INFO-I590 Fundamentals and Applications of LLMs

Language Modeling

Jisun An

Many slides from Graham Neubig
and Dan Jurafsky

What's missing in BOW?

- Handling of *conjugated* or *compound words*
  - I **love** this movie → I **loved** this movie
- Hanging of word similarity
  - I **love** this movie → I **adore** this movie
- Handling of combination features
  - I **love** this movie → I **don't love** this movie
  - I **hate** this movie → I **don't hate** this movie
- Handing of sentence structure
  - It has an interesting story, **but** is boring overall

Subword Models

Word Embeddings

Neural Networks

Sequence Models

source: CMU CS 11-711, Advanced NLP

Language Modeling

Probabilistic Language Models

- **Goal**: **assign a probability to a sentence**
- Why?
  - Machine Translation
    - P(**high** winds tonight) > P (**large** winds tonight)
  - Spell Correction
    - The office is about fifteen **minuets** from my house
      - P(about fifteen **minutes** from) > P(about fifteen **minuets** from)
  - Speech Recognition
    - P(I saw a van) >> P(eyes awe of an)
  - \+ Summarization, question-answering, etc, etc.

Probabilistic Language Models

- Goal: compute the probability of a sentence or sequence of words

$$P(X) = P(x_1, x_2, \ldots, x_n)$$

, where $X = (x_1, x_2, \ldots, x_n)$ is a sequence of words

- Related task: probability of an upcoming words

$$P(x_5 \mid x_1, x_2, x_3, x_4)$$

- A model that computes either of these:

$P(X)$ or $P(x_i \mid x_1, \ldots, x_{i-1})$ is called a **language model**.

How to compute P(X)

- How to compute this joint probability:

$$P(\text{its water is so transparent})$$

- Intuition: let's rely on the Chain Rule of Probability

Reminder: The Chain Rule

- Recall the definition of conditional probabilities

$$P(A \mid B) = \frac{P(A, B)}{P(B)}$$

$$P(A \mid B) \cdot P(B) = P(A, B)$$

$$P(A, B) = P(A \mid B) \cdot P(B)$$

- More variables:

$$P(A, B, C, D) = P(A) \cdot P(B \mid A) \cdot P(C \mid A, B) \cdot P(D \mid A, B, C)$$

- The Chain Rule in General:

$$P(X_1, X_2, \ldots, X_n) = P(X_1) \cdot P(X_2 \mid X_1) \cdot P(X_3 \mid X_1, X_2) \cdots P(X_n \mid X_1, X_2, \ldots, X_{n-1})$$

$$P(\text{its water is so transparent}) =$$
$$P(\text{its}) \cdot P(\text{water} \mid \text{its}) \cdot P(\text{is} \mid \text{its water}) \cdot P(\text{so} \mid \text{its water is}) \cdot P(\text{transparent} \mid \text{its water is so})$$

Auto-regressive Language Models

- The chain rule applied to compute joint probability of words in sentence

$$P(X) = \prod_{i=1}^{I} P(\underbrace{x_i}_{\text{Next Token}} \mid \underbrace{x_1, \ldots, x_{i-1}}_{\text{Context}})$$

The big problem: How do we predict

$$P(x_i \mid x_1, \ldots, x_{i-1})$$

?!?!

How to estimate these probabilities?

$$P(\text{its water is so transparent}) =$$
$$P(\text{its}) \cdot P(\text{water} \mid \text{its}) \cdot P(\text{is} \mid \text{its water}) \cdot P(\text{so} \mid \text{its water is}) \cdot P(\text{transparent} \mid \text{its water is so})$$

- Could we just count and divide?

$$P(\text{transparent} \mid \text{its water is so}) = \frac{\text{Count}(\text{its water is so transparent})}{\text{Count}(\text{its water is so})}$$

- No! Too many possible sentences.
- We’ll never see enough data for estimating these

N-gram Language Models

Markov Assumption

- Simplifying assumption:

$$P(\text{transparent} \mid \text{its water is so}) \approx P(\text{transparent} \mid \text{so})$$

or

$$P(\text{transparent} \mid \text{its water is so}) \approx P(\text{transparent} \mid \text{is so})$$

- Markov Assumption Equation

$$P(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} P(x_i \mid x_{i-k+1}, x_{i-k+2}, \ldots, x_{i-1})$$

- In other words, we approximate each component in the product

$$P(x_i \mid x_1, x_2, \ldots, x_{i-1}) \approx P(x_i \mid x_{i-k+1}, x_{i-k+2}, \ldots, x_{i-1})$$

The Simplest LM: Count-based Unigram Models

- Let’s choose the simplest one for now.
- Independence assumption:

$$P(x_i|x_1,\ldots,x_{i-1}) \approx P(x_i) \qquad P(x_1,x_2,\ldots,x_n) \approx \prod_{i=1}^{n} P(x_i)$$

- some automatically generated sentences from a unigram model:

fifth, an, of, futures, the, an, incorporated, a, a,
the, inflation, most, dollars, quarter, in, is , mass

The Simplest LM: Count-based Unigram Models

- How to estimate $P(x_i)$
- Count-based maximum-likelihood estimation:

$$P_{\mathrm{MLE}}(x_i) = \frac{c_{\mathrm{train}}(x_i)}{\sum_{\tilde{x}} c_{\mathrm{train}}(\tilde{x})}$$

Handling Unknown Words

- If a token doesn't exist in training data, $\frac{c_{\text{train}}(x_i)}{\sum_{\tilde{x}} c_{\text{train}}(\tilde{x})}$ becomes zero!

