Language models

CS 685, Spring 2021
Advanced Topics in Natural Language Processing
http://brenocon.com/cs685
https://people.cs.umass.edu/~brenocon/cs685_s21/

Brendan O'Connor
College of Information and Computer Sciences
University of Massachusetts Amherst

Announcements

- Hope HW1 is going well!
- Reading review #2 due this Monday
- I'll post some sample lit review topics

Watergate ____

Scandal
hotel
incident

airplane

- prosecutor ____ Watergate ____

investigation

scandal

hotel

incident

[illegible]

- order government official president attorney general deputy attorney general prosecutor Watergate ___

- Ms. Yates's order was a remarkable rebuke by a government official to a sitting president, and it recalled the so-called Saturday Night Massacre in 1973, when President Richard M. Nixon fired his attorney general and deputy attorney general for refusing to dismiss the special prosecutor in the Watergate ____ case ("the", "a", "hotel", ...

Language models

P(Watergate case) = high?

P(the the the of the) = low

doc

sentences

- $P(\text{text}) = P(w_1, w_2 \ldots w_N) =$

chain rule

left-to-right

$$= P(w_1)\, P(w_2 \mid w_1)\, P(w_3 \mid w_1, w_2)\, P(w_4 \mid w_1, w_2, w_3) \ldots$$

$$= \prod_{i=1}^{n} P(w_i \mid w_1, w_2, \ldots w_{i-1})$$

"causal"

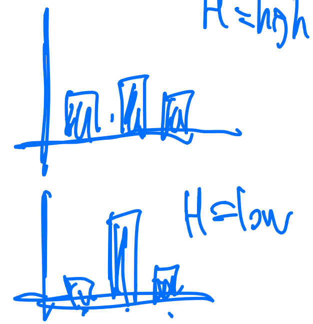
Will prob corr with quality??

tricky!!

Language models

- $P(\text{text}) = P(w_1, w_2 \ldots w_N) = \prod_i P(w_i \mid w_1..w_{i-1})$
  [history-based a.k.a. "causal"]
- **Generation:**
  - Actually generate from the language distribution
  - Evaluate quality of proposed translations/outputs
- **Unsupervised transfer learning**: Induce useful **word or token embeddings** for *other* NLP tasks
  - Pretrained word embeddings (last lecture)
  - Pretrained token (contextual) embeddings: ELMO, BERT…
- Usual assumption: train on very large corpus (>10M, >100M tokens)
- Model types
  - Short- or long-distance?
  - Explicit features or neural representations?
  - Model architecture to transmit sequential/structural information?

Information theory perspective



Entropy: uncertainty in distribution P
(obeys reasonable axioms)

$$H(P) = \sum_{\mathbf{x}} P(\mathbf{x}) \log \frac{1}{P(\mathbf{x})}$$

surprisal

Cross-entropy: model P_M, test distribution P_T
(equiv. to average neg. log-likelihood)

P_T: Test
P_M: Model

$$H'(P_T; P_M) = -\sum_{\mathbf{x}} P_T(\mathbf{x}) \cdot \log P_M(\mathbf{x})$$

- Coding interpretation: average number of bits/nats
- Entropy of uniform V-sided die?

Unif PPL $= \exp\left(-\frac{1}{N}\sum_i \log \frac{1}{V}\right) = \exp(\log V) = V$

Information theory perspective

$$\mathrm{Ppl} = \exp\left(-\frac{1}{N}\sum_{i=1}^{N} \log_2 P(w_i \mid w_1 \ldots w_{i-1})\right)$$

Perplexity =

$S = \left[\frac{1}{V}, \frac{1}{V}, \ldots, \frac{1}{V}\right]$

exp (Cross-entropy: model P_M, test distribution P_T (equiv. to average neg. log-likelihood)

$$H'(P_T; P_M) = -\sum_{\mathbf{x}} P_T(\mathbf{x}) \cdot \log P_M(\mathbf{x})$$

)

Uniform Dist Model PPL = 20,000 = V

$P(w) = \frac{\#(w)}{\#\text{tokens}}$

WSJ Penn Treebank
$\boxed{V = 20{,}000}$
1.5 M test tokens

- Unigram ($n = 1$): 962
- Bigram ($n = 2$): 170
- Trigram ($n = 3$): 109

N-gram models

$$\prod_i P(w_i \mid w_1 \dots w_{i-1}) = \prod_i P(w_i \mid w_{i-5} \dots w_{i-1})$$

