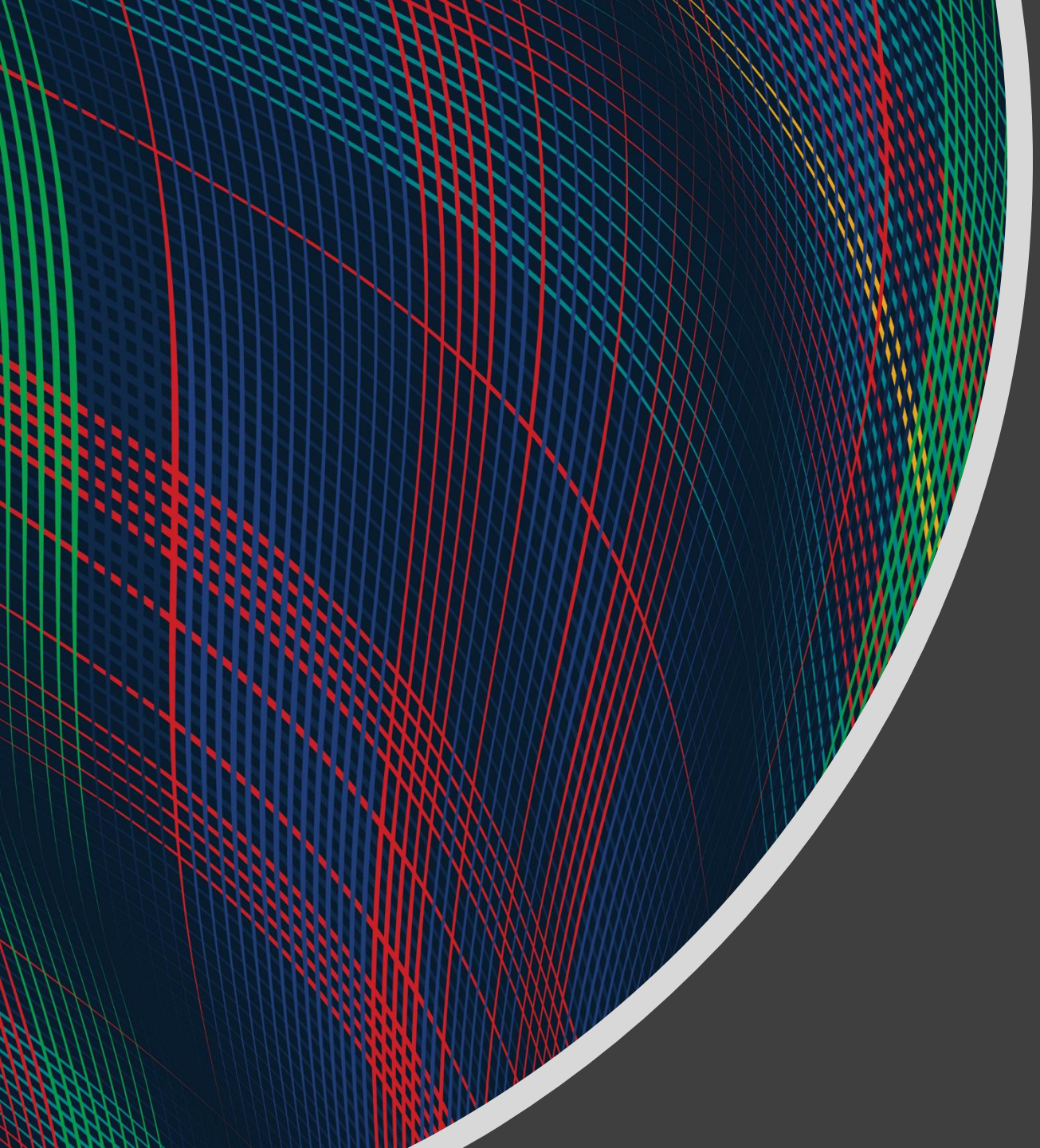


10-315 Introduction to ML

Natural Language Processing

Instructor: Pat Virtue

Natural Language Processing (NLP)

Practical Deep Learning

NLP Intro

Feature Engineering/Learning for Text

N-gram Language Models

Word Embedding Language Models

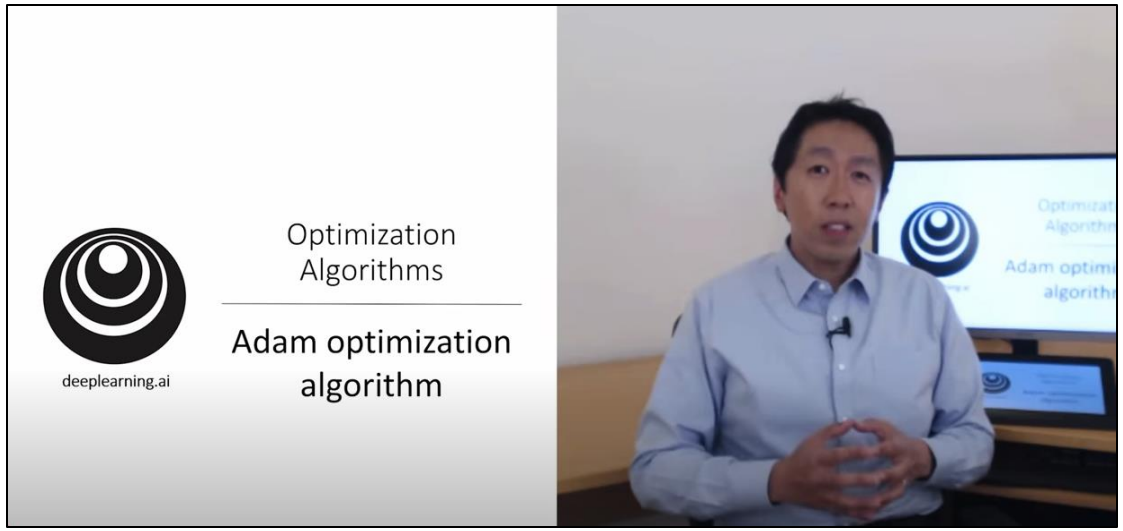
Practical Deep Learning

Deep learning issues and Fixes

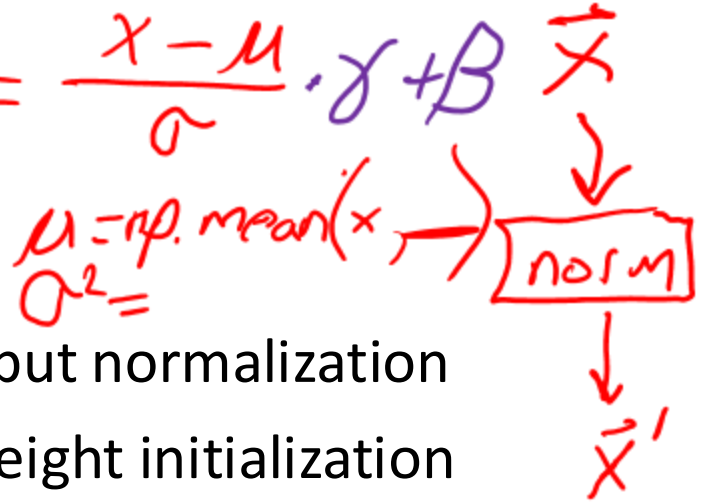
Issues

- Numerical Stability
- Vanishing/Exploding gradients
- Overfitting

[Andrew Ng videos on Hyperparameter Tuning!]



