



# Cornell Bowers C-IS

College of Computing and Information Science

# Deep Learning

Week 02: Word Embeddings

Thanks to:  
Varsha Kishore  
Justin Lovelace  
Vivian Chen  
Anissa Dallmann  
Wentao Guo

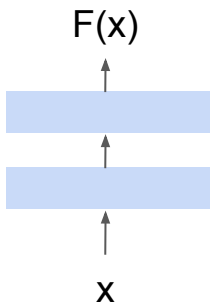
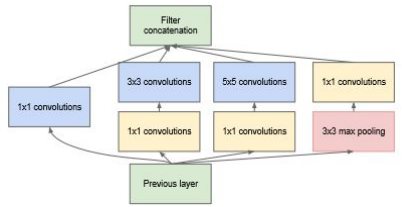
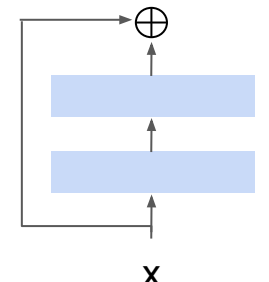
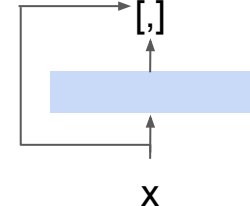
# Logistics

- HW1 is due on Today
- Submit on gradescope
  - If you worked in a group, create a group and then submit
- Clarifications are on Ed
- Come to office hours if you have questions

# Recap

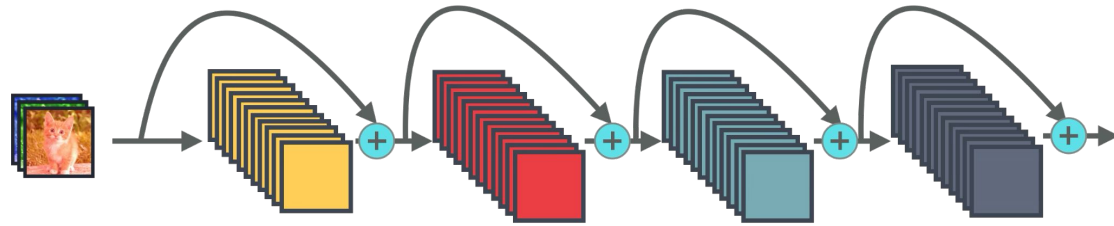
- MLPs
- Loss functions
- Optimizers
- Regularization
- Architecture types (CNNs for images)
  - Today: Words

# Summary of Image Models

“Plain” CNN	Google Net	ResNet	DenseNet
<p>Simple connection from previous to next layer</p>	<p>1x1, 3x3, 5x5 convolutions and pooling between each layer</p>	<p>Skip connections Add output of previous layer to next layer</p>	<p>Dense connections Concatenate output of previous layer to next layer</p>
 <p>The diagram shows a vertical flow from input <math>X</math> through a single blue rectangular layer to output <math>F(x)</math>.</p>	 <p>The diagram shows a 'Previous layer' feeding into a parallel structure of four paths: 1x1 convolutions, 3x3 convolutions, 5x5 convolutions, and 3x3 max pooling. The outputs of these paths are combined in a 'Filter concatenation' block.</p>	 <p>The diagram shows input <math>X</math> entering a blue layer. The output of the layer is added to the original input <math>X</math> at a summation node (<math>\oplus</math>).</p>	 <p>The diagram shows input <math>X</math> entering a blue layer. The output of the layer is concatenated with the original input <math>X</math> at a concatenation node (<math>[,]</math>).</p>

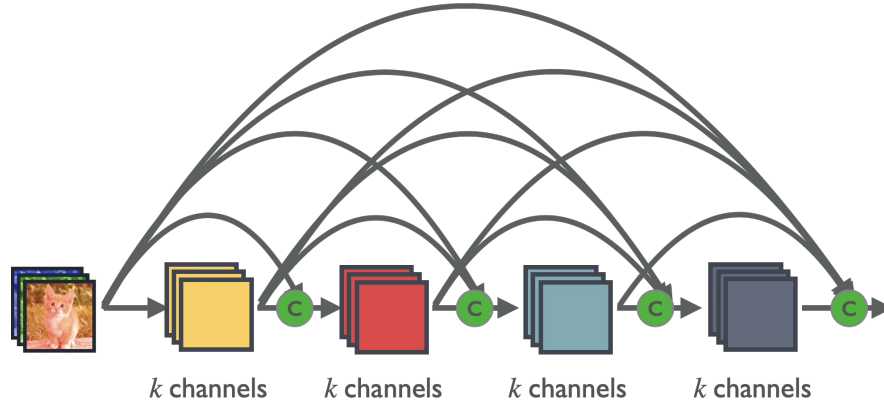
# From ResNets to DenseNets

ResNet



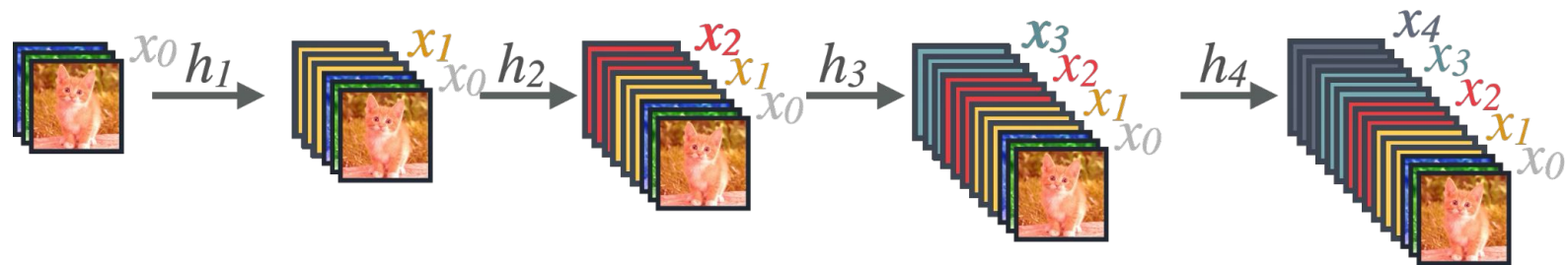
⊕ : Element-wise addition

DenseNet



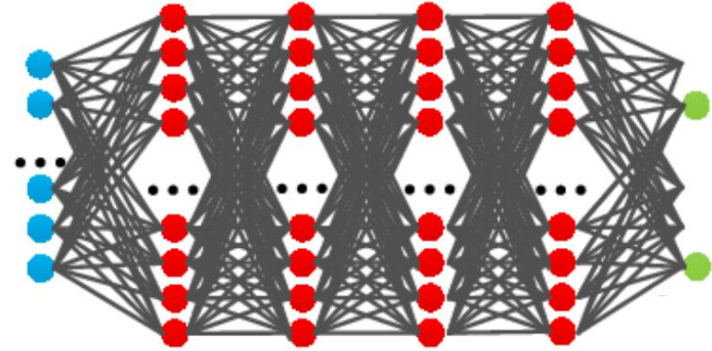
$k$  channels    $k$  channels    $k$  channels    $k$  channels

# DenseNets - feature concatenation



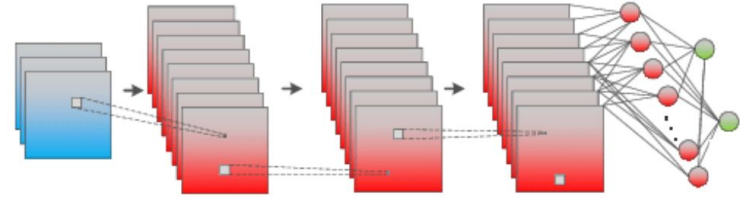
### MLPs:

- Fully connected layers
- Require more parameters and computational resources
- Flexible and can handle various input types



### CNNs:

- Convolutional layers with filters
- Designed specifically for structured input like images
- Inherently translation invariant due to shared weights
- Requires fewer parameters





How to handle text data?

# Cornell Bowers C-IS



What is an easy to make pasta recipe that features garlic and red sauce?




Here is a simple pasta recipe that features garlic and red sauce:

Ingredients:



- 1 pound pasta (such as spaghetti or penne)
- 1 tablespoon olive oil
- 2 cloves garlic, minced
- 1 (28-ounce) can crushed tomatoes
- 1/4 teaspoon salt
- 1/4 teaspoon black pepper
- 1/4 teaspoon red pepper flakes (optional)
- 1/2 cup grated Parmesan cheese
- Fresh parsley or basil, chopped, for garnish (optional)




san f 






- san francisco weather
- san francisco
- san francisco giants
- san fernando valley
- san francisco state university
- san francisco hotels
- san francisco 49ers
- san fernando
- san fernando mission
- san francisco zip code

Google Search I'm Feeling Lucky

Spanish - detected  ↔ English 

¿Cómo estás? |  ×

How are you?