- Markov assumption: only use *short distance* information, within a fixed window, say k=5 (Markov window)
- "N-gram LMs": Markov models with count-based parameter fitting

suff. stats = N-gram counts

$$P(w_3 = \text{case} \mid w_1, w_2 = (\text{the}, \text{Watergate}))$$

$$= \frac{\#(\text{the}, \text{Watergate}, \text{case})}{\#(\text{the Watergate})}$$

Rel. freq. count estimator

Smoothing

$\alpha = \frac{1}{V}$

$\alpha = .1$

- Pseudocount Smoothing (Dirichlet prior) for parameter estimation:

$$\mathrm{p}_{\text{smooth}}(w_m \mid w_{m-1}) = \frac{\text{count}(w_{m-1}, w_m) + \alpha}{\sum_{w' \in \mathcal{V}} \text{count}(w_{m-1}, w') + V\alpha}$$

- Note smoothing usually redistributes mass from seen words to unseen words

- Absolute Discounting: when count>0, subtract d $(0<d<1)$

- Smoothing is important for many other word statistics-based preprocessing methods, like identifying multiword expressions a.k.a. collocations ("social security")

Interpolation

- Idea: higher Markov orders are more sparse. So *combine* multiple order models
- Interpolation: weighted averaging ($\lambda \geq 0, \Sigma\, \lambda_n = 1$):

$$\begin{aligned}\mathrm{p}_{\text{Interpolation}}(w_m \mid w_{m-1}, w_{m-2}) = &\lambda_3 \mathrm{p}_3^*(w_m \mid w_{m-1}, w_{m-2}) \\ &+ \lambda_2 \mathrm{p}_2^*(w_m \mid w_{m-1}) \\ &+ \lambda_1 \mathrm{p}_1^*(w_m).\end{aligned}$$

- It's a generative model:

> **for** Each token $w_m, m = 1, 2, \ldots, M$ **do**:
> draw the n-gram size $z_m \sim \text{Categorical}(\lambda)$;
> draw $w_m \sim \mathrm{p}_{z_m}^*(w_m \mid w_{m-1}, \ldots, w_{m-z_m})$.

- If only we knew z, learning would be easy.
- So… use EM!

EM for the interpolation model

for Each token $w_m, m = 1, 2, \ldots, M$ **do**:
  draw the n-gram size $z_m \sim \text{Categorical}(\lambda)$;
  draw $w_m \sim \mathrm{p}^*_{z_m}(w_m \mid w_{m-1}, \ldots, w_{m-z_m})$.

Algorithm 10 Expectation-maximization for interpolated language modeling

```
procedure ESTIMATE INTERPOLATED n-GRAM (w_{1:M}, {p*_n}_{n∈1:n_max})
    for z ∈ {1, 2, ..., n_max} do                          ▷ Initialization
        λ_z ← 1/n_max
    repeat
        for m ∈ {1, 2, ..., M} do                          ▷ E-step
            for z ∈ {1, 2, ..., n_max} do
                q_m(z) ← p*_z(w_m | w_{1:m-}) × λ_z
            q_m ← Normalize(q_m)
        for z ∈ {1, 2, ..., n_max} do                      ▷ M-step
            λ_z ← (1/M) Σ_{m=1}^{M} q_m(z)
    until tired
    return λ
```

$$\lambda_z \leftarrow \frac{1}{n_{\max}}$$

$$q_m(z) \leftarrow \mathrm{p}^*_z(w_m \mid \boldsymbol{w}_{1:m-}) \times \lambda_z$$

