ..

How do we know which words to calculate probabilities for?
Vocabularies: Defining a finite set of words

Vocabularies: the set of all words “known” to the model

Why?
- We need a finite set of words in order to define a discrete distribution over it.

How?
- Choose a hyperparameter `vocab_size` for how many words the model should know
- Keep only the `vocab_size` with most frequent words – replace everything else with “UNK”
Original sentence:

“They galloped to the Ratty for dinner, and ate exactly seventy-three waffle fries and chocolate peamilk.”
Vocabularies: how

- Original sentence:
  
  “They galloped to the Ratty for dinner, and ate exactly seventy-three waffle fries and chocolate peamilk.”

- Tokenized:
  
"They galloped to the Ratty for dinner, and ate exactly seventy-three waffle fries and chocolate peamilk."

Vocabularies: how

- Original sentence:
  
  “They galloped to the Ratty for dinner, and ate exactly seventy-three waffle fries and chocolate peamilk.”

- Tokenized:
  

- UNKed:
  
Language models: the math

At each step, we look at a probability distribution for what the next word might be.

*They went to the grocery store and bought ..*

How to calculate the probability for words in our vocabulary?
LM implementation: counting

- Goal: predict next word given a preceding sequence

  \[ P(\text{word}_n | \text{word}_1, \text{word}_2, ..., \text{word}_{n-1}) = \frac{\text{Count(\text{word}_1, \text{word}_2, ..., \text{word}_{n-1}, \text{word}_n)}}{\text{Count(\text{word}_1, \text{word}_2, ..., \text{word}_{n-1})}} \]
LM implementation: counting

- Goal: predict next word given a preceding sequence

  \[ P(\text{word}_n | \text{word}_1, \text{word}_2, \ldots \text{word}_{n-1}) = \frac{\text{Count}(\text{word}_1, \text{word}_2, \ldots \text{word}_{n-1}, \text{word}_n)}{\text{Count}(\text{word}_1, \text{word}_2, \ldots \text{word}_{n-1})} \]

- Example task: predict the next word
  - *he danced* ___
LM implementation: counting

- Goal: predict next word given a preceding sequence

\[
P(\text{word}_n | \text{word}_1, \text{word}_2, \ldots \text{word}_{n-1}) = \frac{\text{Count(word}_1,\text{word}_2,\ldots\text{word}_{n-1},\text{word}_n)}{\text{Count(word}_1,\text{word}_2,\ldots\text{word}_{n-1})}
\]

- Example task: predict the next word
  - he danced

- Strategy: iterate through all words in vocabulary, and calculate

\[
\frac{\text{Count(he danced <word>)}}{\text{Count(he danced)}} \quad \text{for each word}
\]
LM implementation: counting

- Our training sentences were:
  - “She danced happily”
  - “They sang beautifully”
  - “He danced energetically”
  - “He sang happily”
  - “She danced gracefully”
  - “He danced ___”
  - “He danced happily”

\[
\frac{\text{Count}(\text{he danced} \ < \text{word} \ >)}{\text{Count}(\text{he danced})}
\]

Why doesn’t this work?

This strategy depends on having instances of sentence prefixes.

Has 0 probability
LM implementation: N-gram counting

Improvement: N-gram model – only look at N words at a time
LM implementation: N-gram counting

Improvement: N-gram model – only look at N words at a time
(in this case, bigrams look at 2 words at a time)

- “She danced happily”
- “They sang beautifully”
- “He danced energetically”
- “He sang happily”
- “She danced gracefully”
LM implementation: N-gram counting

Improvement: **N-gram** model – only look at **N** words at a time
(in this case, **big**rams look at **2** words at a time)

- “danced happily”
- “sang beautifully”
- “danced energetically”
- “sang happily”
- “danced gracefully”

“He danced **happily**” now has 1/3 probability!

But what if the answer was “He danced **beautifully**”??
LM implementation

Problem: it’s impossible for the training set to have every possible valid sequence of words!

Let’s try to learn a better **numerical** representation

*What is the simplest thing you can think of?*
LM implementation: Simple approach

- “She danced happily”
- “They sang beautifully”
- “He danced energetically”
- “He sang happily”
- “She danced gracefully”

```
<table>
<thead>
<tr>
<th>they</th>
<th>danced</th>
<th>sang</th>
<th>happily</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
```

Any potential issues with this?
Problem: one-hot encoding does not capture any relationships between the words!

