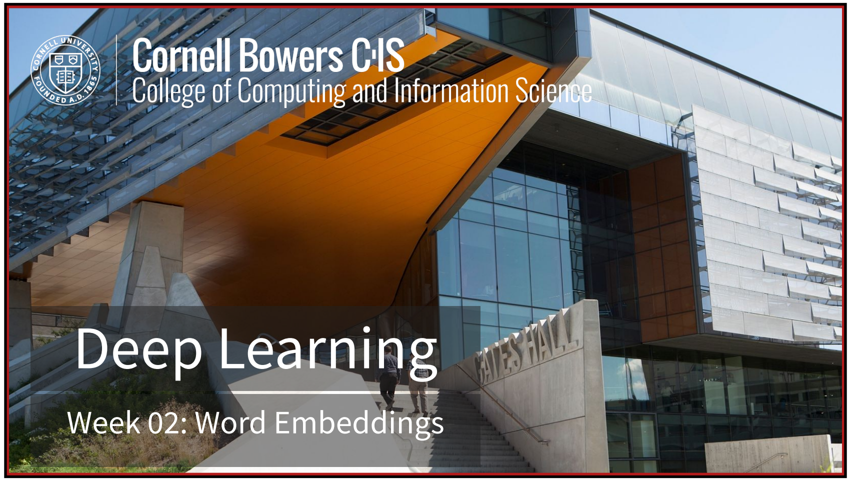


Deep Learning

Week 02: Word Embeddings

Cornell Bowers C·IS

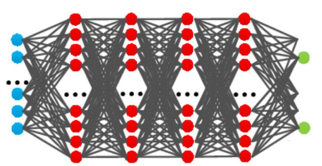
Logistics

- HW1 is due on Thursday
- 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

Cornell Bowers C·IS

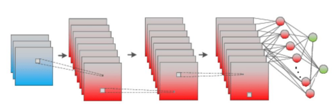
MLPs:

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



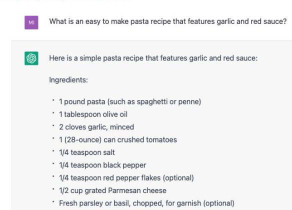
CNNs:

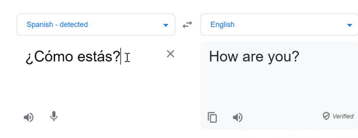
- Convolutional layers with filters
- Designed specifically for structured input like images
- Inherently translation invariant due to shared weights
- Requires fewer parameters

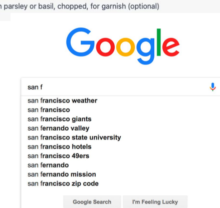


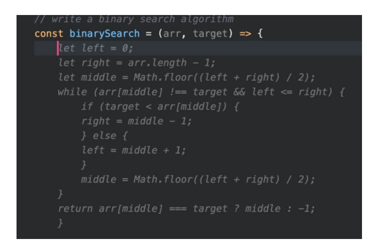
https://www.researchgate.net/figure/The-architecture-of-MLP-and-CNN-MLP-is-consisted-of-fullyconnected-FC-layers-and-CNN_fig2_334489445

Cornell Bowers C·IS









Cornell Bowers C·IS

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

Cornell Bowers C·IS

How to handle text data?

Cornell Bowers C·IS

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(I like cats because they look cute) = P(I) P(like | I) P(cats | I like) P(as | I like cats) P(they | I like cats because) P(look | I like cats because they) P(cute | I like cats because they look)

Predict the next word given current text!

Cornell Bowers C·IS

n-Gram Language Model

n-Gram: chunk of *n* consecutive words

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

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" …

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

$$P(\mathbf{x}_t|\mathbf{x}_1,\dots,\mathbf{x}_{t-1}) = P(\mathbf{x}_t|\mathbf{x}_{t-n+1},\dots,\mathbf{x}_{t-1}) = \frac{\mathrm{count}(\mathbf{x}_{t-n+1},\dots,\mathbf{x}_{t-1},\mathbf{x}_t)}{\mathrm{count}(\mathbf{x}_{t-n+1},\dots,\mathbf{x}_{t-1})}$$

In *bi-gram* LM

P(I like cats as they look cute) = P(I) P(like | I) P(cats | like) P(as | cats) P(they | because) P(look | they) P(cute | look)

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

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” …

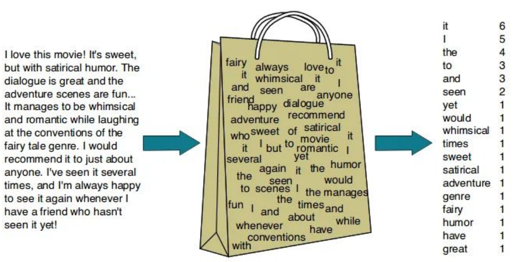
Count the frequency of each *n*-grams and predict next word!