$$\boldsymbol{q}_m \leftarrow \text{Normalize}(\boldsymbol{q}_m)$$

$$\lambda_z \leftarrow \frac{1}{M}\sum_{m=1}^{M} q_m(z)$$

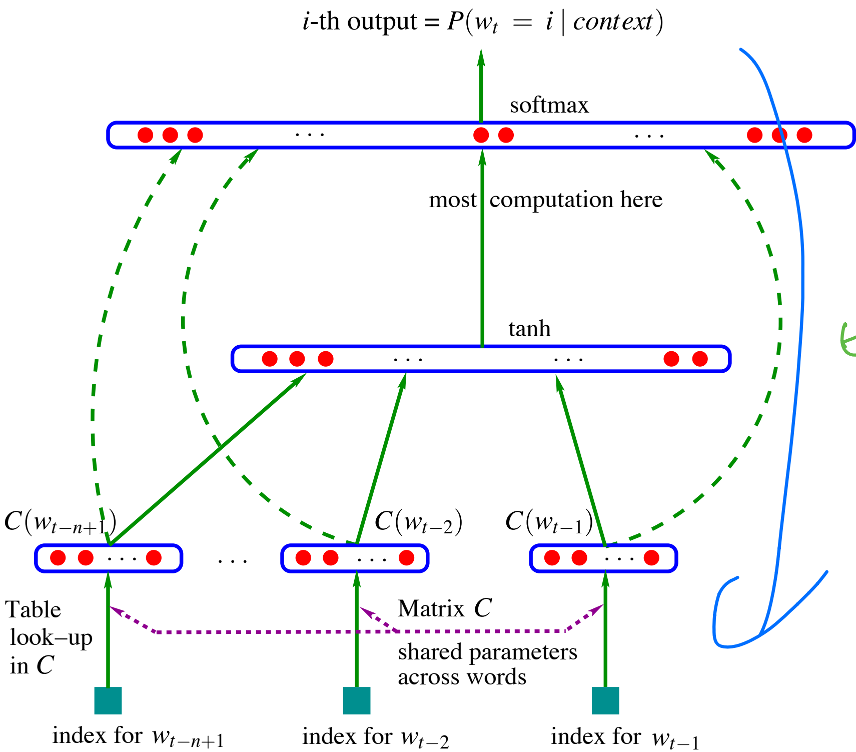
Logistic word prediction

- No more counts. Model next-word as softmax over the vocabulary.
- We can use anything to help predictions: features (Rosenfeld 1996) or neural nets (Bengio et al. 2003) to compose $\mathbf{v_u}$:

$$\mathrm{p}(w \mid u) = \frac{\exp(\boldsymbol{\beta}_w \cdot \boldsymbol{v}_u)}{\sum_{w' \in \mathcal{V}} \exp(\boldsymbol{\beta}_{w'} \cdot \boldsymbol{v}_u)} \qquad \begin{aligned} \boldsymbol{\beta}_w &\in \mathbb{R}^K \\ \boldsymbol{v}_u &\in \mathbb{R}^K \end{aligned}$$

- Can use character-level models (helps with out-of-vocabulary words), long-distance topical information, or *any* type of other information from the left context!

Bengio et al. 2003: Markov multilayer perceptron



Learn: C, W, U, H, d (chain rule)

$$\hat{P}(w_t | w_{t-1}, \cdots w_{t-n+1}) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}}.$$

Output layer (softmax / log-linear)

$$y = \underbrace{b + Wx}_{\text{shortcut linear layer}} + \underbrace{U \tanh(d + Hx)}_{\text{another hidden layer, size } h}$$

Vocab output: log-probs size V

$$x = (C(w_{t-1}), C(w_{t-2}), \cdots, C(w_{t-n+1}))$$

Lookup layer with concatenation: (kinda) hidden layer size $(n-1)m$

$C(i) \in \mathbb{R}^m$ Word embedding parameters

Pros/cons

V^k

Neural Markov (vs. Ngrams)

- Fewer params
- Add more info
  - + out-of-vocab smartness
- Generalization!

Ngram Pros

- Fast to train
- Simple
- Interp

Long-distance: Human LMs?

2.5 Long Distance (Triggers)

2.5.1 Evidence for Long Distance Information

Shannon Game at IBM [Mercer and Roukos 92]. A "Shannon game" program was implemented at IBM, where a person tries to predict the next word in a document while given access to the entire history of the document. The performance of humans was compared to that of a trigram language model. In particular, the cases where humans outsmarted the model were examined. It was found that in 40% of these cases, the predicted word, or a word related to it, occurred in the history of the document.