Fixes



- Input normalization
- Weight initialization
- Batch normalization
- Adam
  - Exponential moving average
  - Momentum
  - RMSprop
- Learning rate decay
- Skip connections
- Dropout

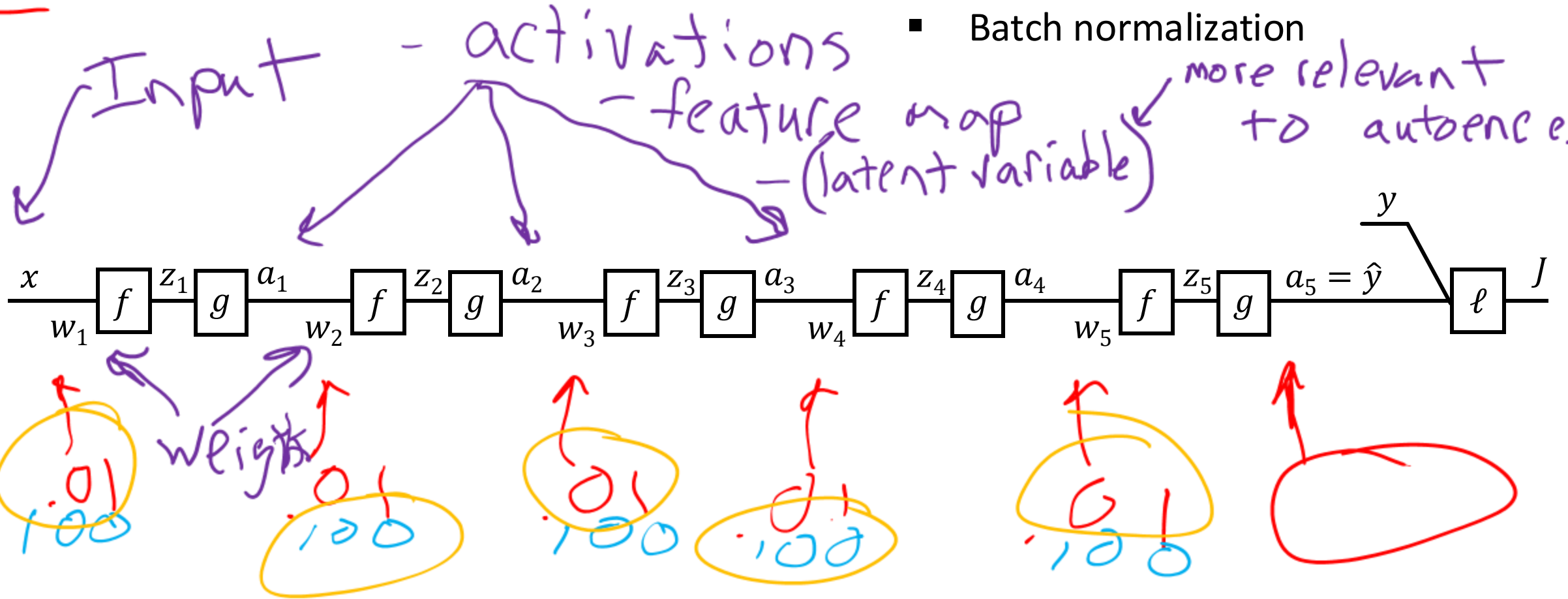
Deep learning issues and Fixes

Issues

- Numerical Stability
- Vanishing/Exploding gradients

Fixes

- Input normalization
- Weight initialization
- Batch normalization



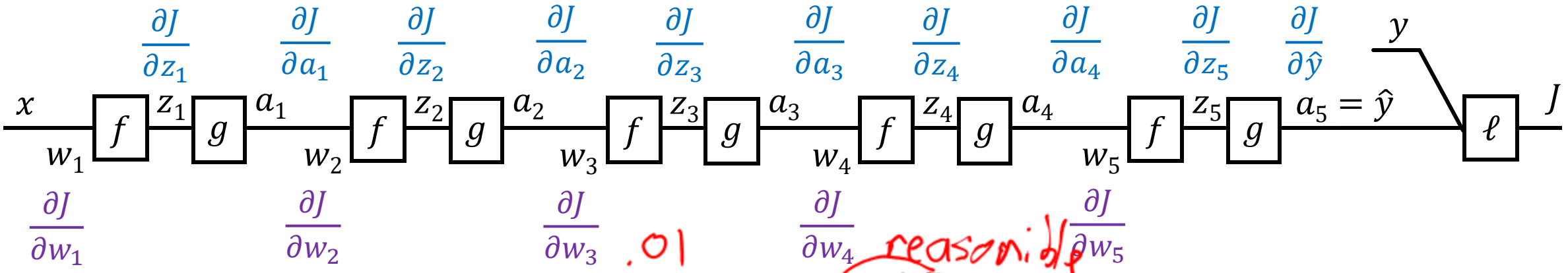
Deep learning issues and Fixes

Issues

- Numerical Stability
- Vanishing/Exploding gradients

Fixes

- Input normalization
- Weight initialization
- Batch normalization



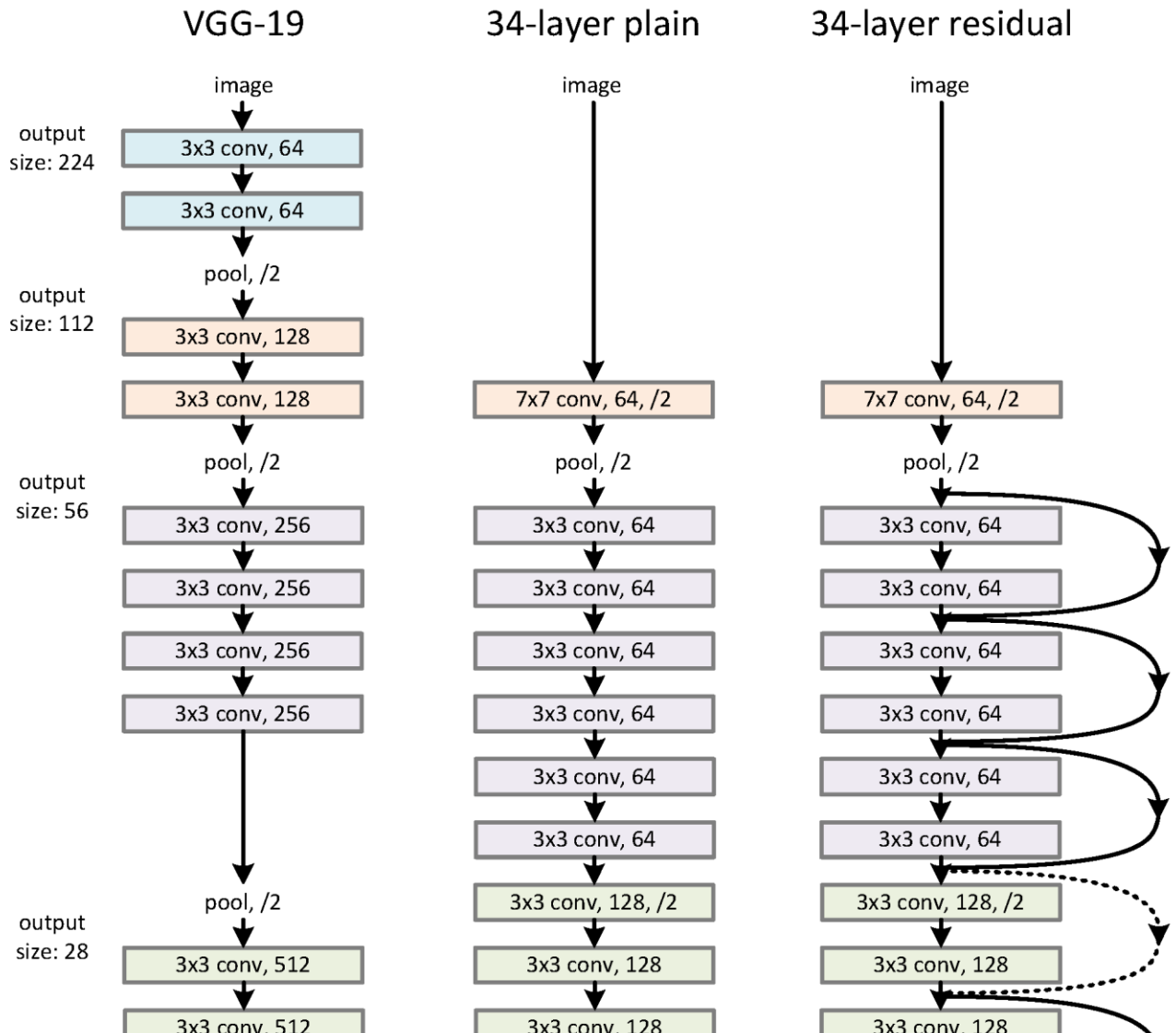
.01 reasonable

$$w_j \leftarrow w_j - \alpha \frac{\partial J}{\partial w_j}$$

100 2

Deep learning issues and Fixes

Issues



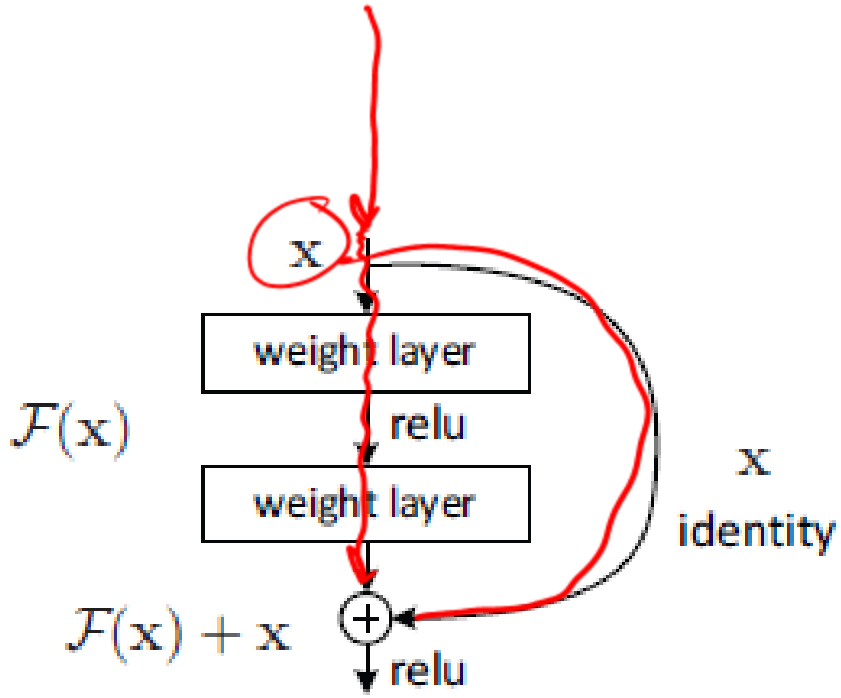


Figure 2. Residual learning: a building block

Natural Language Processing (NLP)