- Two options:
  - **Segment to characters/subwords**: Make sure that all tokens are in vocabulary
  - **Unknown word model**: create a character/word based model for unknown words and interpolate.

$$P(x_i) = (1 - \lambda_{\text{unk}}) * P_{\text{MLE}}(x_i) + \lambda_{\text{unk}} * P_{\text{unk}}(x_i)$$

Parameterizing in Log Space

- Multiplication of probabilities can be re-expressed as addition of log probabilities

$$P(X) = \prod_{i=1}^{|X|} P(x_i) \longrightarrow \log P(X) = \sum_{i=1}^{|X|} \log P(x_i)$$

- Why?
  - Avoid underflow
  - (also adding is faster than multiplying)

Bigram Model

$$P(x_i \mid x_1, x_2, \ldots, x_{i-1}) \approx P(x_i \mid x_{i-1})$$

texaco, rose, one, in, this, issue, is, pursuing, growth, in, a, boiler, house, said, mr., gurria, mexico, 's, motion, control, proposal, without, permission, from, five, hundred, fifty, five, yen

Higher-order *n*-gram Models

- Limit context length to *n*, count, and divide

$$P_{ML}(x_i \mid x_{i-n+1}, \ldots, x_{i-1}) := \frac{c(x_{i-n+1}, \ldots, x_i)}{c(x_{i-n+1}, \ldots, x_{i-1})}$$

$$\text{P(example | this is an)} = \frac{\text{c(this is an example)}}{\text{c(this is an)}}$$

- Add smoothing, to deal with zero counts:

$$\begin{aligned} P(x_i \mid x_{i-n+1}, \ldots, x_{i-1}) =& \lambda P_{ML}(x_i \mid x_{i-n+1}, \ldots, x_{i-1}) \\ &+ (1-\lambda) P(x_i \mid x_{1-n+2}, \ldots, x_{i-1}) \end{aligned}$$

Problems?

- Cannot share strength among similar words

she bought a car	she bought a bicycle
she purchased a car	she purchased a bicycle

- Cannot condition on context with intervening words

Dr. Jane Smith   Dr. Gertrude Smith

- Cannot handle long-distance dependencies

for tennis class he wanted to buy his own racquet

for programming class he wanted to buy his own computer

When to use n-gram models?

- Neural language models achieve better performance, but
- n-gram models are extremely fast to estimate/apply
- n-gram models can be better at modeling low-frequency phenomena

LM Evaluation

Evaluation: How good is our model?

- Does our language model prefer good sentences to bad ones?
  - We train parameters of our model on a **training set**.
  - We test the model's performance on data we haven't seen.
    - A **test set** is an unseen dataset
    - An **evaluation metric** tells us how well our model does on the test set.
- Extrinsic evaluation of language model
  - Put a model in a task (e.g., spelling corrector, MT system, etc) and evaluate its performance
- But, extrinsic evaluation can be time-consuming
- **Intrinsic evaluation** can be useful as pilot experiments

Likelihood

- Log-likelihood:

$$LL(\mathcal{X}_{\text{test}}) = \sum_{X \in \mathcal{X}_{\text{test}}} \log P(X)$$

- Per-word Log Likelihood:

$$WLL(\mathcal{X}_{\text{test}}) = \frac{1}{\sum_{X \in \mathcal{X}_{\text{test}}} |X|} \sum_{X \in \mathcal{X}_{\text{test}}} \log P(X)$$

Papers often also report negative log likelihood (lower better), as that is used in loss.

Entropy

- Per-word (cross) Entropy

$$H(\mathcal{X}_{\text{test}}) = \frac{1}{\sum_{X \in \mathcal{X}_{\text{test}}} |X|} - \sum_{X \in \mathcal{X}_{\text{test}}} \log_2 P(X)$$

Why log_2?

Perplexity

$$PPL(\mathcal{X}_{\text{test}}) = 2^{H(\mathcal{X}_{\text{test}})} = e^{-WLL(\mathcal{X}_{\text{test}})}$$

When a dog sees a squirrel it will usually _____

Token: ' be' - Probability: 0.0352 → PPL= 28.4
Token: ' jump' - Probability: 0.0338 → PPL= 29.6
Token: ' start' - Probability: 0.0289 → PPL= 34.6
Token: ' run' - Probability: 0.0277 → PPL= 36.1
Token: ' try' - Probability: 0.0219 → PPL= 45.7