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

$$P(\mathbf{x}_t|\mathbf{x}_1,\ldots,\mathbf{x}_{t-1}) = P(\mathbf{x}_t|\mathbf{x}_{t-n+1},\ldots,\mathbf{x}_{t-1}) = \frac{\mathrm{count}(\mathbf{x}_{t-n+1},\ldots,\mathbf{x}_{t-1},\mathbf{x}_t)}{\mathrm{count}(\mathbf{x}_{t-n+1},\ldots,\mathbf{x}_{t-1})}$$

Increase *n* provides contextual information, but exponentially increase the size of the counting table!

Bag of Words



https://koushik1102.medium.com/nlp-bag-of-words-and-tf-idf-explained-fd1f49dce7c4

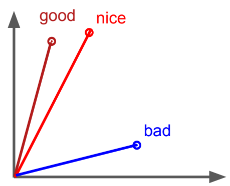
What are word embeddings

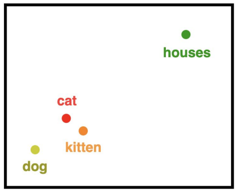
- What are Word Embeddings?
  - vector representations of words that capture semantic relationships

Cornell Bowers CIS

Semantic similarity

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

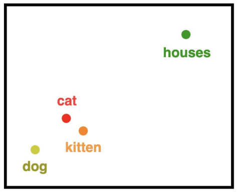




Cornell Bowers CIS

Why Do We Need Word Embeddings?

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



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

embedding using features of words

Cornell Bowers CIS

What does ong choi mean?

Suppose you see these sentences:

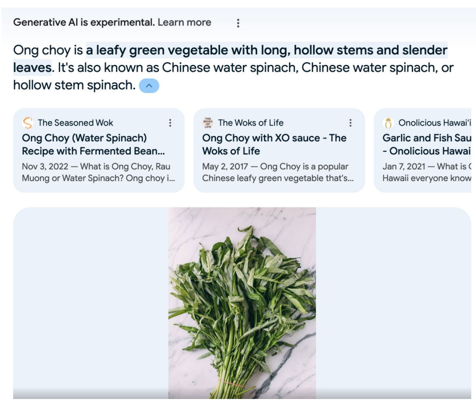
- Ong choi is delicious sautéed with garlic.
- Ong choi is superb over rice
- Ong choi leaves with salty sauces

And you've also seen these:

- ...spinach sautéed with garlic over rice
- Chard stems and leaves are delicious
- Collard greens and other salty leafy greens

https://web.stanford.edu/~jurafsky/slp3/

Cornell Bowers CIS



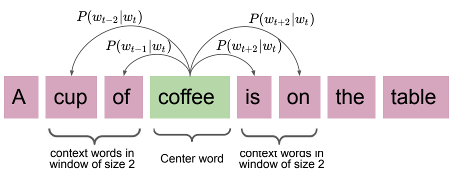
Word2Vec

- 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**

A cup of **coffee** is on the table.
Coffee helps me focus.
Espresso is my favorite type of **coffee**.

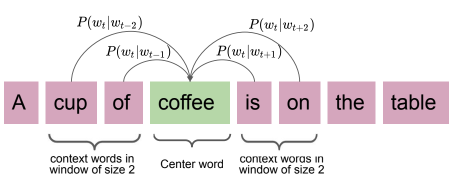
Word2Vec - Training

SkipGram - Predict context from target



Word2Vec - Training

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



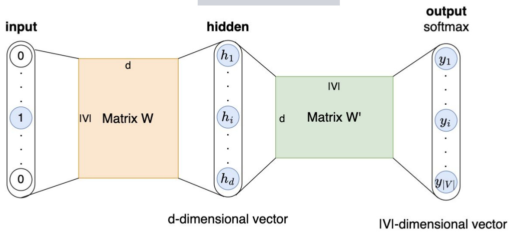
SkipGram - Training samples

(coffee, cup)
(coffee, of)
(coffee, is)
(coffee, on)

Word2Vec Architecture - SkipGram

Predict every target word from each context word!