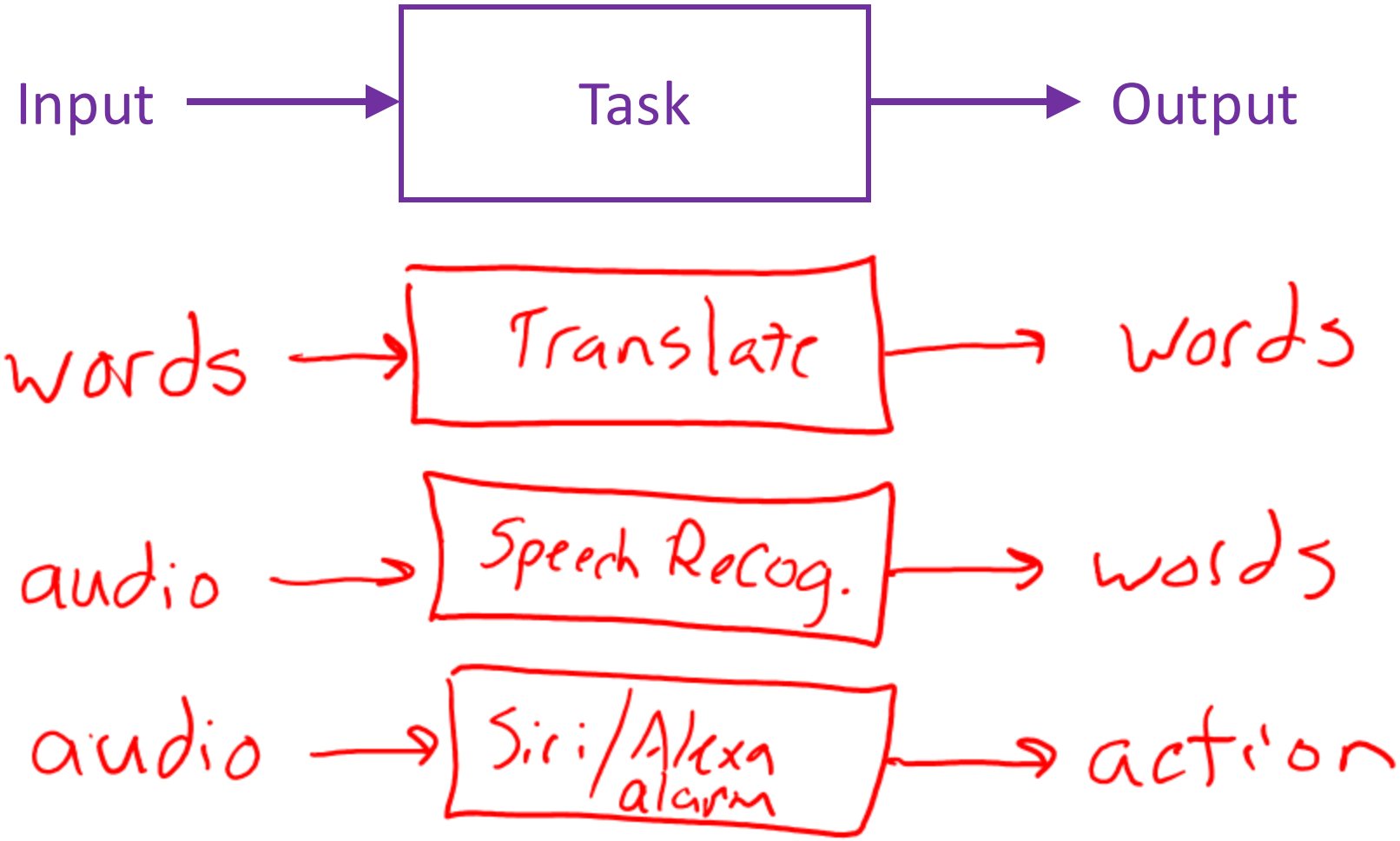
Language is Hard

Emphasis can drastically change meaning

I didn’t eat your dog

NLP Tasks Examples

How many different NLP Input/Output agents can you think of?

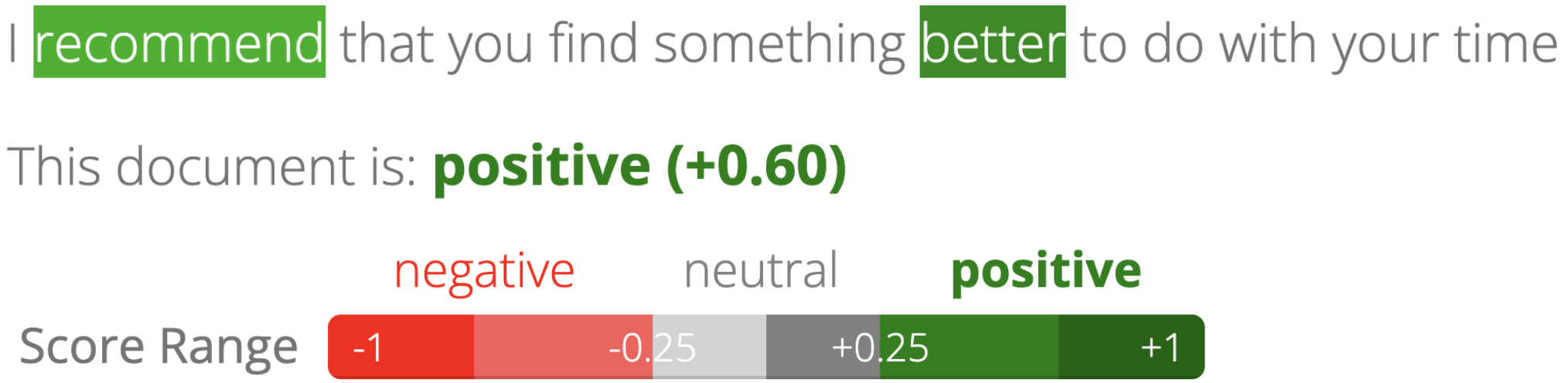


NLP Task: Sentiment Analysis

Sentiment analysis demo

https://text2data.com/Demo

“I recommend that you find something better to do with your time”



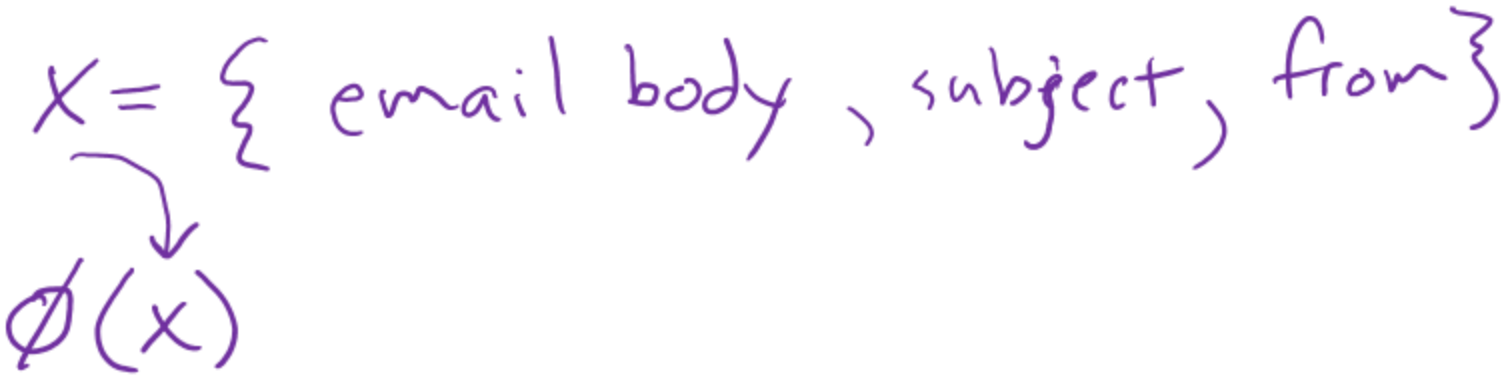
raw

"i will not eat"

Text Features

Text Features

SPAM Classification



Text Features

SPAM Classification

$x = \{$ email body, subject, from $\}$

$\phi(x)$

$$\begin{bmatrix} \phi_1(x) \\ \phi_2(x) \\ \\ \\ \end{bmatrix}$$

$\phi_1(x)$ ← # money

$\phi_2(x)$ ← # All caps words

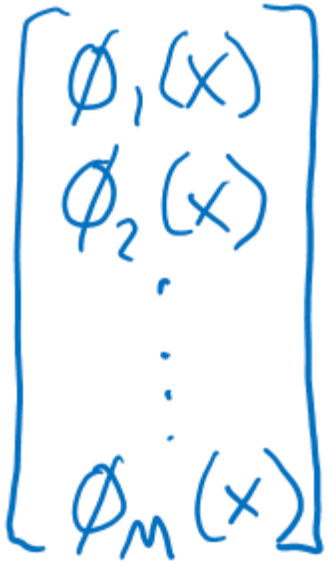
← 0/1 My name

Text Features

Bag of words: Vector of length the size of the vocabulary

Two options

- Word occurrence: Binary: does the word exist (at least once) or not
- Word histogram: Integer: count of how many times the word appears



Text Features

Predicting Rating from Written Movie Review

$x = \{\text{text}\}$

$\phi(x) \in \mathbb{R}^M$

$$\begin{bmatrix} \phi_1(x) \\ \phi_2(x) \\ \vdots \\ \phi_M(x) \end{bmatrix} \leftarrow \#\ \text{times fantastic}$$

$$\hat{y} = h(\vec{x})$$

$$= g(\vec{\theta}^T \vec{x})$$

$$= g(\vec{\theta}^T \phi(x))$$

Text Features: Tokenization

Text encoding: token → index within vocabulary

What is our "vocabulary", i.e., what are our tokens?