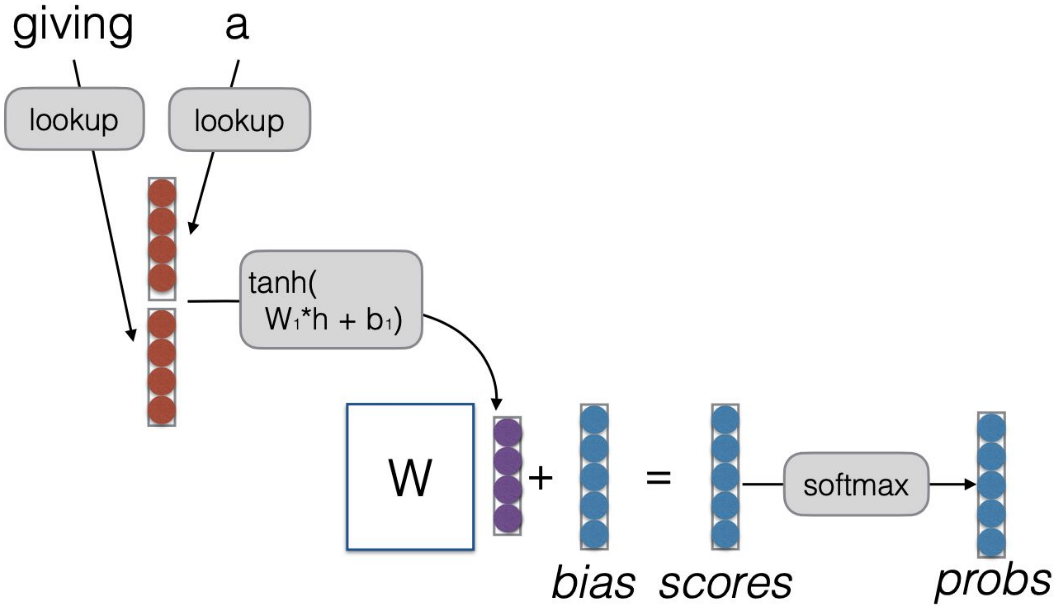
Back to Language Models

Neural Language Models

- Language Modeling: Calculating the probability of the next word in a sequence given some history.
  - We've seen N-gram based LMs
  - But neural network LMs far outperform n-gram language models
- State-of-the-art neural LMs are based on more powerful neural network technology like Transformers
- But simple feedforward LMs can do almost as well!

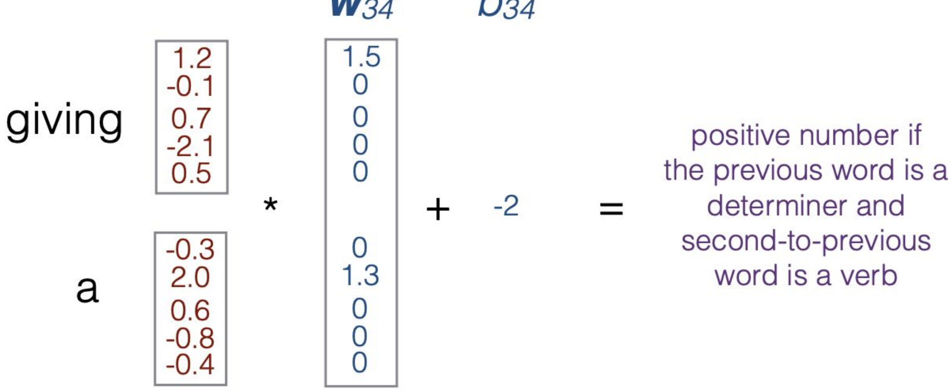
Feed-forward Neural Language Models (Bengio et al. 2003)

- Task: predict next word w_t , given prior words w_{t-1}, w_{t-2}, w_{t-3}, ...

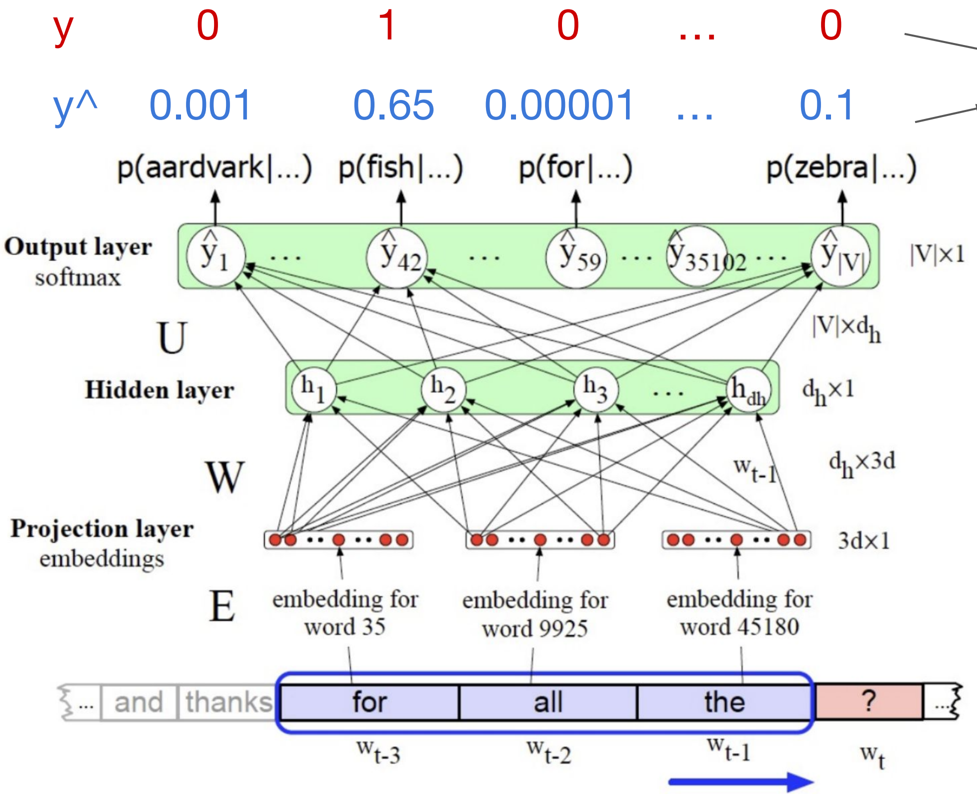


Example of Combination Features

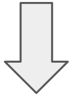
- Word embeddings capture features of words
  - e.g. feature 1 indicates verbs, feature 2 indicates determiners
- A row in the weight matrix (with the bias) can capture particular combinations of these features
  - e.g. the 34th row in the weight matrix looks at feature 1 in the second-to-previous word, and feature 2 in the previous word