     Verified

[Open in Google Translate](#) • [Feedback](#)

```
// write a binary search algorithm
const binarySearch = (arr, target) => {
  let left = 0;
  let right = arr.length - 1;
  let middle = Math.floor((left + right) / 2);
  while (arr[middle] !== target && left <= right) {
    if (target < arr[middle]) {
      right = middle - 1;
    } else {
      left = middle + 1;
    }
    middle = Math.floor((left + right) / 2);
  }
  return arr[middle] === target ? middle : -1;
}
```

*I don't know how to parallel park.*

*I'm taking my dog for a walk at the park.*

Homonyms

*We ate outside and swam in the lake all week.*

*We ate outside and in the lake all week.*

Typos

*Biden speaks to the media in Illinois.*

*The president greets the press in Chicago.*

Paraphrases/ Synonyms

*Although interchangeable, the body pieces on the 2 cars are not similar.*

*Although similar, the body pieces are not interchangeable on the 2 cars.*

Word order

# Language Modeling: predict the next word

**Assign probabilities to text.**

Given a sequence  $(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T)$ , we want to **maximize**  $P(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T)$ .

$$P(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T) = P(\mathbf{x}_1)P(\mathbf{x}_2|\mathbf{x}_1)P(\mathbf{x}_3|\mathbf{x}_1, \mathbf{x}_2)P(\mathbf{x}_4|\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) \dots P(\mathbf{x}_T|\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{T-1})$$

$P(\text{I like cats because they look cute}) = P(\text{I}) P(\text{like} | \text{I}) P(\text{cats} | \text{I like}) P(\text{as} | \text{I like cats}) P(\text{they} | \text{I like cats because})$

$P(\text{look} | \text{I like cats because they}) P(\text{cute} | \text{I like cats because they look})$

**Predict the next word given current text!**

# $n$ -Gram Language Model

$n$ -Gram: chunk of  $n$  consecutive words

**Count** the frequency of each  $n$ -grams and predict next word!

**Assume** each word only depends on previous  $n - 1$  words.

**Uni-gram:** “I” “like” “cats” “as” “they” “look” “cute”

**Bi-gram:** “I like” “like cats” “cats as” “as they” ...

**Tri-gram:** “I like cats” “like cats as” “cats as they” ...

$$\begin{aligned} P(\mathbf{x}_t | \mathbf{x}_1, \dots, \mathbf{x}_{t-1}) &= P(\mathbf{x}_t | \mathbf{x}_{t-n+1}, \dots, \mathbf{x}_{t-1}) \\ &= \frac{\text{count}(\mathbf{x}_{t-n+1}, \dots, \mathbf{x}_{t-1}, \mathbf{x}_t)}{\text{count}(\mathbf{x}_{t-n+1}, \dots, \mathbf{x}_{t-1})} \end{aligned}$$

In *bi-gram* LM

$P(\text{I like cats as they look cute}) = P(\text{I}) P(\text{like} | \text{I}) P(\text{cats} | \text{like}) P(\text{as} | \text{cats}) P(\text{they} | \text{because}) P(\text{look} | \text{they}) P(\text{cute} | \text{look})$

Discuss:

Do you want to have a large  $n$  or a small  $n$  in a  $n$ -gram model?

What is special about this sentence by Noam Chomsky:  
**“Colorless green ideas sleep furiously.”**

## $n$ -Gram Language Model: issue

$n$ -Gram: chunk of  $n$  consecutive words

**Uni-gram:** “I” “like” “cats” “as” “they” “look” “cute”

**Bi-gram:** “I like” “like cats” “cats as” “as they” ...

**Tri-gram:** “I like cats” “like cats as” “cats as they” ...

**Count** the frequency of each  $n$ -grams and predict next word!

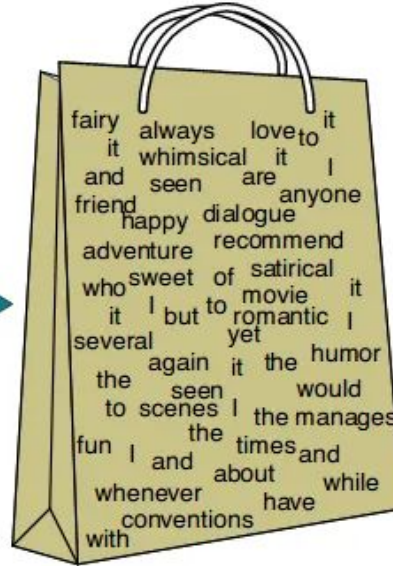
**Assume** each word only depends on previous  $n - 1$  words.

$$\begin{aligned} P(\mathbf{x}_t | \mathbf{x}_1, \dots, \mathbf{x}_{t-1}) &= P(\mathbf{x}_t | \mathbf{x}_{t-n+1}, \dots, \mathbf{x}_{t-1}) \\ &= \frac{\text{count}(\mathbf{x}_{t-n+1}, \dots, \mathbf{x}_{t-1}, \mathbf{x}_t)}{\text{count}(\mathbf{x}_{t-n+1}, \dots, \mathbf{x}_{t-1})} \end{aligned}$$

**Increasing  $n$  provides contextual information, but exponentially increases the size of the counting table!**

# Bag of Words (to represent documents)

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!



it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1
...	...

## Drawbacks:

- High dimensionality
- No semantic information



# Document similarity?

document 1

**Obama**  
**speaks**  
to  
the  
**media**  
in  
**Illinois**

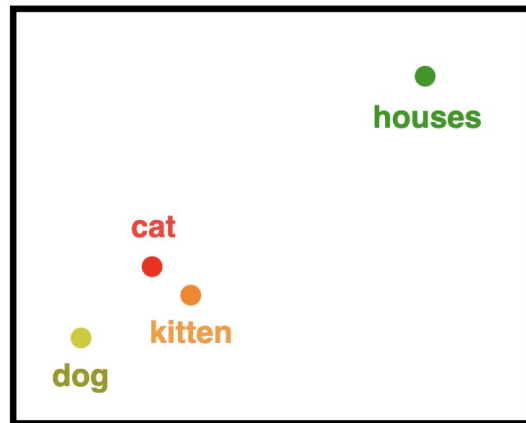
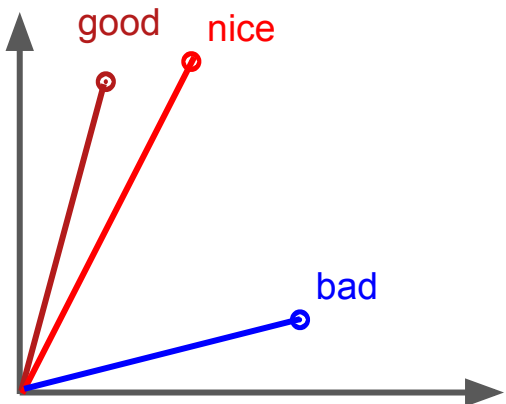
Documents have no words in common.  
How can we quantify that they convey  
similar meanings?  
(Assume B. Obama is president.)

document 2

The  
**President**  
**greet**s  
the  
**press**  
in  
**Chicago**

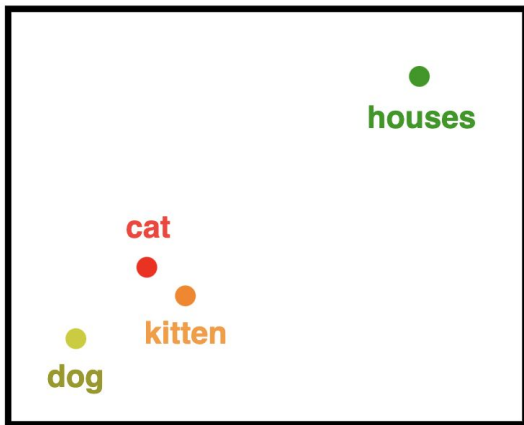
# Semantic similarity

- Motivation
  - Put words into vectors so we can measure the similarity between words
  - Use cosine similarity



# Why Do We Need Word Embeddings?

- Why Do We Need Word Embeddings?
  - Numerical Input
  - Shows Similarity and Distance