- Cognitive science & human behavioral evidence can inspire better NLP modeling
- Inspecting *differences* in two models' performance (here, human-vs-machine; can also do machine-vs-machine)

[Rosenfeld 1996]

Long-distance conditioning

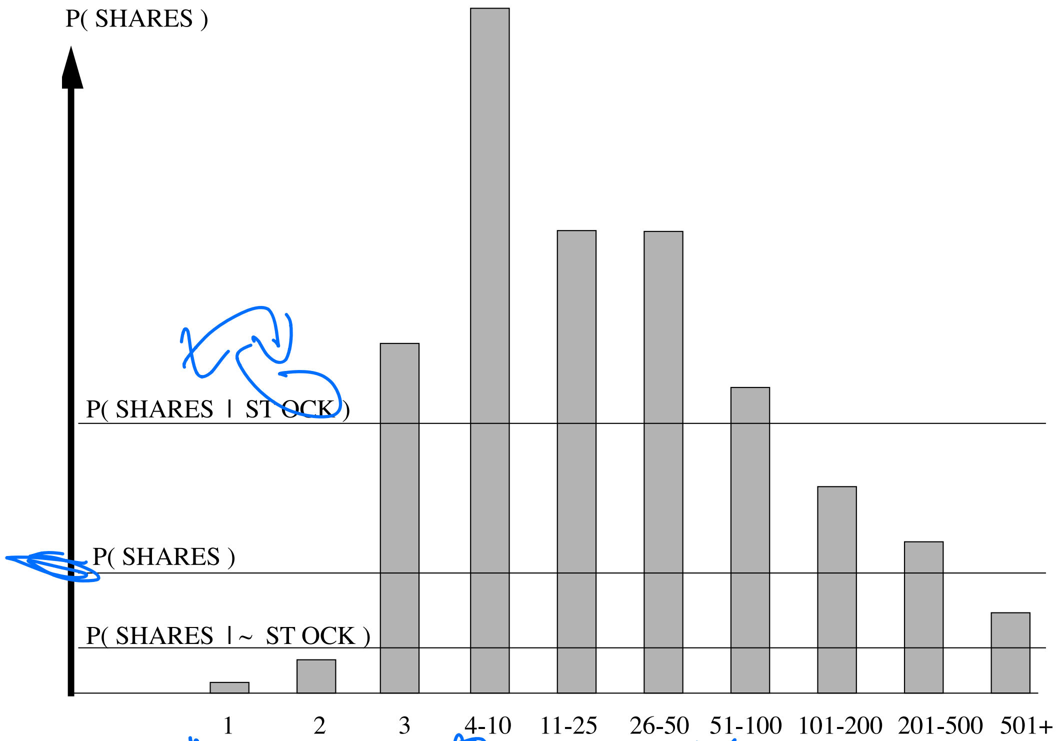


Figure 2: Probability of 'SHARES' as a function of the distance from the last occurrence of 'STOCK' in the same document. The middle horizontal line is the unconditional probability. The top (bottom) line is the probability of 'SHARES' given that 'STOCK' occurred (did not occur) before in the document.

[Rosenfeld 1996]

Recurrent neural networks

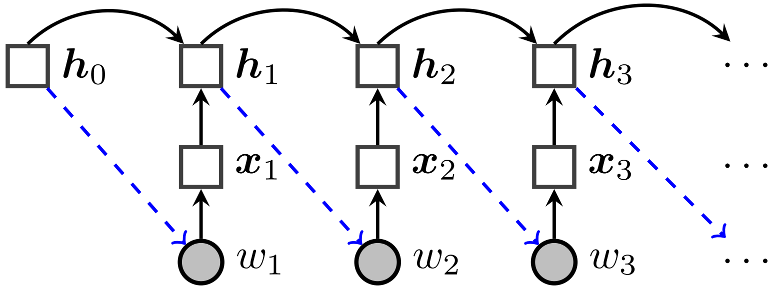


Figure 6.1: The recurrent neural network language model, viewed as an "unrolled" computation graph. Solid lines indicate direct computation, dotted blue lines indicate probabilistic dependencies, circles indicate random variables, and squares indicate computation nodes.

- Idea: extend a feedforward net to sequential data by iterating an NN at each position.
- Theoretically, an RNN can learn *any* update function. (Represent any Turing machine!)