Feed-forward Neural Language Models

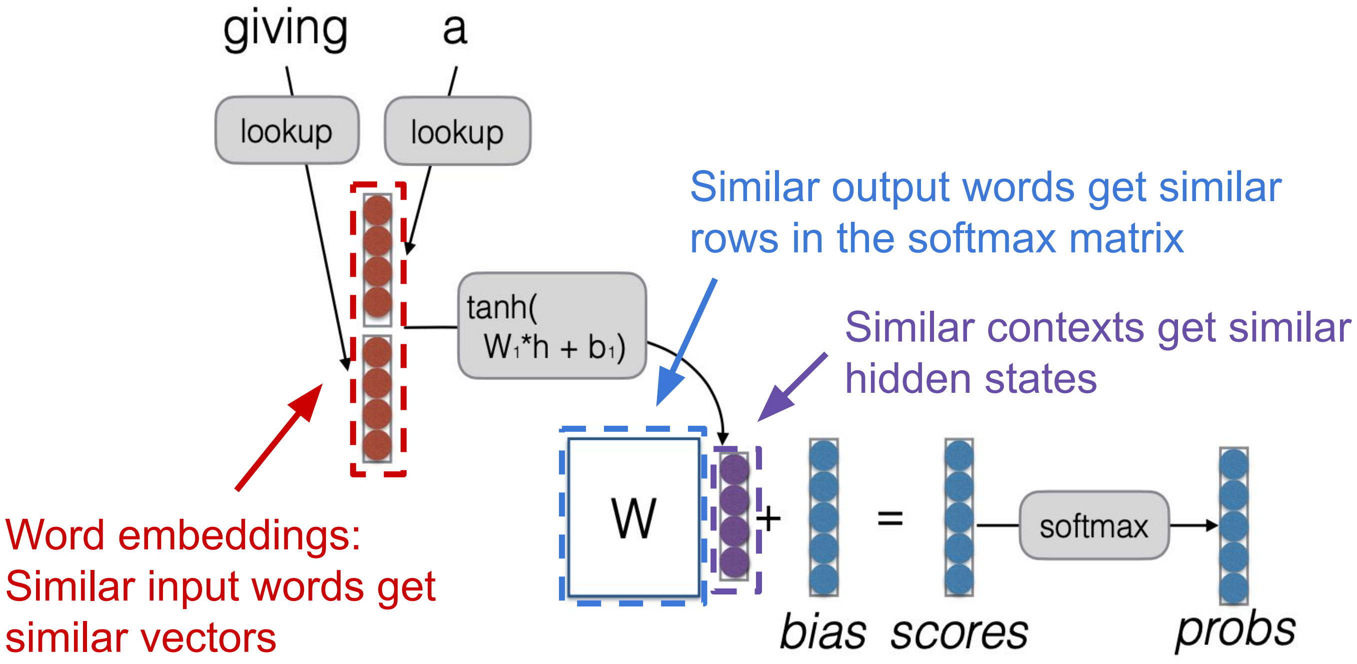


Compute the loss using cross entropy loss

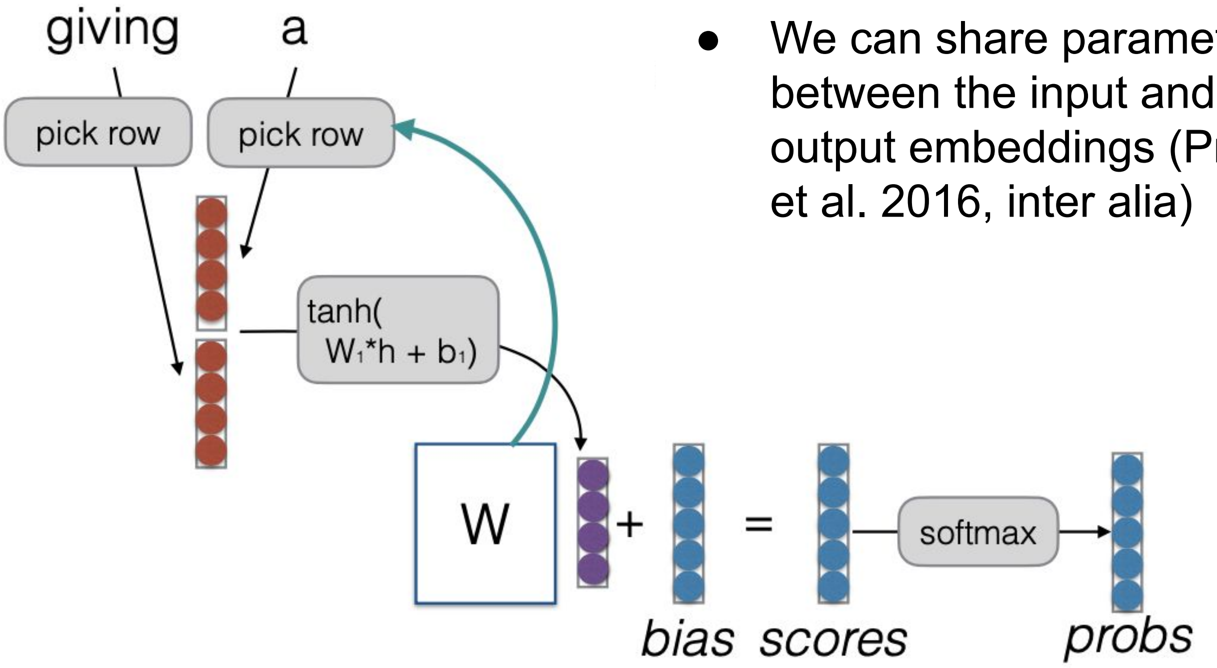


Backprop

Where is Strength Shared?