- Characters
- Byte pair encoding
- Words and punctuation

Corpus

```
I am Sam. I am
Sam. Sam I am.
That Sam-I-am.
That Sam-I-am!
```

Text Features : Tokenization

```
I am Sam. I am Sam. Sam I am.
That Sam-I-am. That Sam-I-am!
```

Text encoding: token → index within vocabulary

What is our "vocabulary", i.e., what are our tokens?

Character token vocabulary

```
 0: !
 1: -
 2: .
 3: I
 4: S
 5: T
 6: a
 7: h
 8: m
 9: t
10: _ (space)
```

BPE token vocabulary

```
 0: !
 1: -
 2: .
 3: I
 4: S
 5: T
 6: a
 7: h
 8: m
 9: t
10: _(space)
11: Sam
12: Th
13: am
14: _Sam
15: _am
```

Word token vocabulary

```
0: !
1: .
2: am
3: I
4: Sam
5: That
6: Sam-I-am
```

Text Features : Tokenization

```
I am Sam. I am Sam. Sam I am.
That Sam-I-am. That Sam-I-am!
```

Text encoding: token → index within vocabulary

Byte pair encoding (BPE)

Current token vocabulary

(Initialize to characters in corpus)

```
!       h
-       m
.       t
I       _  (space)
S
T
a
```

Pair frequencies

Count frequencies of all pairs of current tokens appearting together in corpus

```
_a: 3
am: 10
_S: 4
Sa: 5
_I: 1
Th: 2
ha: 2
at: 2
```

New tokens

Add most frequent pair to tokens

```
!       h
-       m
.       t
I       _  (space)
S       am
T
a
```

Text Features : Tokenization

```
I am Sam. I am Sam. Sam I am.
That Sam-I-am. That Sam-I-am!
```

Text encoding: token → index within vocabulary

Byte pair encoding (BPE)

Current token vocabulary

(Initialize to characters in corpus)

```
!
-
.
I
S
T
a
h
m
t
_ (space)
am
```

Pair frequencies

Count frequencies of all pairs of current tokens appearting together in corpus

```
_S:  4
_I:  1
Th:  2
ha:  2
at:  2
_am: 3
Sam: 5
```

New tokens

Add most frequent pair to tokens

```
!
-
.
I
S
T
a
h
m
t
_ (space)
am
Sam
```

Text Features : Tokenization

```
I am Sam. I am Sam. Sam I am.
That Sam-I-am. That Sam-I-am!
```

Text encoding: token → index within vocabulary

What is our "vocabulary", i.e., what are our tokens?

Character token vocabulary

```
 0: !
 1: -
 2: .
 3: I
 4: S
 5: T
 6: a
 7: h
 8: m
 9: t
10: _ (space)
```

BPE token vocabulary

```
 0: !
 1: -
 2: .
 3: I
 4: S
 5: T
 6: a
 7: h
 8: m
 9: t
10: _(space)
11: Sam
12: Th
13: am
14: _Sam
15: _am
```