coffee → cup



Discuss: Word2Vec Architecture - CBOW

What is the output of multiplying the one-hot vector [0,1,0,0,0,0,0] with **W**?

d = embedding size

IVI = vocab size

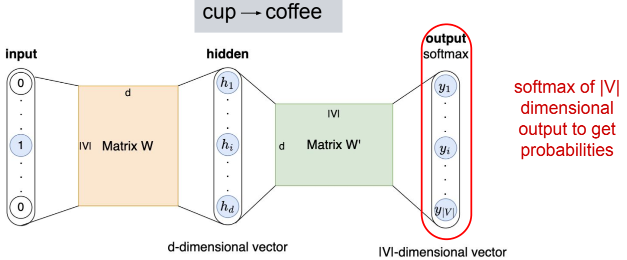
0.1	0.5	0.3	-0.9	0.4
0.8	-0.4	0.7	0.3	-0.1
0.4	0.3	-0.9	-0.2	0.7
-0.5	-0.1	0.7	0.8	0.6
-0.9	0.6	-0.5	0.6	-0.2
0.2	0.8	0.6	0.3	0.6
0.2	0.2	-0.9	-0.5	0.3

Matrix W (learnt from training)

Word2Vec Architecture - SkipGram

Predict every target word from each context word!

cup → coffee



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 as:

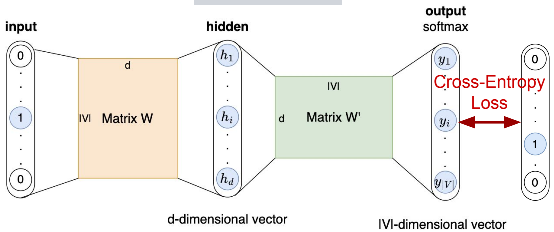
$$p(w_c|w_t) = \frac{exp(W_t {W'}_c^T)}{\sum_{i=1}^{|V|} exp(W_t {W'}_i^T)}$$

Cornell Bowers CIS

Word2Vec Architecture - SkipGram

Predict every target word from each context word!

cup → coffee



Cornell Bowers CIS

Cross Entropy

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

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

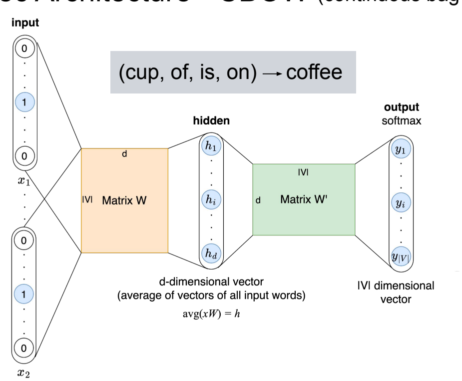
- So the loss function is:

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

Cornell Bowers CIS

Word2Vec Architecture - CBOW (continuous bag of words)

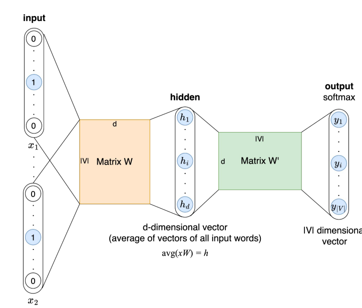
(cup, of, is, on) → coffee



Cornell Bowers CIS

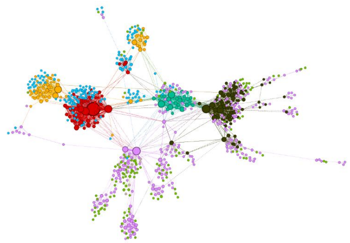
Word2Vec Architecture - CBOW (continuous bag of words)

Where do we get the word embeddings from in this version of Word2Vec (CBOW)?



X 2 vec

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



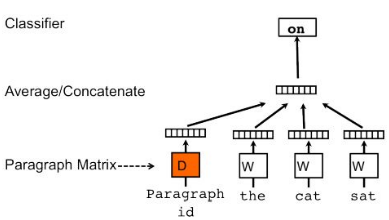
Demo

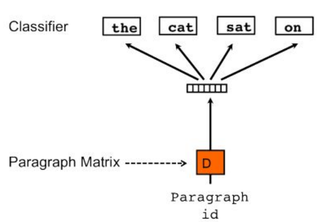
Visualize: https://projector.tensorflow.org/