Tying Input/Output Embeddings



- We can share parameters between the input and output embeddings (Press et al. 2016, inter alia)

What Problems are Handled?

- Cannot share strength among similar words

she bought a car	she bought a bicycle
she purchased a car	she purchased a bicycle

→ solved, and similar contexts as well!

- Cannot condition on context with intervening words

Dr. Jane Smith	Dr. Gertrude Smith

→ solved!

- Cannot handle long-distance dependencies

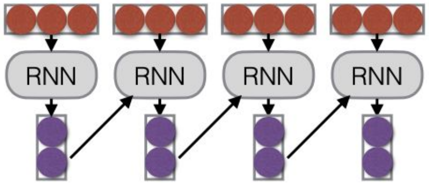
for tennis class he wanted to buy his own racquet
for programming class he wanted to buy his own computer

→ not solved yet!

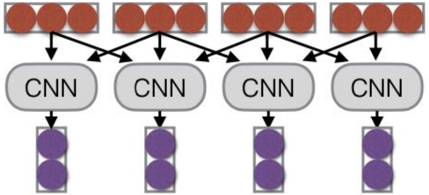
Full Sequence Models

Three Major Types of Sequence Models

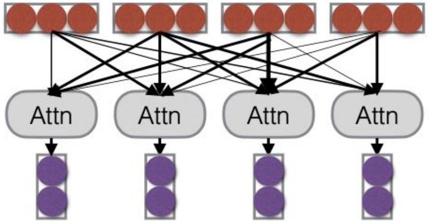
- **Recurrence**: Condition representations on an encoding of the history



- **Convolution**: Condition representations on local context



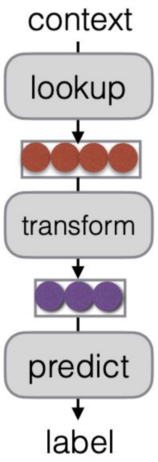
- **Attention**: Condition representations on a weighted average of all tokens



Recurrent Neural Networks (RNN) (Elman 1990)