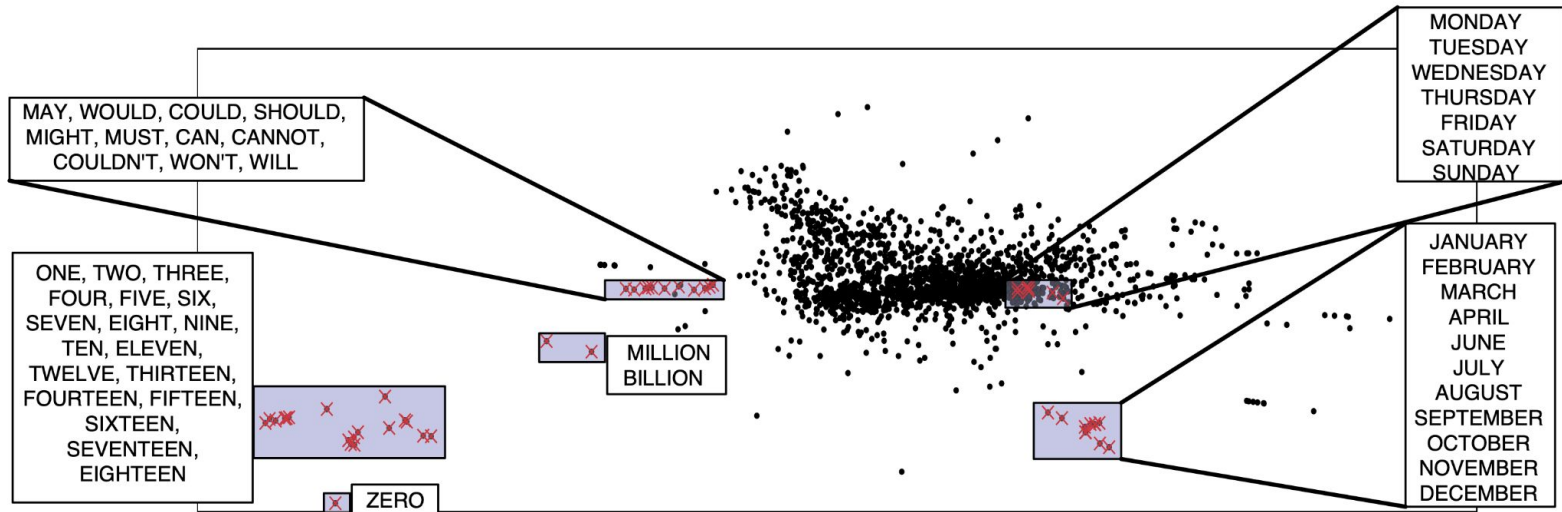


	<i>living being</i>	<i>feline</i>	<i>human</i>	<i>gender</i>	<i>royalty</i>	<i>verb</i>	<i>plural</i>
<b>cat</b>	0.6	0.9	0.1	0.4	-0.7	-0.3	-0.2
<b>kitten</b>	0.5	0.8	-0.1	0.2	-0.6	-0.5	-0.1
<b>dog</b>	0.7	-0.1	0.4	0.3	-0.4	-0.1	-0.3
<b>houses</b>	-0.8	-0.4	-0.5	0.1	-0.9	0.3	0.8
<b>man</b>	0.6	-0.2	0.8	0.9	-0.1	-0.9	-0.7
<b>woman</b>	0.7	0.3	0.9	-0.7	0.1	-0.5	-0.4
<b>king</b>	0.5	-0.4	0.7	0.8	0.9	-0.7	-0.6
<b>queen</b>	0.8	-0.1	0.8	-0.9	0.8	-0.5	-0.9

embedding using features of words

# What are word embeddings

- What are Word Embeddings?
  - vector representations of words that capture semantic relationships
  - Latent Semantic Analysis / Indexing [S. Deerwester et al 1988]



# Demo

Visualize: <https://projector.tensorflow.org/>

Explore: [http://epsilon-it.utu.fi/wv\\_demo/](http://epsilon-it.utu.fi/wv_demo/)

# What is **Underberg**?

Suppose you see these sentences:

- I love drinking **Underberg** after a meal.
- I find **Underberg** is quite strong.
- A few bottles of **Underberg** make me very drunk.

## Word2Vec:

- We want vectors for words so that the context of a word can suggest the vector of this word, and vice versa
- Idea: **Similar words appear in similar contexts**

---

## Efficient Estimation of Word Representations in Vector Space

---

**Tomas Mikolov**

Google Inc., Mountain View, CA  
tmikolov@google.com

**Kai Chen**

Google Inc., Mountain View, CA  
kaichen@google.com

**Greg Corrado**

Google Inc., Mountain View, CA  
gcorrado@google.com

**Jeffrey Dean**

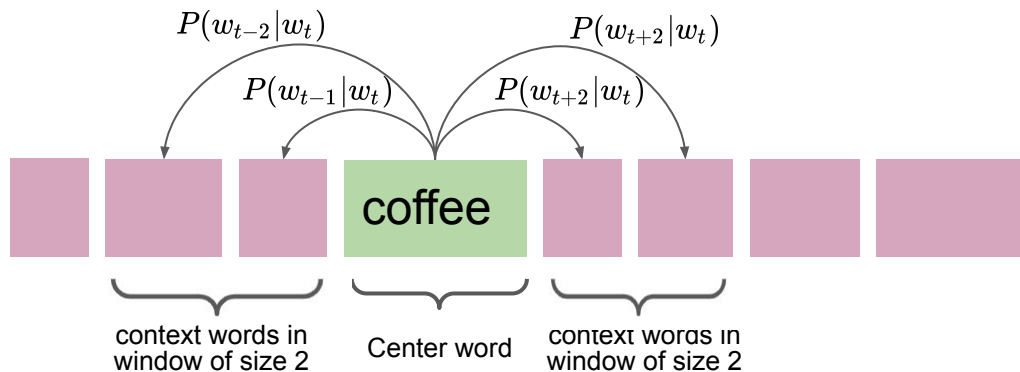
Google Inc., Mountain View, CA  
jeff@google.com

### Abstract

We propose two novel model architectures for computing continuous vector representations of words from very large data sets. The quality of these representations is measured in a word similarity task, and the results are compared to the previously best performing techniques based on different types of neural networks. We observe large improvements in accuracy at much lower computational cost, i.e. it takes less than a day to learn high quality word vectors from a 1.6 billion words data set. Furthermore, we show that these vectors provide state-of-the-art performance on our test set for measuring syntactic and semantic word similarities.

# Word2Vec - Training

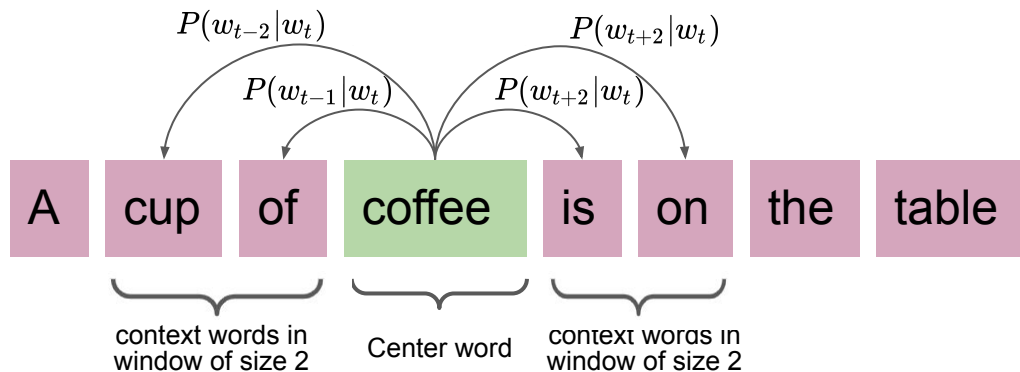
SkipGram - Predict context from target





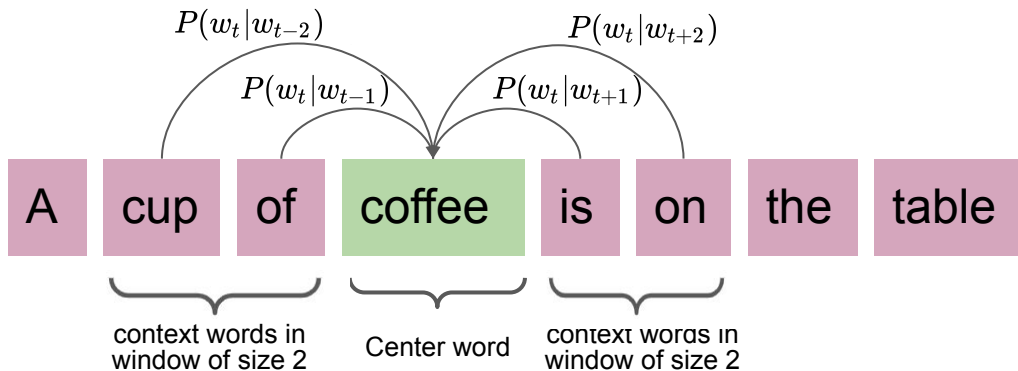
# Word2Vec - Training

SkipGram - Predict context from target



# Word2Vec - Training

Continuous Bag of Words (CBOW) - predict target from context



## SkipGram - Training samples

A cup of coffee is on the table



(coffee, cup)

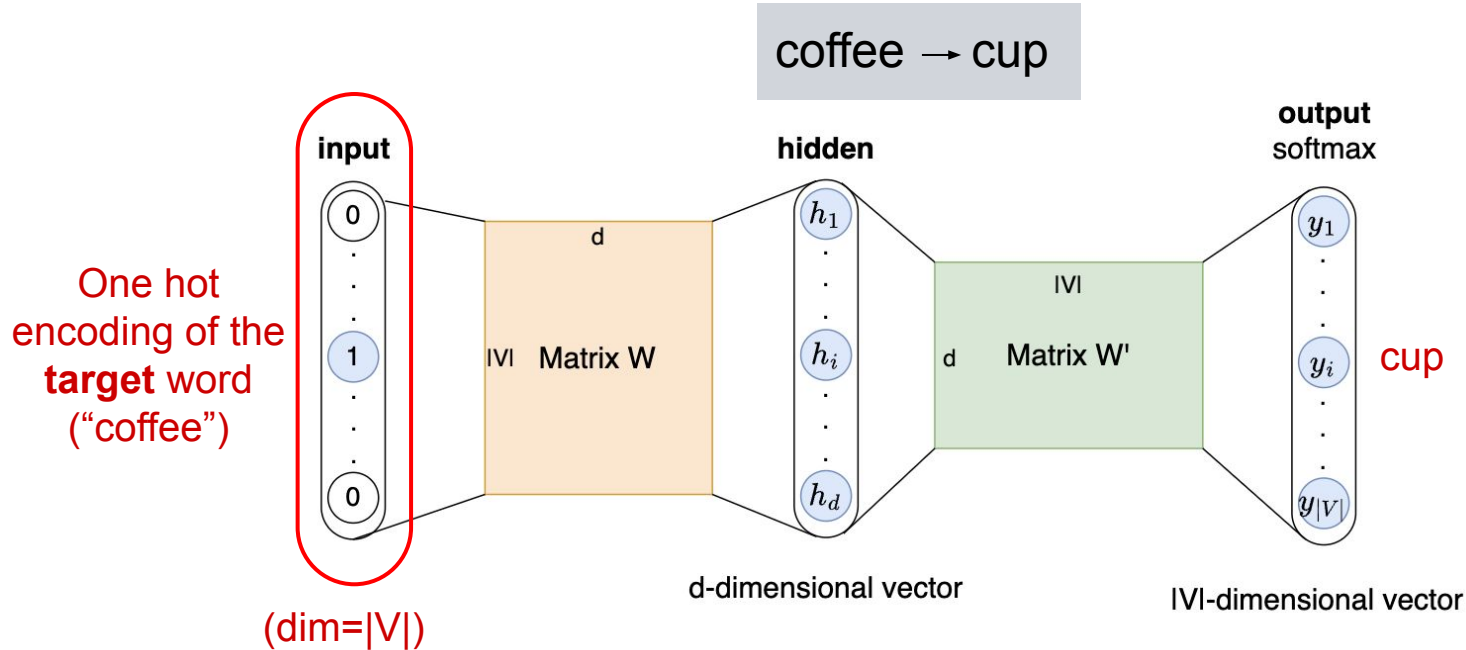
(coffee, of)