Explore: http://epsilon-it.utu.fi/wv_demo/

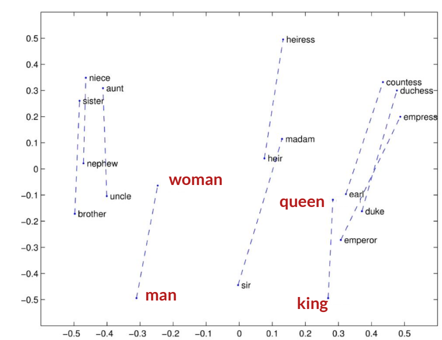
Doc2Vec

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





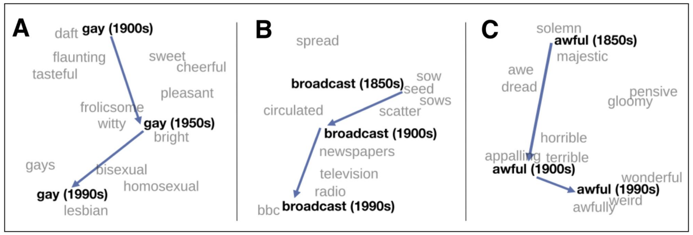
In vector space…



Cornell Bowers C·IS

Word embeddings are time-dependent (why?)

- Semantic similarity of words depends on *time.*



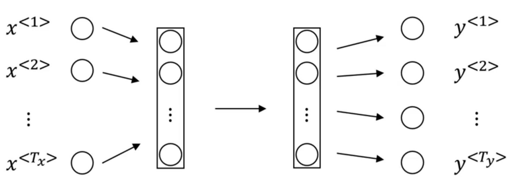
Cornell Bowers C·IS

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

Cornell Bowers C·IS

How to use word vectors with neural networks?

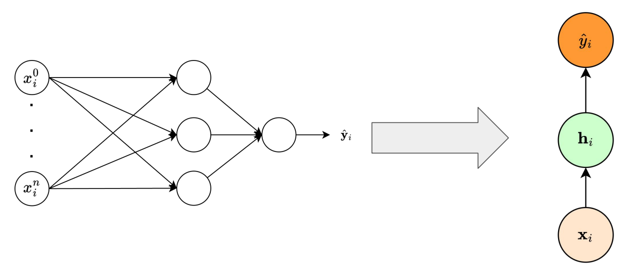


- Inputs and outputs don’t have fixed lengths
- Features are not shared

Cornell Bowers C·IS

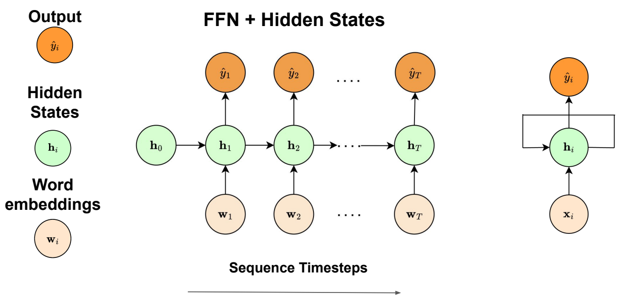
Let’s simplify!

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



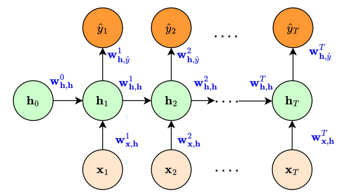
Cornell Bowers C·IS

Recurrent neural network (RNN)



Cornell Bowers C·IS

Parameterize RNN

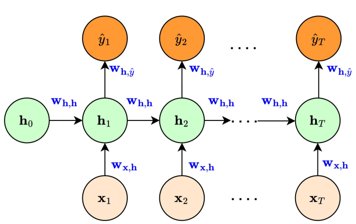


- Too many parameters if we have a long sequence!
- Longer sequence parameters will not receive many updates

Cornell Bowers C·IS

RNN w/ parameter-sharing

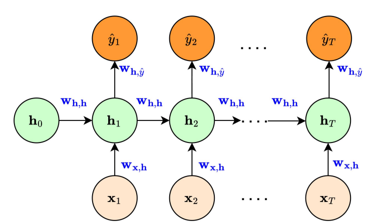
Simple fix: use **the same parameters** across different timesteps.