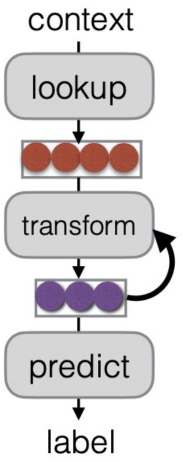
- Tools to “remember” information

Feed-forward NN



$$h_t = \mathrm{f}(W_x x_t + b)$$

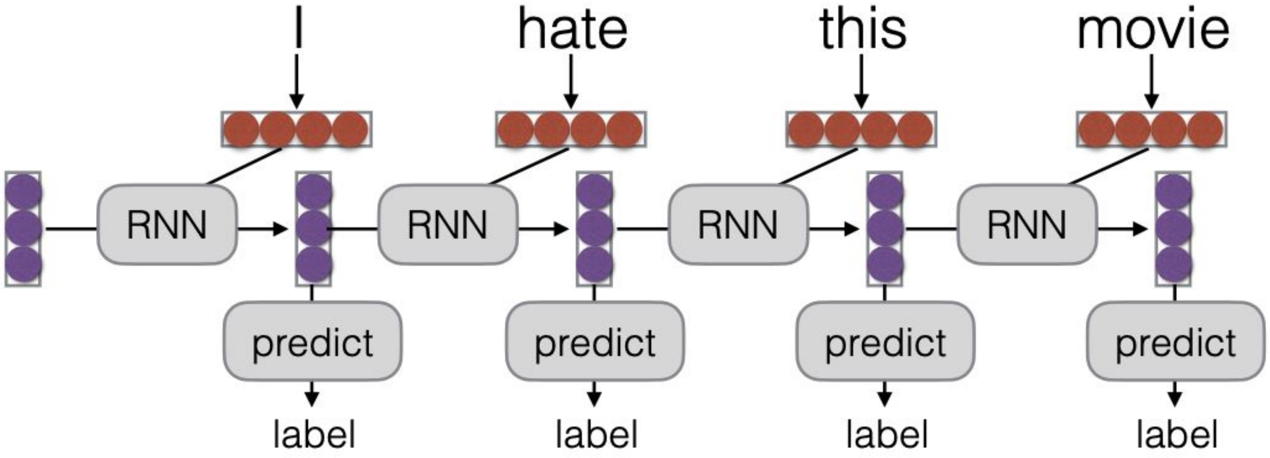
Recurrent NN



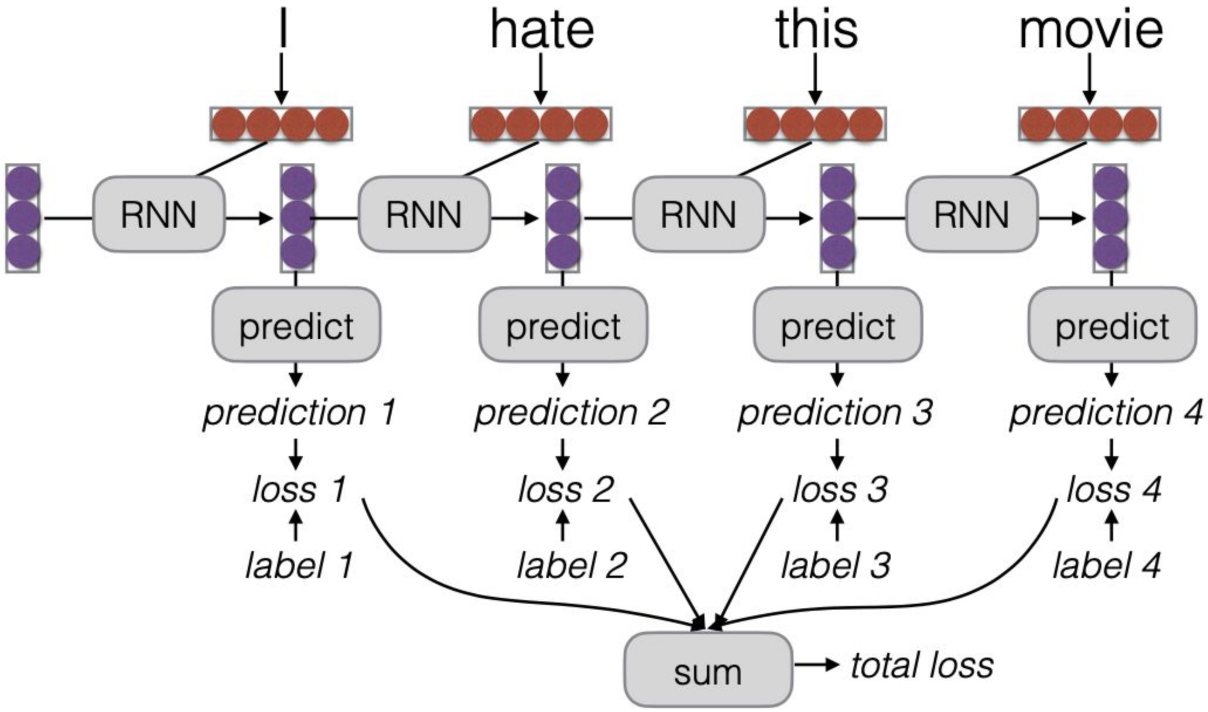
$$h_t = \mathrm{f}(W_h h_{t-1} + W_x x_t + b)$$

Unrolling in Time

- What does processing a sequence look like?



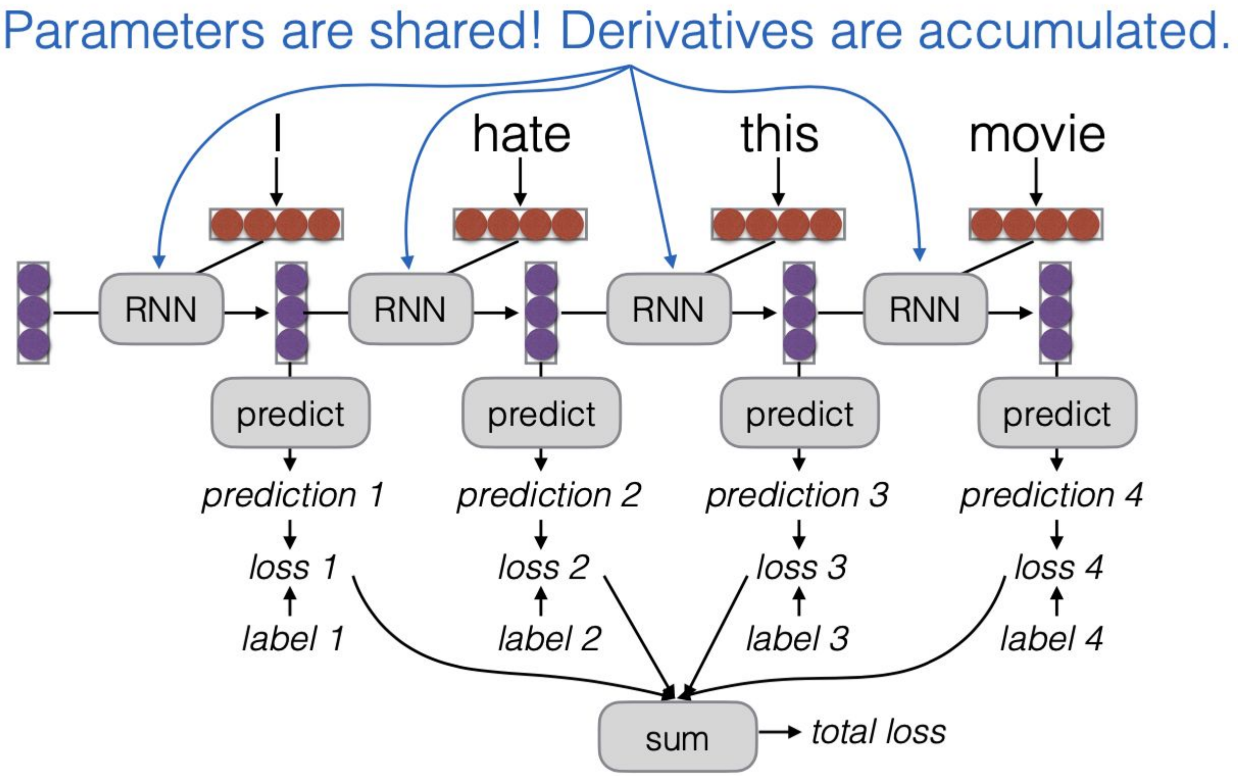
Training RNNs



RNN Training

- The unrolled graph is a well-formed RNN Training (DAG) computation graph—we can run backprop!
- Parameters are tied across time, derivatives are aggregated across all time steps
- This is historically called “backpropagation through time” (BPTT)