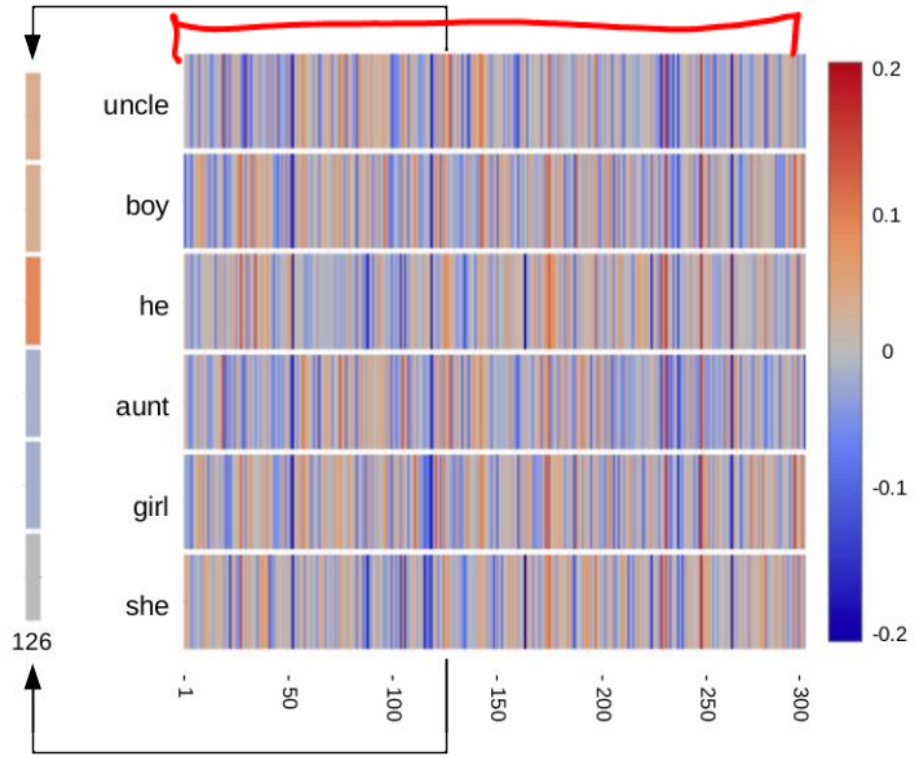
Word token vocabulary

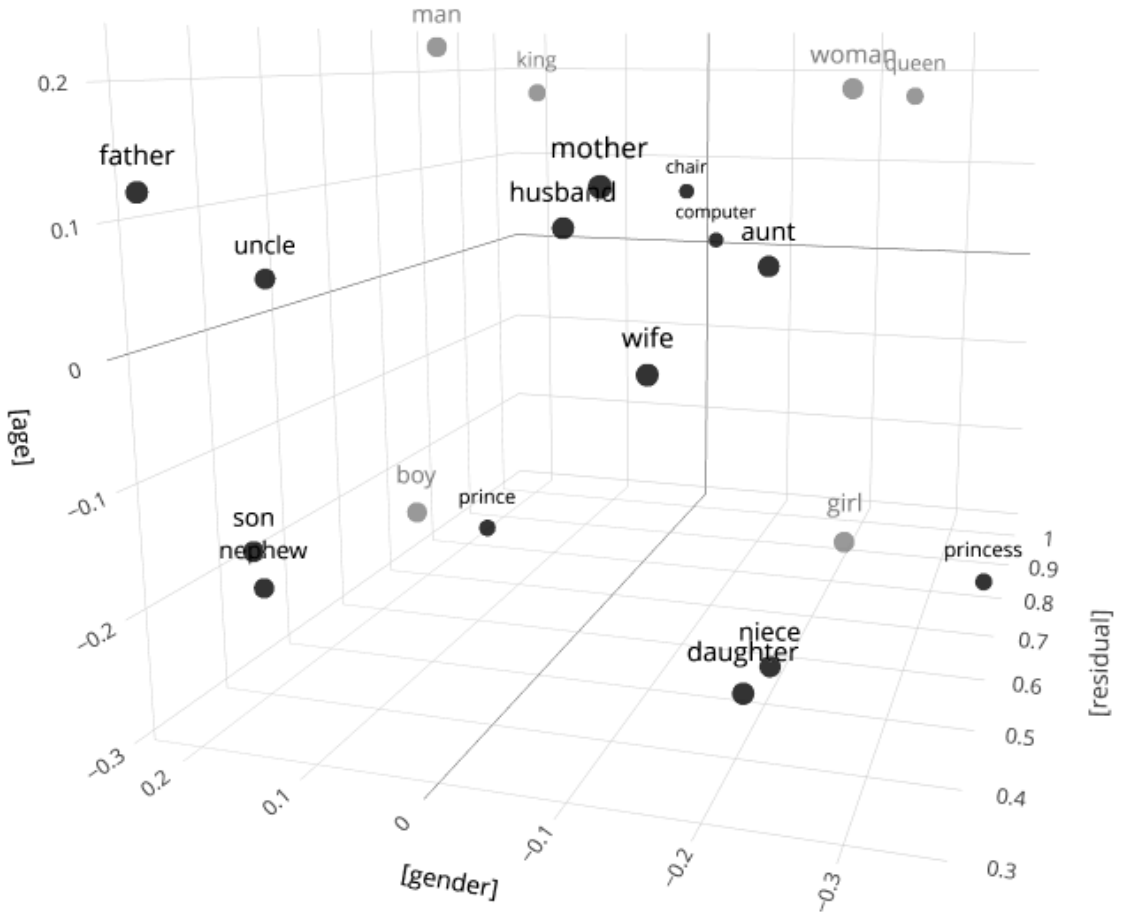
```
0: !
1: .
2: am
3: I
4: Sam
5: That
6: Sam-I-am
```

Text Feature Learning

Feature learning

Word to Vec

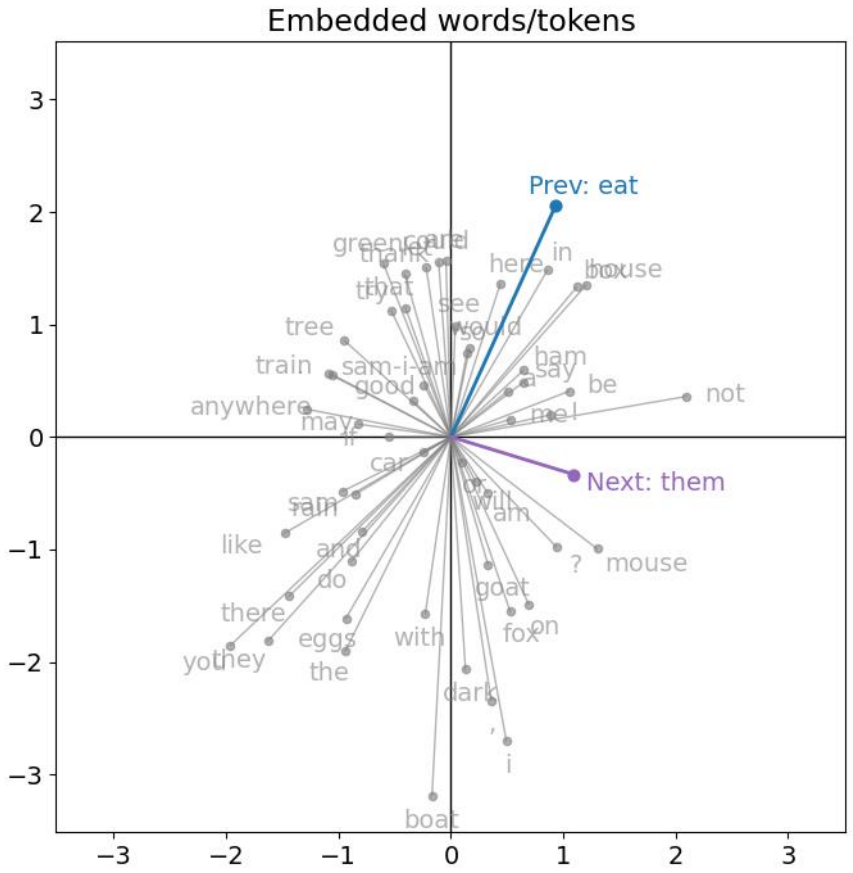


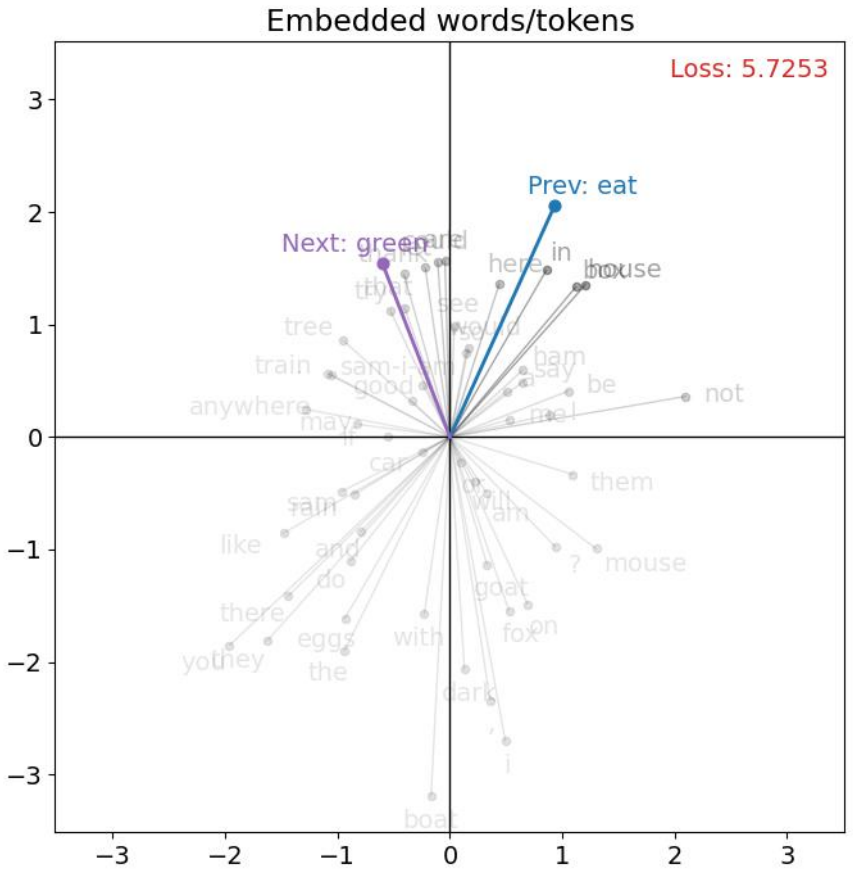


https://www.cs.cmu.edu/~dst/WordEmbeddingDemo/tutorial.html

Text Feature Learning

Word embeddings





Feature Engineering vs Feature Learning

Feature engineering

- Humans decide what the features may be useful
- Humans implement feature extraction algorithms