(coffee, is)

(coffee, on)

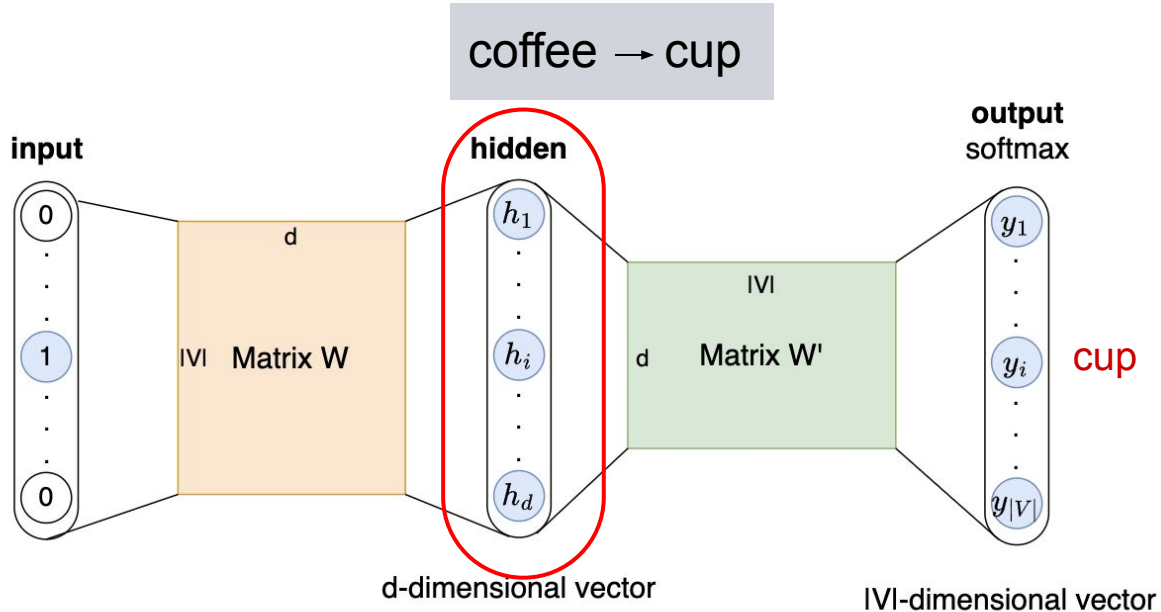
# Word2Vec Architecture - SkipGram

Predict context word from target word!



# Word2Vec Architecture - SkipGram

Predict every target word from each context word!



# Word2Vec Architecture - SkipGram

What is the output of multiplying the one-hot vector  $[0, 1, 0, 0, 0, 0, 0]$  with  $\mathbf{W}$ ?

For every target word  $\mathbf{t}$  we have one vector  $\mathbf{W}_t$

$d = \text{embedding size}$

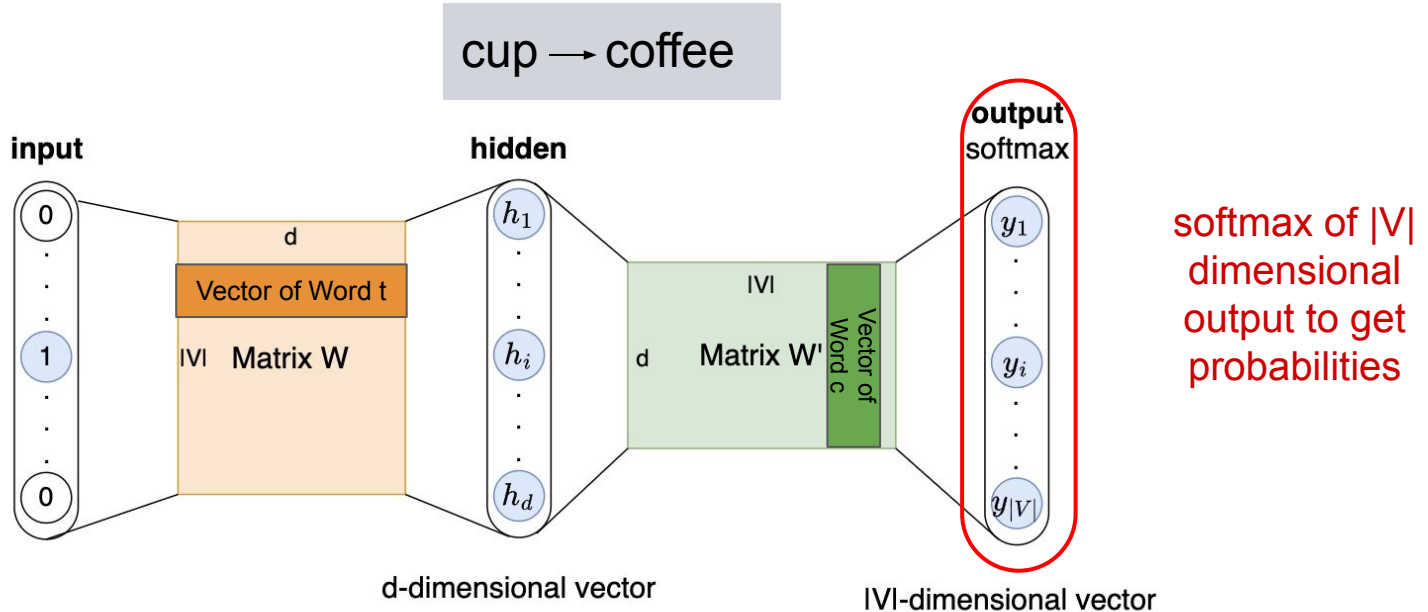
0.1	0.5	0.3	-0.9	0.4
0.8	-0.4	0.7	0.3	-0.1
0.4	0.3	-0.9	-0.2	0.7
-0.5	-0.1	0.7	0.8	0.6
-0.9	0.6	-0.5	0.6	-0.2
0.2	0.8	0.6	0.3	0.6
0.2	0.2	-0.9	-0.5	0.3

$|V| = \text{vocab size}$

**Matrix  $\mathbf{W}$  (learnt from training)**

# Word2Vec Architecture - SkipGram

Predict every target word from each context word!



## Looking closer...

- We observe that every row of the  $W$  matrix corresponds to a target word and every column of the  $W'$  matrix corresponds to a context word.
- We compute the probability of a target-context pair with the **soft-max** as:

$$p(w_c | w_t) = \frac{\exp(W_t W'_c{}^T)}{\sum_{i=1}^{|V|} \exp(W_t W'_i{}^T)}$$

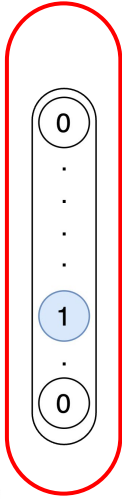
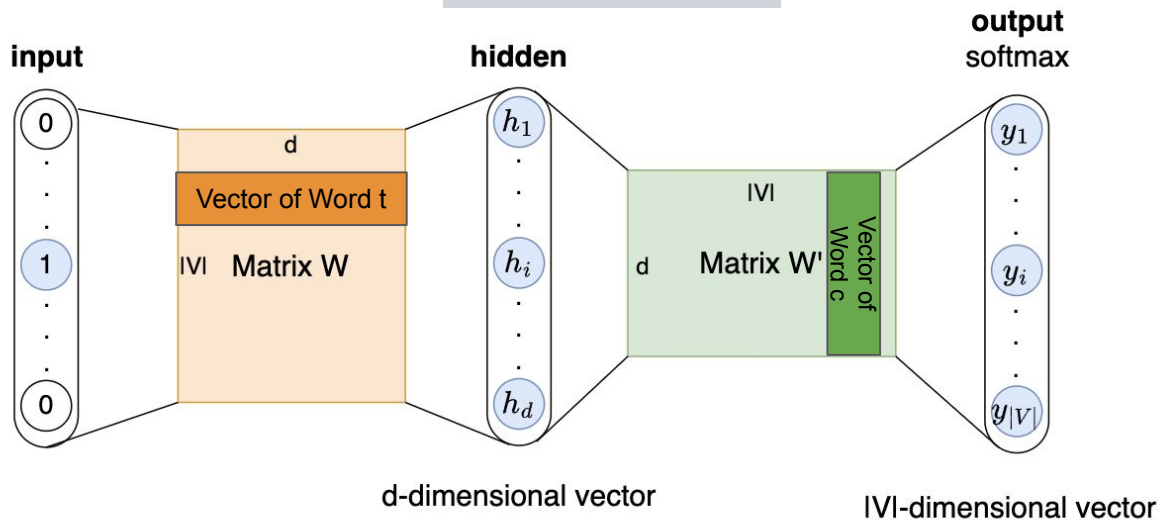
Here,  $W_t$  indicates the t-th row of matrix  $W$ .