Can we learn a better numerical representation which associates related words with one another?
Embedding matrix

```
they  2  0  1  3  0  4
danced  0  1  1  0  2  1
sang  0  0  2  0  1  3
happily  0  1  1  1  0  2
gleefully  4  0  0  1  1  0
```

vocab_sz
Embedding matrix

```
<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>they</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>danced</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>sang</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>happily</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>gleeefully</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
```

Any questions?
Embedding matrix

- 2d matrix: \texttt{vocab\_sz} \times \texttt{embedding\_sz}

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Embedding matrix

- 2d matrix: `vocab_sz` x `embedding_sz`
- each word corresponds to an index, or word ID – hence the `vocab_sz` dimension
Embedding matrix

- 2d matrix: \( \text{vocab\_sz} \times \text{embedding\_sz} \)
- each word corresponds to an index, or word ID – hence the \( \text{vocab\_sz} \) dimension
- \( \text{embedding\_sz} \) is a hyperparameter

### How to build this embedding matrix?

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
LM implementation: deep learning

Deep learning helps solve this! How?

We can learn an embedding matrix that associates related words with one another for solving a prediction task.
Using the Embedding Matrix in a Network

If you want to input a [batch of] words into a neural net, this is how:

```
they, danced, happily
```

1

**Input: word indices**

```
batch_sz
```

**index lookups**

```
vocab_sz
```

**embedding matrix**

```
embedding_sz
```

```
batch_sz
```

**Embedding of each word in batch**

Rest of model...

Critical bit: the entries of this matrix can be learned!

The network learns what word embeddings are most effective for performing its task.
Using the Embedding Matrix in a Network

Let’s look at the 0\textsuperscript{th} word in this batch; its ID in the vocab is 2.

they, danced, happily

1

2

batch\_sz

index lookups

eMBEDDING MATRIX

eMBEDDING\_sz

batch\_sz

Embedding of each word in batch

Rest of model…
Using the Embedding Matrix in a Network

So we look at row 2 of the embedding matrix.

they, danced, happily
Using the Embedding Matrix in a Network

We can then pull out this embedding so we can use it in the rest of the model!

they, danced, happily

```
  1
  2

batch_sz

index lookups

2 6 3 3 1 4 0
```

```
2 6 3 3 1 4 0
```

embedding_sz

```
2 6 3 3 1 4 0
```

batch_sz

Embedding of each word in batch

Rest of model...
Using the Embedding Matrix in a Network

In tensorflow, we can use

\texttt{tf.nn.embedding_lookup}

which takes in an embedding matrix and a list of indices, and returns the embedding corresponding to each index.
What does the embedding matrix represent?

• Each row in the matrix can be viewed as a vector in vector space

Example 2-D vector space:

Vocab size: 3
Embed size: 2
What does the embedding matrix represent?

• Each row in the matrix can be viewed as a vector in vector space

• “Embedding”: We’re embedding a non-Euclidian entity [a word] into Euclidian space
What does the embedding matrix represent?

• Each row in the matrix can be viewed as a vector in vector space
• “Embedding”: We’re embedding a non-Euclidian entity [a word] into Euclidian space
• Each row represents the “embedding” for a single word

Example 2-D vector space:

Vocab size: 3
Embed size: 2
What does the embedding matrix represent?

- Each row in the matrix can be viewed as a vector in vector space.
- “Embedding”: We’re embedding a non-Euclidian entity [a word] into Euclidian space.
- Each row represents the “embedding” for a single word.
- This has pretty remarkable properties!
Vector arithmetic in the embedding matrix

Demo [here](#)
More ‘semantic directions’ in embedding space

Semantic: relating to meaning in language
More ‘semantic directions’ in embedding space

\[
\begin{align*}
E(\text{queen}) - E(\text{king}) & \approx E(\text{woman}) - E(\text{man}) \\
E(\text{swam}) - E(\text{swimming}) & \approx E(\text{walked}) - E(\text{walking})
\end{align*}
\]

Semantic: relating to meaning in language
More ‘semantic directions’ in embedding space

Any questions?