Feature learning

- Humans choose data and performance measure
- Humans decide on structure/algorithm to learn features
  - Features are just intermediate values between input and output
  - Structure defines number of feature values
- Allow machine to map data to feature values as needed

Feature Engineering vs Feature Learning

Feature engineering

- Humans decide what the features may be useful
  - Leverage human experience for that input
  - Consider data available
- Humans implement feature extraction algorithms
  - Leverage data and compute power

Pro: Human interpretable features → trained models easier to explain

Con: Features may not be sufficient for effective training

Con: Humans likely needed to adapt to new tasks

Pro: Less likely to overfit as humans have selected impactful features

Feature Engineering vs Feature Learning

Feature learning

- Humans choose data and performance measure
- Humans decide on structure/algorithm to learn features
  - Features are just intermediate values between input and output
  - Structure defines number of feature values
- Allow machine to map data to feature values as needed

Con: Humans cannot interpret features → trained models unexplainable

Pro: Allow machine to search larger space for efficient features

Pro: Humans may not be needed to adapt to new tasks (just new/more data)

Con: Can overfit to data as there is no human logic in feature definition

Natural Language Processing (NLP)

- Practical Deep Learning
- NLP Intro
- Feature Engineering/Learning for Text
- N-gram Language Models
- Word Embedding Language Models

Language Models

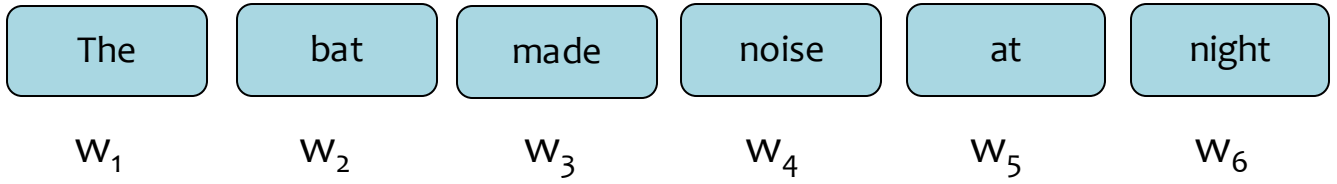
Language Models

N-grams

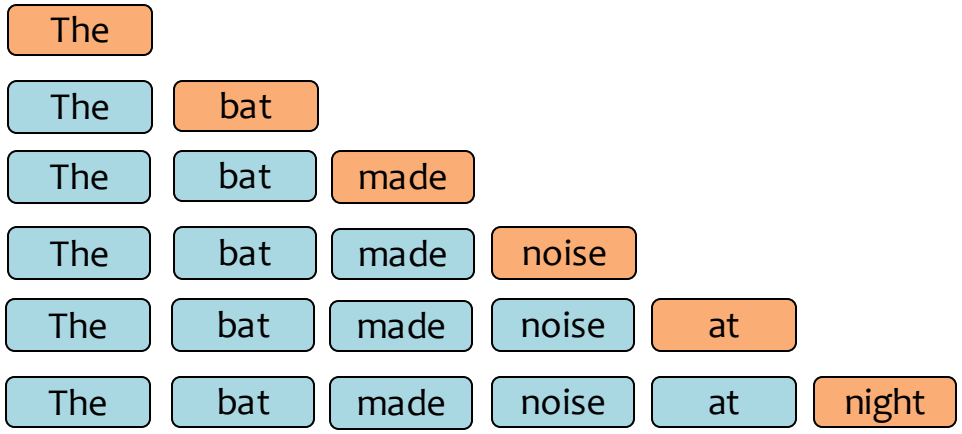
N-gram Exercise

The Chain Rule of Probability

Question: How can we **define** a probability distribution over a sequence of length T?



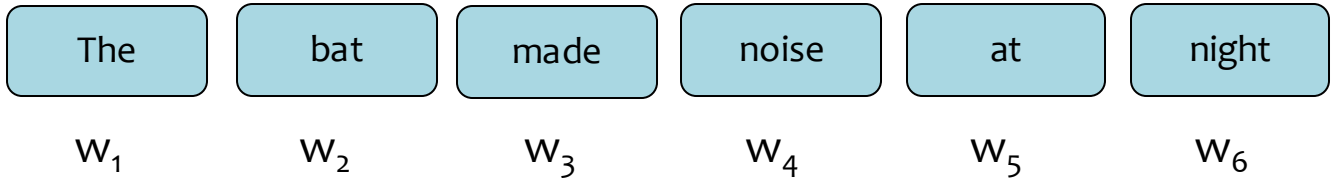
$$p(w_1, w_2, w_3, \ldots, w_6) =$$



$$p(w_1)$$
$$p(w_2 \mid w_1)$$
$$p(w_3 \mid w_2, w_1)$$
$$p(w_4 \mid w_3, w_2, w_1)$$
$$p(w_5 \mid w_4, w_3, w_2, w_1)$$
$$p(w_6 \mid w_5, w_4, w_3, w_2, w_1)$$

n-Gram Language Model

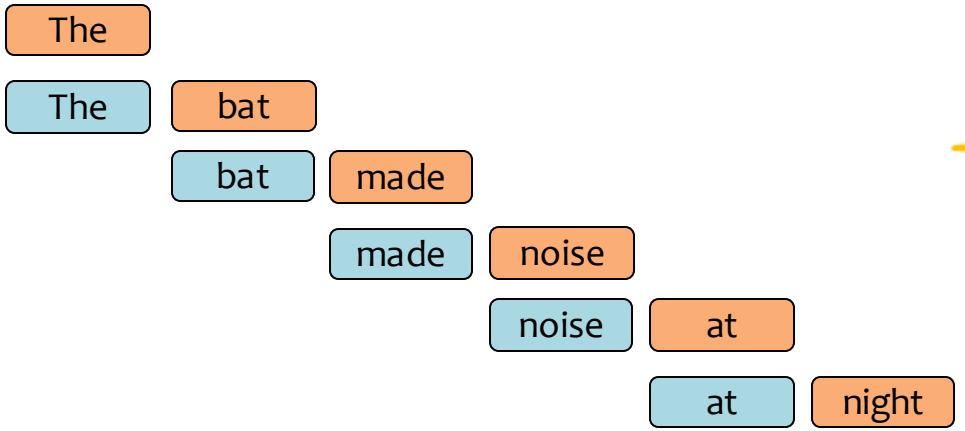
Question: How can we **define** a probability distribution over a sequence of length T?