# Word2Vec Architecture - SkipGram

Predict context word from target word!

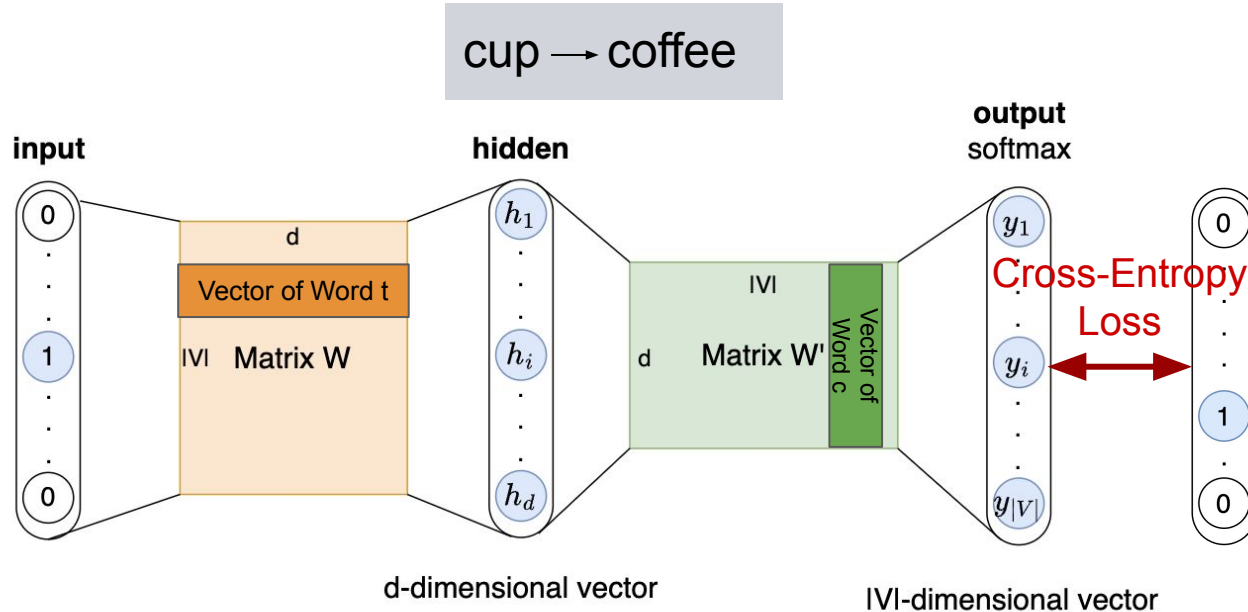
cup → coffee



One hot encoding of the context word (dim= $|V|$ )

# Word2Vec Architecture - SkipGram

Predict context word from target word!



# Cross Entropy

- Cross Entropy: lower cross entropy indicates high similarity between two distributions

$$\mathcal{L}_\theta = - \sum_{i=1}^{|V|} y_i \log p(w_i | w_t) = -\log p(w_c | w_t)$$

- So the loss function is:

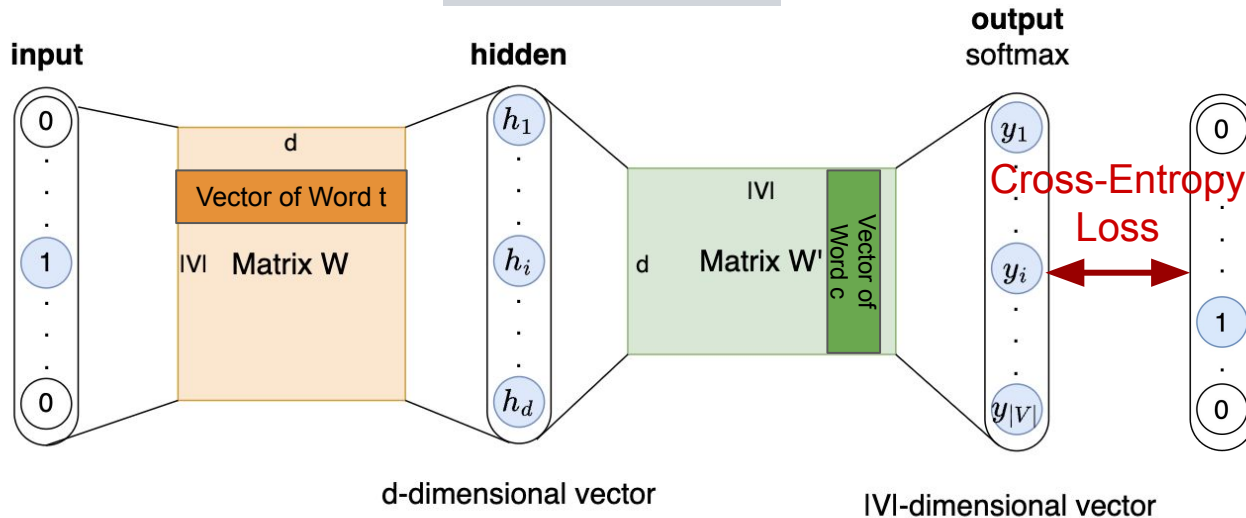
**In the end, we only keep  $W$  and discard  $W'$ .**

$$\mathcal{L}_\theta = -\log \frac{\exp(W_t W'_c{}^T)}{\sum_{i=1}^{|V|} \exp(W_t W'_i{}^T)}$$

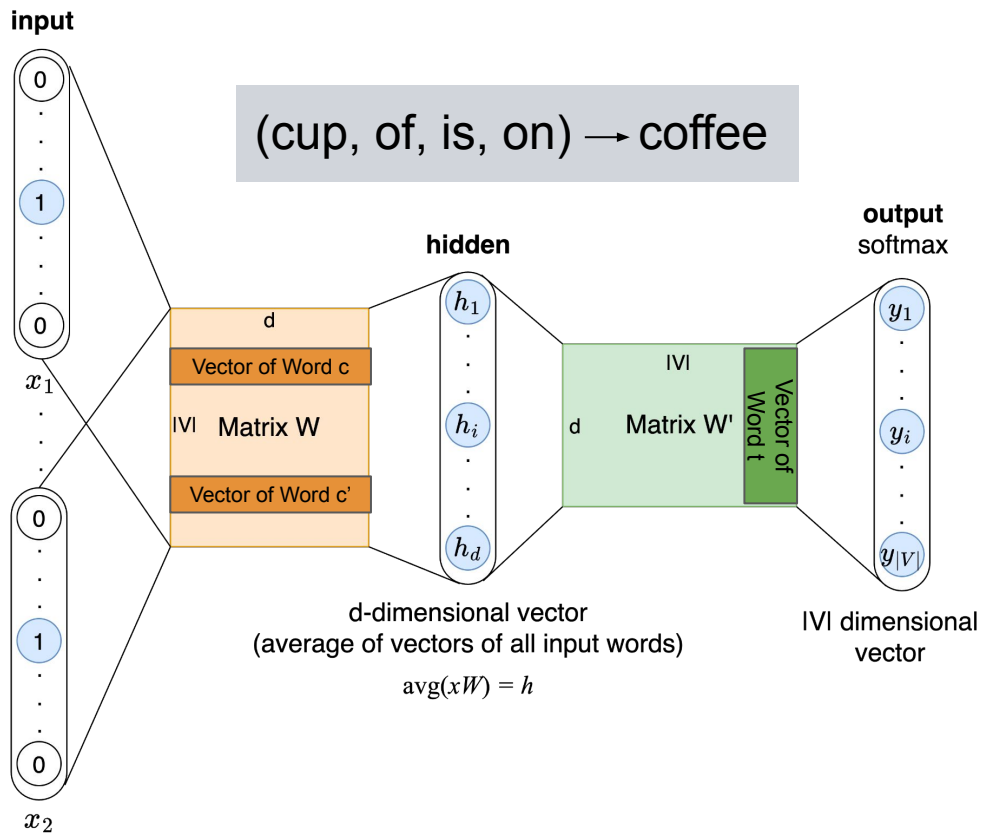
# Word2Vec Architecture - SkipGram

Predict context word from target word!