Cornell Bowers C·IS

Discuss: RNN w/ parameter-sharing

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

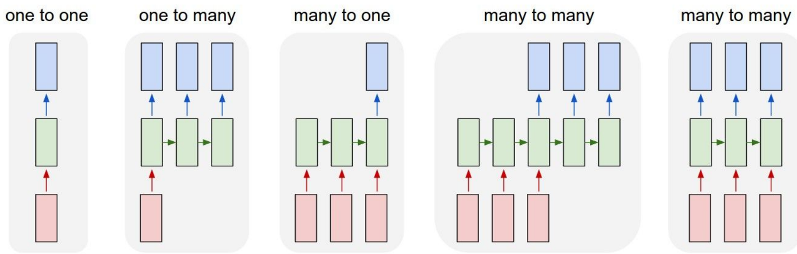


A non-linearity is applied to the output of the recurrent unit before it is passed to the next time step or to the output layer of the network.

Write a closed-form solution for **$\mathbf{h}_i$** and $\hat{y}_i$

Cornell Bowers CIS

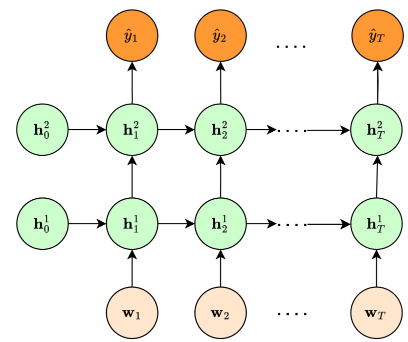
Types of RNNs



https://www.analyticsvidhya.com/blog/2021/06/time-series-analysis-recurrence-neural-network-in-python/

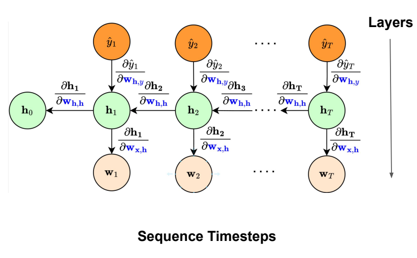
Cornell Bowers CIS

Stacking RNN Layers



Cornell Bowers CIS

Backpropagation through the Time (BPTT)

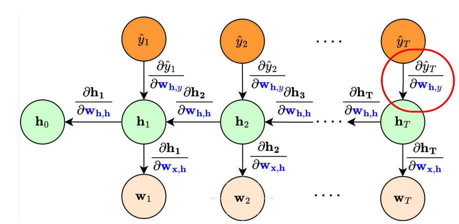


- Unfold a recurrent neural network in time
- Gradients are accumulated across all time steps by applying the chain rule
- Propagate gradients backwards through time steps

Cornell Bowers CIS

Backpropagation through the Time (BPTT)

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}_{\mathbf{h},\hat{y}}} = \frac{\partial \mathcal{L}}{\partial \hat{y}_T} \cdot \frac{\partial \hat{y}_T}{\partial \mathbf{w}_{\mathbf{h},\hat{y}}}$$



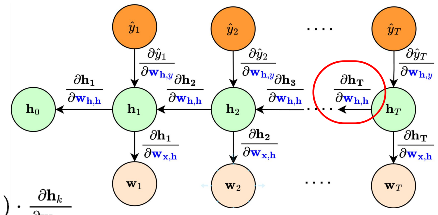
Cornell Bowers C·IS

Backpropagation through the Time (BPTT)

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w_{h,h}}} = \frac{\partial \mathcal{L}}{\partial \hat{y}_T} \cdot \frac{\partial \hat{y}_T}{\partial \mathbf{h}_T} \cdot \frac{\partial \mathbf{h}_T}{\partial \mathbf{h}_{T-1}} \cdot \frac{\partial \mathbf{h}_{T-1}}{\partial \mathbf{w}_{h,h}}$$

$$= \sum_{k=1}^{T} \frac{\partial \mathcal{L}}{\partial \hat{y}_T} \cdot \frac{\partial \hat{y}_T}{\partial \mathbf{h}_T} \cdot \frac{\partial \mathbf{h}_T}{\partial \mathbf{h}_k} \cdot \frac{\partial \mathbf{h}_k}{\partial \mathbf{w}_{h,h}}$$

$$= \sum_{k=1}^{T} \frac{\partial \mathcal{L}}{\partial \hat{y}_T} \cdot \frac{\partial \hat{y}_T}{\partial \mathbf{h}_T} \cdot \left(\prod_{j=k}^{T-1} \frac{\partial \mathbf{h}_{j+1}}{\partial \mathbf{h}_j}\right) \cdot \frac{\partial \mathbf{h}_k}{\partial \mathbf{w}_{h,h}}$$



Cornell Bowers C·IS

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

Cornell Bowers C·IS

Image credits:

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

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