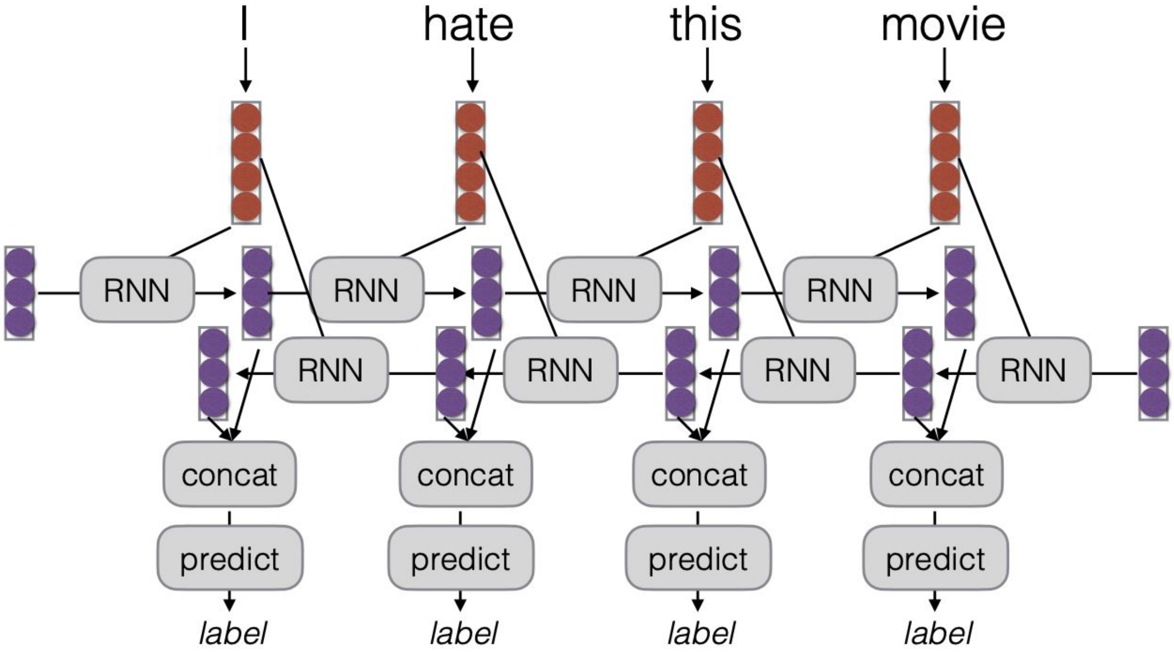
Parameter Tying



(Same for attention, convolutional networks)