cup → coffee

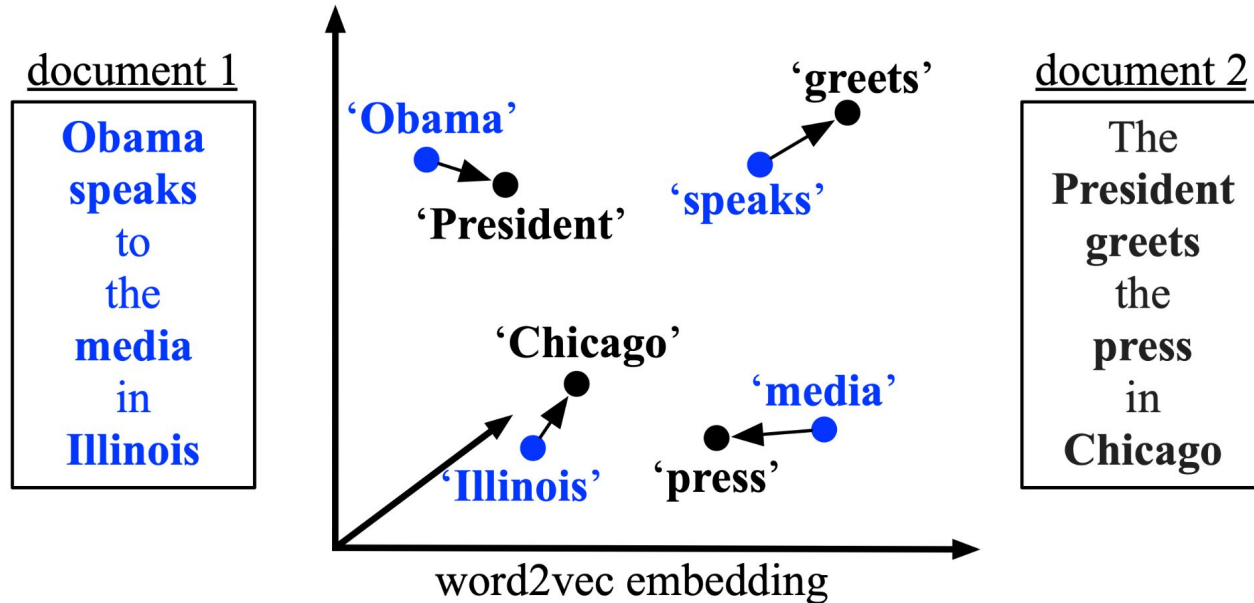


# Word2Vec Architecture - CBOW (continuous bag of words)



# Word Mover's Distance [Kusner et al., 2015]

Measure similarity between documents as the minimum travel distance to match all words from one document to those of the other in word2vec space.



## X 2 vec

- Generate vector representations (embeddings) for various data types
- Examples:
  - Word2Vec
  - Doc2Vec
  - Node2Vec
  - Item2Vec
  - Sent2Vec



# Demo

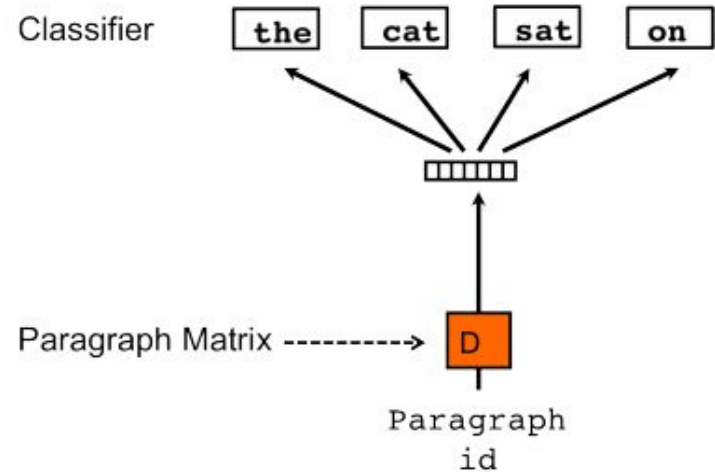
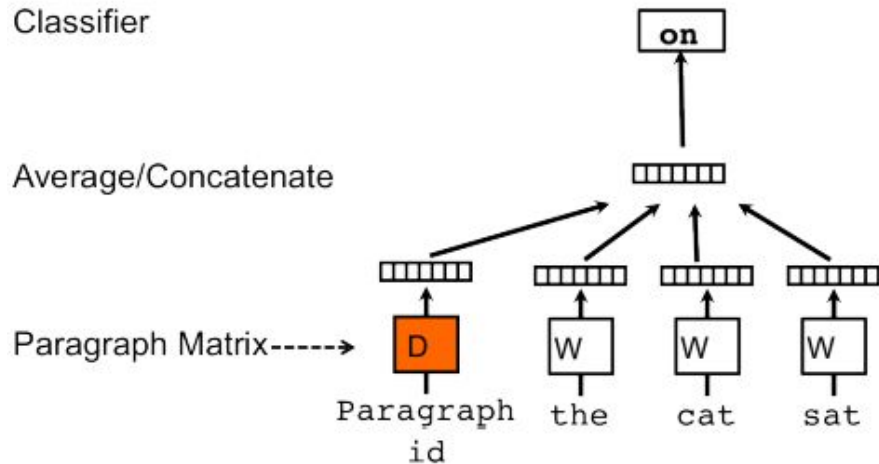
Visualize: <https://projector.tensorflow.org/>

Explore: [http://epsilon-it.utu.fi/wv\\_demo/](http://epsilon-it.utu.fi/wv_demo/)

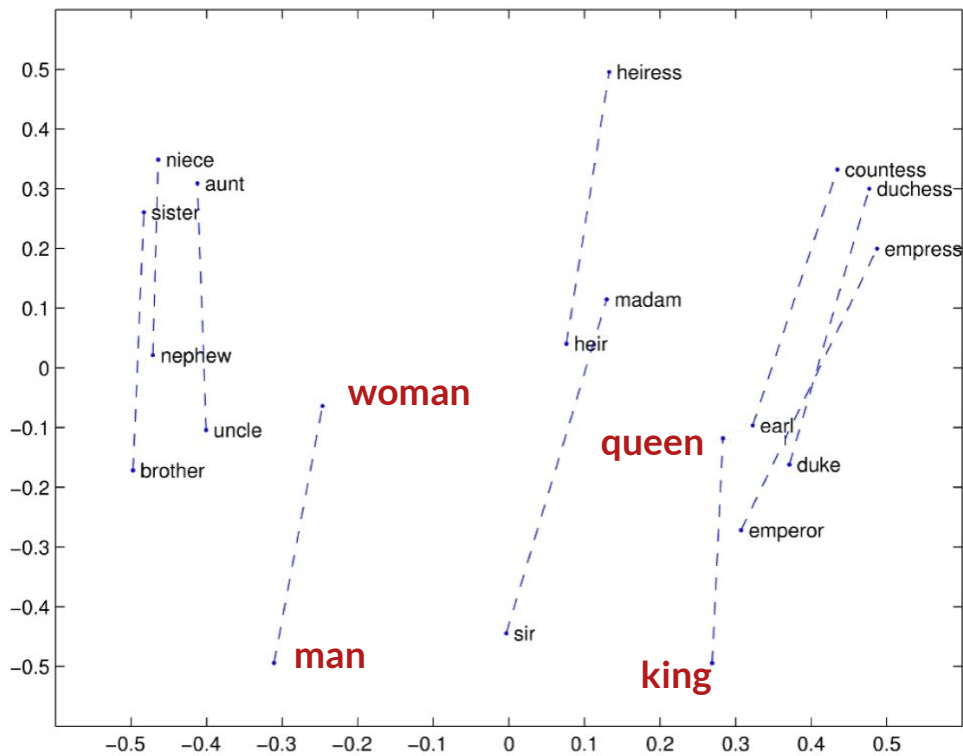


# Doc2Vec

- A vector to represent a paragraph, regardless of length
  - embeddings for paragraph and words
  - Applications: Document classification, sentiment analysis, recommendation systems, and information retrieval



# In vector space...



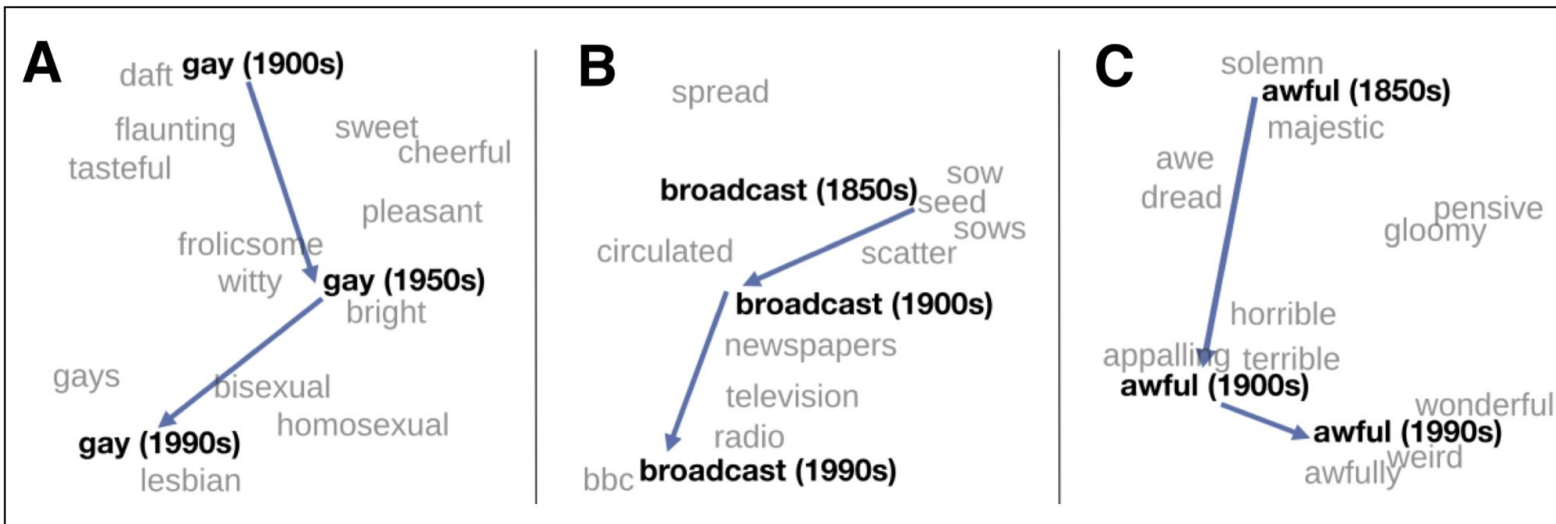
# Word embeddings capture societal biases