Semantic: relating to meaning in language

Male-Female

E(queen) – E(king) ≈ E(woman) – E(man)

Verb tense

E(walked) – E(walking) ≈ E(swam) – E(swimming)

Country-Capital

E(Spain) – E(Madrid) ≈ E(Vietnam) – E(Hanoi)
Why do embedding matrices work like this?

When the language model is trained, it’s incentivized to put words with similar context near each other in the embedding space.
Why do embedding matrices work like this?

Let’s say in the middle of training...

P(“happily” | “they danced”) = high

P(“gleefully” | “they danced”) = low

Then, the model sees a lot of “danced gleefully”

How do we increase P(“gleefully” | “they danced”)?
Why do embedding matrices work like this?

Let’s say in the middle of training...

\[
\begin{align*}
    P(\text{“happily”}| \text{“they danced”}) &= \text{high} \\
    P(\text{“gleefully”}| \text{“they danced”}) &= \text{low}
\end{align*}
\]

Since probability is calculated based on the embedding matrix...

Modify the embedding of “gleefully” so that it’s similar to the embedding of “happily”!

Context-based learning!

Input: word indices

1

Input: word indices

index lookups

embedding matrix

embedding size

vocab size

batch size

embedding of each word in batch

batch size

Rest of model...
Quantifying “similarity”

\[
\text{cosine similarity} = \cos(\theta) = \frac{A \cdot B}{||A|| ||B||} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}
\]

\[
\cos(0^\circ) = 1 \\
\cos(90^\circ) = -0.448 \\
\cos(180^\circ) = -0.598
\]
Limitations of the context-based approach

- Context is correlated with meaning, but context ≠ meaning
- Synonyms typically have similar context:
  - \( P(\text{“happily”} \mid \text{“they danced”}) \)
  - \( P(\text{“gleefully”} \mid \text{“they danced”}) \)
- ...but often antonyms do, too:
  - \( P(\text{“happily”} \mid \text{“they danced”}) \)
  - \( P(\text{“unwillingly”} \mid \text{“they danced”}) \)
- “happily” and “unwillingly” might be used in similar contexts, but have the \textbf{opposite} meaning → a language model might (erroneously) give them similar embeddings
Other failure modes are even more dire

What happens when your dataset reflects historical / societal biases?
Other failure modes are even more dire

What happens when your dataset reflects historical / societal biases?

**Google News word2vec:**
- Large set of *pretrained* word embeddings, published 2013
- Dataset: news articles aggregated by Google News (100 billion words)
Other failure modes are even more dire

What happens when your dataset reflects historical / societal biases?

**Google News word2vec:**
- Large set of *pretrained* word embeddings, published 2013
- Dataset: news articles aggregated by Google News (100 billion words)

What kinds of relationships do these embeddings contain?
Google News word2vec

http://www.mattkenney.me/google-word2vec-biases/
Why did this happen?
- The training dataset (news articles) was biased.
- The news cycle over-represents crimes by black perpetrators
  - (Entman 94, Gilliam et.al. 96, Dixon 08, Dixon 15) – this is true over time as well
- Viewers respond more strongly to news stories about crimes by black perpetrators.
  - (Dixon and Maddox 06, Dixon and Azocar 07, Hurley et.al. 15)
  - (News outlets optimize for clicks, therefore report crime by black people more)
why are black women so angry
why are black women so loud
why are black women so mean
why are black women so attractive
why are black women so lazy
why are black women so annoying
why are black women so confident
why are black women so sassy
why are black women so insecure

ALGORITHMS
OF
OPPRESSION

HOW SEARCH ENGINES REINFORCE RACISM

SAFIYA UMOJA NOBLE
• In ~2010, when Noble started working on this book, these were the real Google autocomplete suggestions

• Takeaway: language models reproduce the biases of the data on which they are trained
  • ...unless special care is taken—we have an upcoming lab on this!
• Think about the algorithms behind autocomplete, or ad recommendation...

• The math might be cool, but there’s more to algorithms than math. It is important to consider their potential ethical and social implications once deployed
Recap

Language modeling

- Natural Language
- Counting/N-grams
- Limitations of traditional methods

Learning embedding matrix

- Useful properties of embeddings
- Limitations of context-based learning

And this I pray, that your love may abound yet more and more like a controlled use of shared memory.