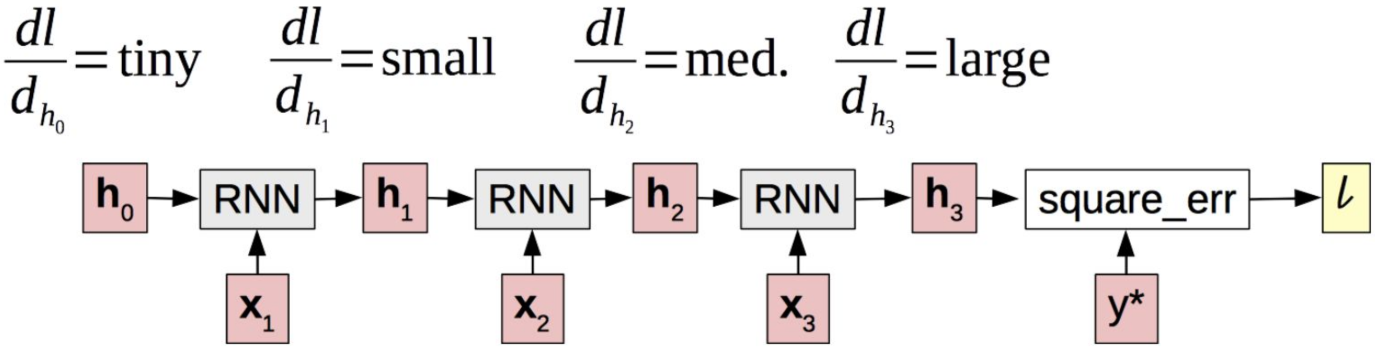
Bi-RNNs

- A simple extension, run the RNN in both directions



Vanishing Gradient

- Gradients decrease as they get pushed back

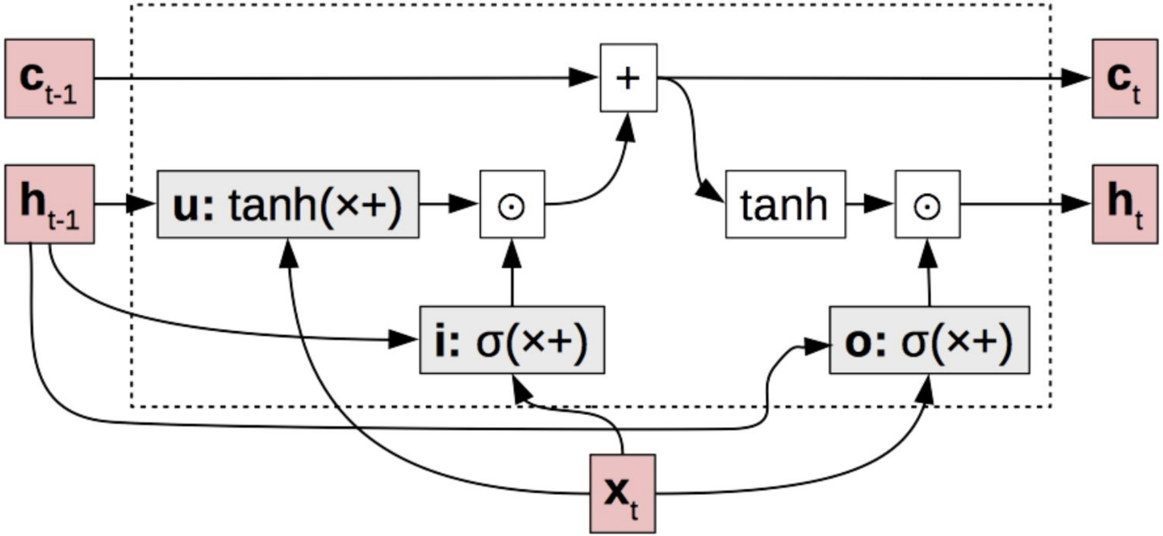


- Why? “Squashed” by non-linearities or small weights in matrices

A Solution: Long Short-term Memory (Hochreiter and Schmidhuber 1997)

- **Basic idea**: make additive connections between time steps
- Addition does not modify the gradient, no vanishing
- Gates to control the information flow

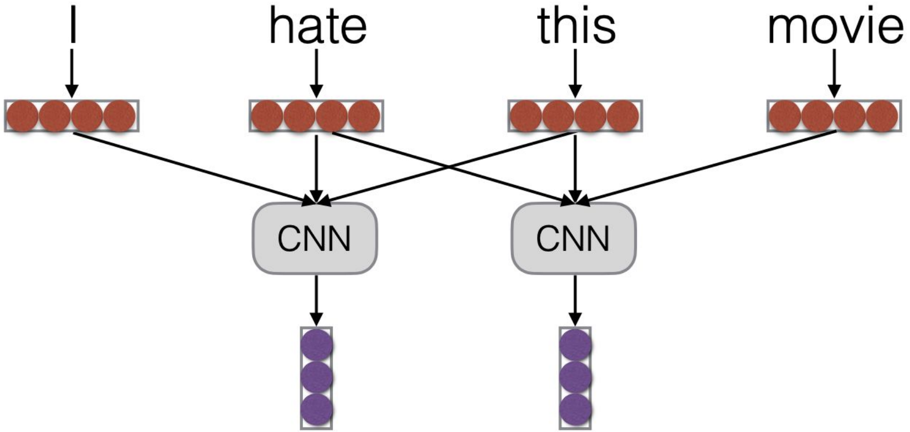
LSTM Structure



update **u**: what value do we try to add to the memory cell?
input **i**: how much of the update do we allow to go through?
output **o**: how much of the cell do we reflect in the next state?

Convolution

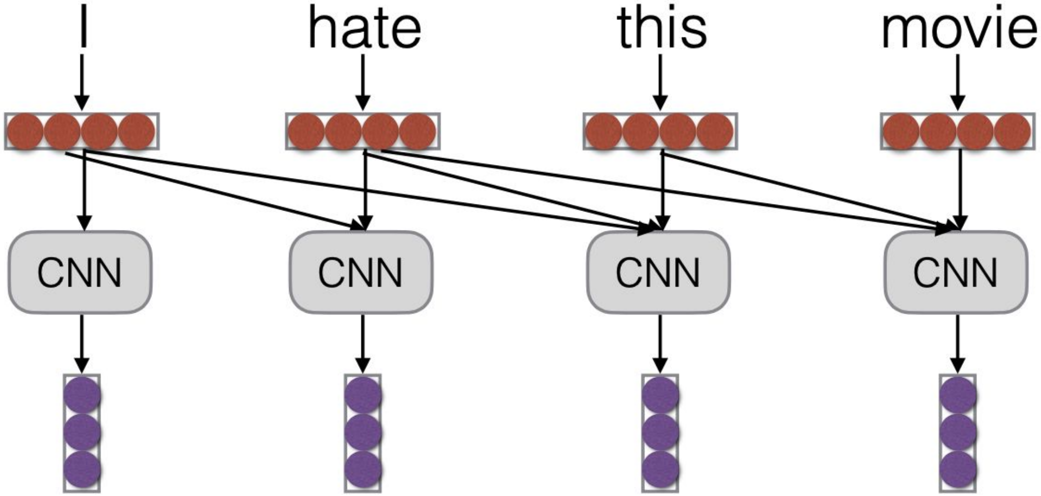
- Calculate based on local context



$$h_t = \mathrm{f}(W[x_{t-1}; x_t; x_{t+1}])$$

Convolution for Auto-regressive Models

- Functionally identical, just consider previous context



Any Questions?