<b>Extreme <i>she</i></b>	<b>Extreme <i>he</i></b>		<b>Gender stereotype <i>she-he</i> analogies</b>	
1. homemaker	1. maestro	sewing-carpentry	registered nurse-physician	housewife-shopkeeper
2. nurse	2. skipper	nurse-surgeon	interior designer-architect	softball-baseball
3. receptionist	3. protege	blond-burly	feminism-conservatism	cosmetics-pharmaceuticals
4. librarian	4. philosopher	giggle-chuckle	vocalist-guitarist	petite-lanky
5. socialite	5. captain	sassy-snappy	diva-superstar	charming-affable
6. hairdresser	6. architect	volleyball-football	cupcakes-pizzas	lovely-brilliant
7. nanny	7. financier			
8. bookkeeper	8. warrior	queen-king	<b>Gender appropriate <i>she-he</i> analogies</b>	
9. stylist	9. broadcaster	waitress-waiter	sister-brother	mother-father
10. housekeeper	10. magician		ovarian cancer-prostate cancer	convent-monastery

Figure 1: **Left** The most extreme occupations as projected on to the *she*–*he* gender direction on w2vNEWS. Occupations such as *businesswoman*, where gender is suggested by the orthography, were excluded. **Right** Automatically generated analogies for the pair *she-he* using the procedure described in text. Each automatically generated analogy is evaluated by 10 crowd-workers to whether or not it reflects gender stereotype.

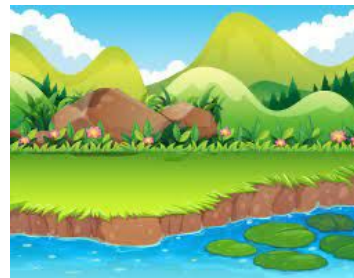
# Word embeddings are time-dependent (why?)

- Semantic similarity of words depends on *time*.

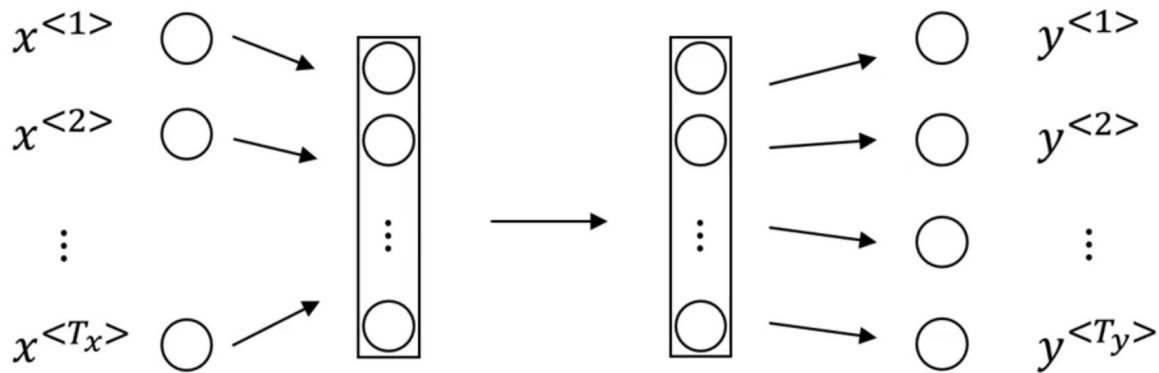


## Problems with word2vec

- Words with multiple meanings only have one representation
  - eg. **bank** of river or **bank** of money
  - Need contextual information
- Limited Context
  - only trained on words within the context window



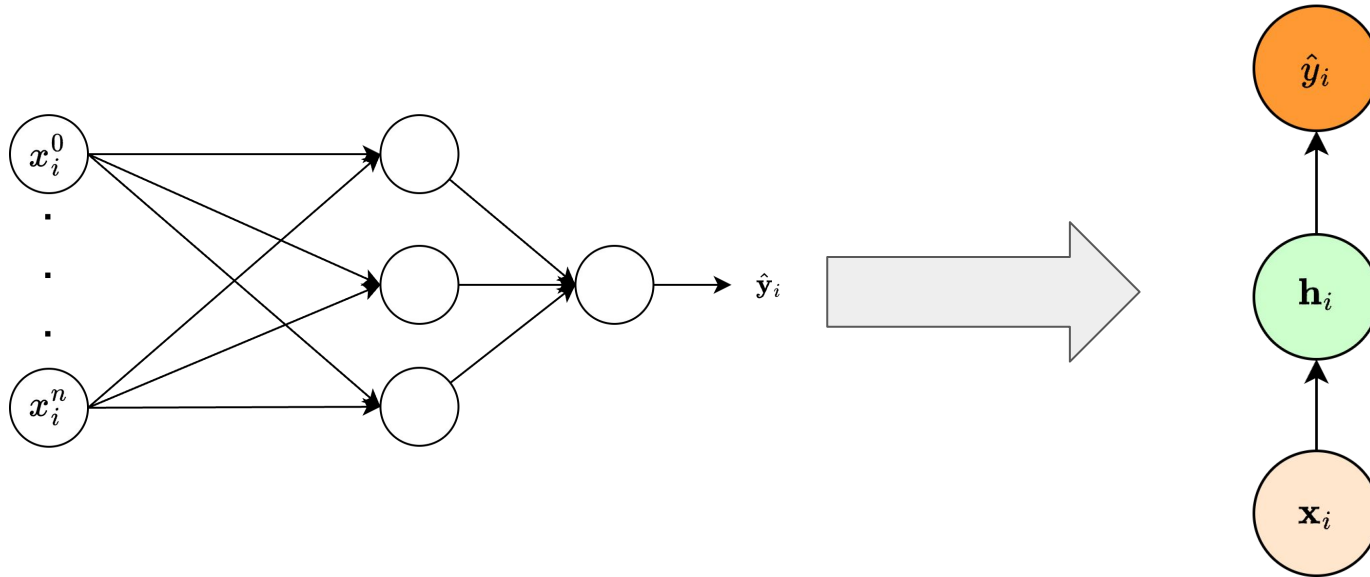
## How to use word vectors with neural networks?



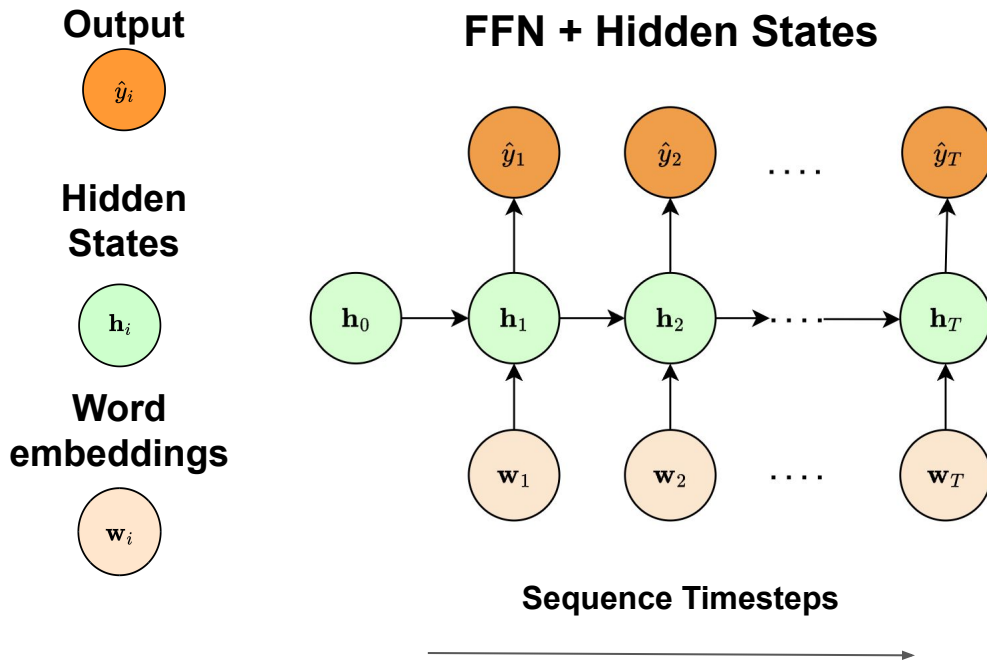
- Inputs and outputs don't have fixed lengths
- Features are not shared

# Let's simplify!

What if we have a single word and a single output?