n-Gram Model (n=2)

$$p(w_1, w_2, \ldots, w_T) = \prod_{t=1}^{T} p(w_t \mid w_{t-1})$$

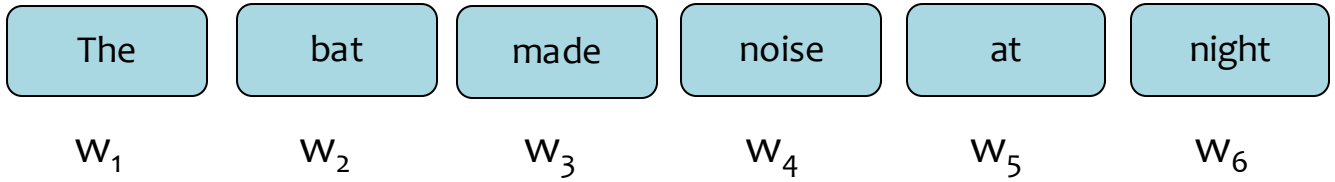
$p(w_1, w_2, w_3, \ldots, w_6) =$



$p(w_1)$
$p(w_2 \mid w_1)$
$p(w_3 \mid w_2)$
$p(w_4 \mid w_3)$
$p(w_5 \mid w_4)$
$p(w_6 \mid w_5)$

n-Gram Language Model

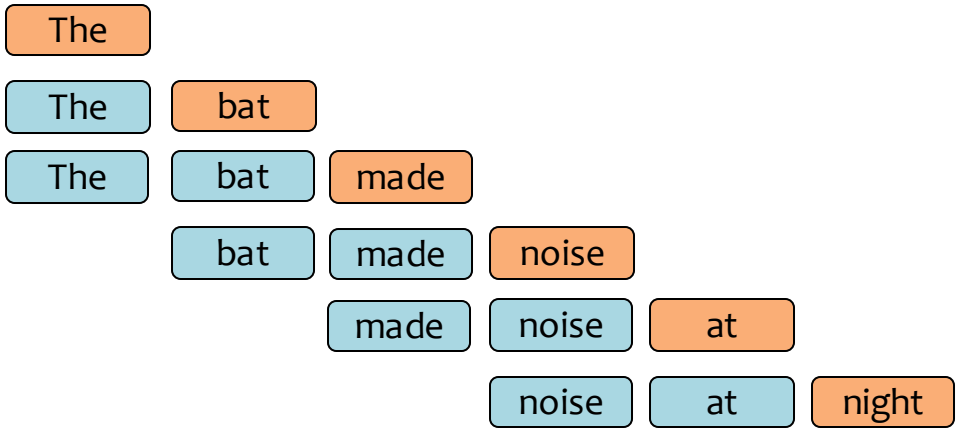
Question: How can we **define** a probability distribution over a sequence of length T?



n-Gram Model (n=3)

$$p(w_1, w_2, \ldots, w_T) = \prod_{t=1}^{T} p(w_t \mid w_{t-1}, w_{t-2})$$

$p(w_1, w_2, w_3, \ldots, w_6) =$



$p(w_1)$
$p(w_2 \mid w_1)$
$p(w_3 \mid w_2, w_1)$
$p(w_4 \mid w_3, w_2)$
$p(w_5 \mid w_4, w_3)$
$p(w_6 \mid w_5, w_4)$

N-gram Training

Where do the n-gram probabilities come from?

Google n-grams demo

N-gram Training

N-gram probabilities

Vocabulary size: 30,000

N-gram Training

Self-supervised

Example: Jane Austen, *Pride and Prejudice*

Vanity and pride are different things, though the words are often used synonymously. A person may be proud without being vain. Pride relates more to our opinion of ourselves, vanity to what we would have others think of us.

N-gram Training

Self-supervised learning (auto-regressive)

Example: Jane Austen, *Pride and Prejudice*

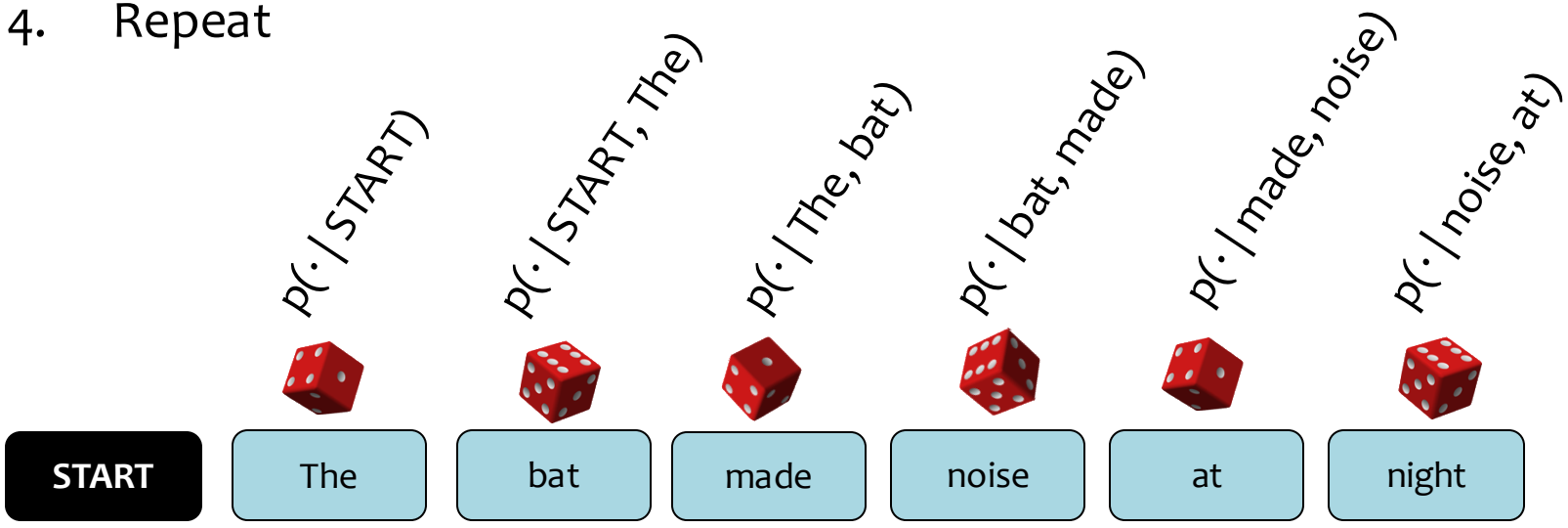
Vanity and pride are different things, though the words are often used synonymously. A person may be proud without being vain. Pride relates more to our opinion of ourselves, vanity to what we would have others think of us.

Sampling from a Language Model

Question: How do we sample from a Language Model?

Answer:

1. Treat each probability distribution like a (50k-sided) weighted die
2. Pick the die corresponding to $p(w_t \mid w_{t-2}, w_{t-1})$
3. Roll that die and generate whichever word w_t lands face up
4. Repeat



N-gram Examples

Random samples from language model trained on Shakespeare:

n=1: "in as , stands gods revenge ! france pitch good in fair hoist an what fair shallow-rooted , . that with wherefore it what a as your . , powers course which thee dalliance all"

n=2: "look you may i have given them to the dank here to the jaws of tune of great difference of ladies . o that did contemn what of ear is shorter time ; yet seems to"

n=3: "believe , they all confess that you withhold his levied host , having brought the fatal bowels of the pope ! ' and that this distemper'd messenger of heaven , since thou deniest the gentle desdemona ,"

n=7: "so express'd : but what of that ? 'twere good you do so much for charity . i cannot find it ; 'tis not in the bond . you , merchant , have you any thing to say ? but little"

This is starting to look a lot like Shakespeare… because it is Shakespeare

Slide: CMU, Zico Kolter