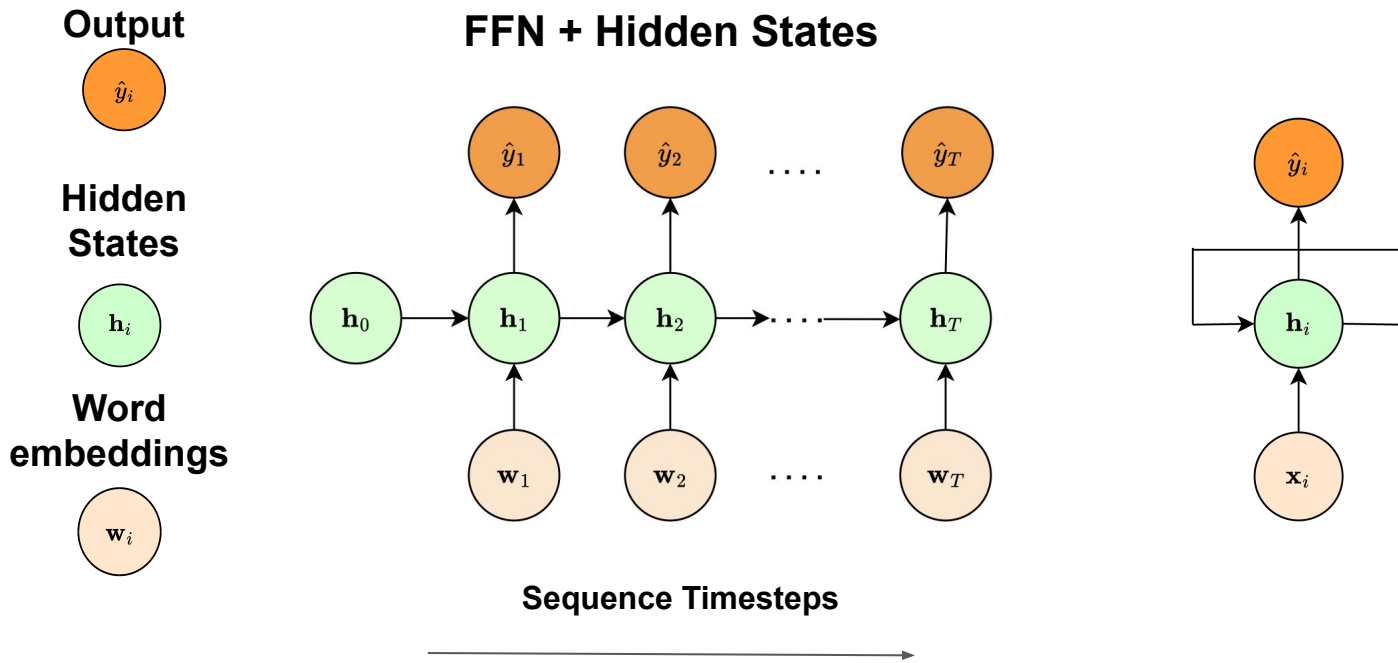


# Recurrent neural network (RNN)

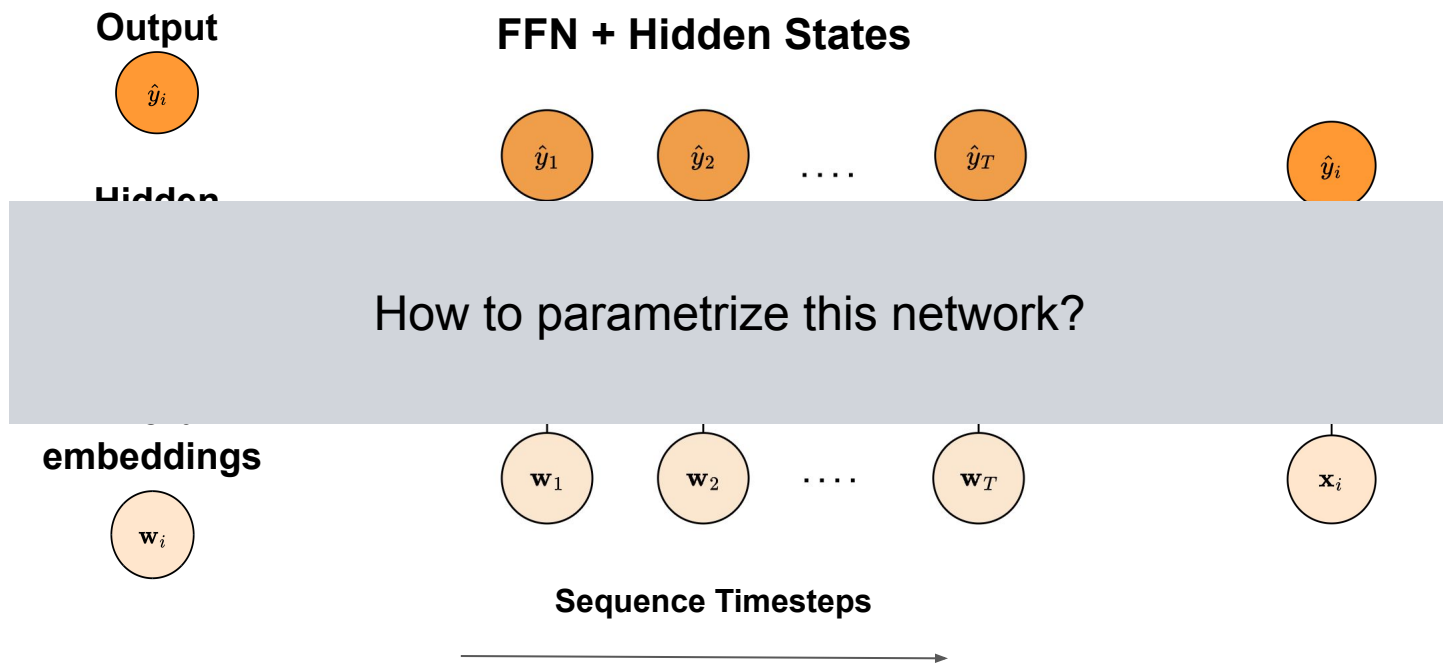




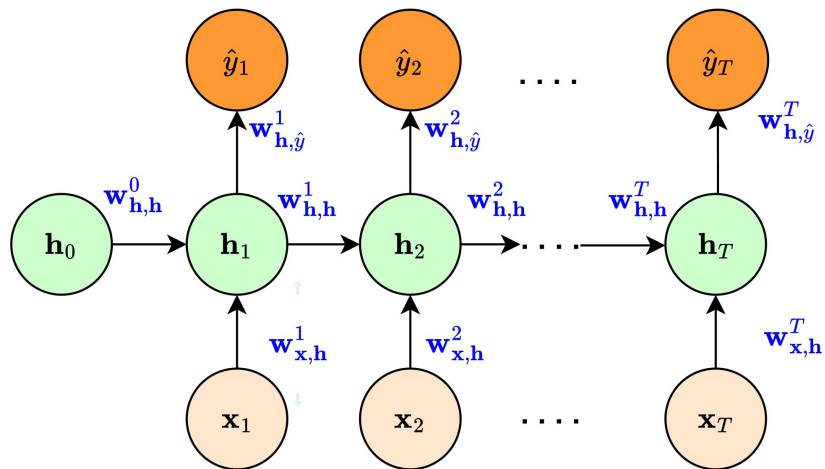
# Recurrent neural network (RNN)



# Recurrent neural network (RNN)



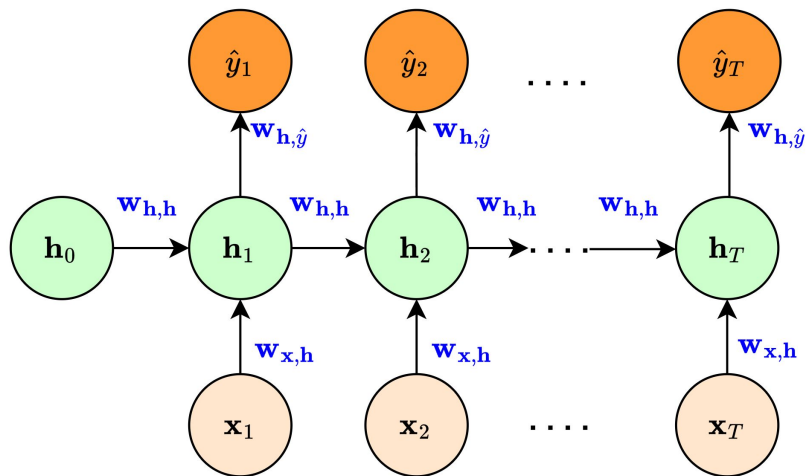
## Parameterize RNN



- Too many parameters if we have a long sequence!
- Longer sequence parameters will not receive many updates
- What if sequence lengths vary?

# RNN w/ parameter-sharing

Simple fix: use **the same parameters** across different timesteps.



## Recap

- **N-gram models**
- **Bag-of-words representations**
- **Word2Vec**
  - CBOW: use context to predict target word
  - SkipGram: use target word to predict context
- **RNN**
  - Has an internal state (memory)
  - Can handle arbitrary sequences of inputs
  - Trained with back propagation through time

## Image credits:

<https://web.stanford.edu/~jurafsky/slp3/6.pdf>

<https://lilianweng.github.io/posts/2017-10-15-word-embedding/>