Log-linear models (part 1)

CS 690N, Spring 2018

Advanced Natural Language Processing

http://people.cs.umass.edu/~brenocon/anlp2018/

Brendan O'Connor

College of Information and Computer Sciences

University of Massachusetts Amherst

NLP/ML talks this week!

- Michael Jordan - Distinguished Lecture, today (4:30pm)
  - *On Computational Thinking, Inferential Thinking and Data Science*
- Graham Neubig - MLFL, Thursday, 12pm
  - *What Can Neural Networks Teach us about Language*
- Allison Chaney - DS Seminar, Thursday, 4pm
  - *The Social Side of Recommendation Systems: How Groups Shape Our Decisions*

- Issues in n-gram models
  - Lexical generalization
    - S+P: latent lexical variables
  - Non-local constraints

- Rosenfeld (1996)
  - Data analysis of long-distance lexical effects (topicality??)
  - Incorporate into "MaxEnt" (a.k.a. Log-Linear) language model: allow **multiple sources of information**
  - Early example of machine learning-based prediction for language modeling
  - *[The iterative scaling algorithm less interesting; gradient descent, or rather L-BFGS, has since been found to be better]*

-

Watergate ___

-

prosecutor Watergate ___

- 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 ___

Information theory perspective

Entropy: uncertainty in distribution P
(obeys reasonable axioms)

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

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

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

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

Information theory perspective

Perplexity =

$$\exp\left(\begin{array}{c}\text{Cross-entropy: model } P_M\text{, test distribution } P_T \\ \text{(equiv. to average neg. log-likelihood)} \\ H'(P_T; P_M) = -\sum_{\mathbf{x}} P_T(\mathbf{x}) \cdot \log P_M(\mathbf{x})\end{array} \right)$$

WSJ Penn Treebank
V = 20,000
1.5 M test tokens

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

Will this trend continue?

Behavioral data!

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 much better NLP modeling
- Inspecting *differences* in two models' performance (here, human-vs-machine; can also do machine-vs-machine)

- How to capture long distance information?
- Attempt 1: fixed distance a.k.a. *skip-grams*
  - $P(w_t \mid w_{t-1})$, $P(w_t \mid w_{t-10})$ ...

distance	1	2	3	4	5	6	7	8	9	10	1000
PP	83	119	124	135	139	138	138	139	139	139	141

still a gap:
information
"spread thinly"

- Attempt 2: Trigger pairs
  - event "A -> B": ngram A occurred anywhere in document before ngram B (brest -> litovsk, stock -> bond)
  - count(A,B): sparsity compared to Markov bigram model?
  - $P(w_t = A \mid B \in (w_0 \ldots w_{t-1}))$

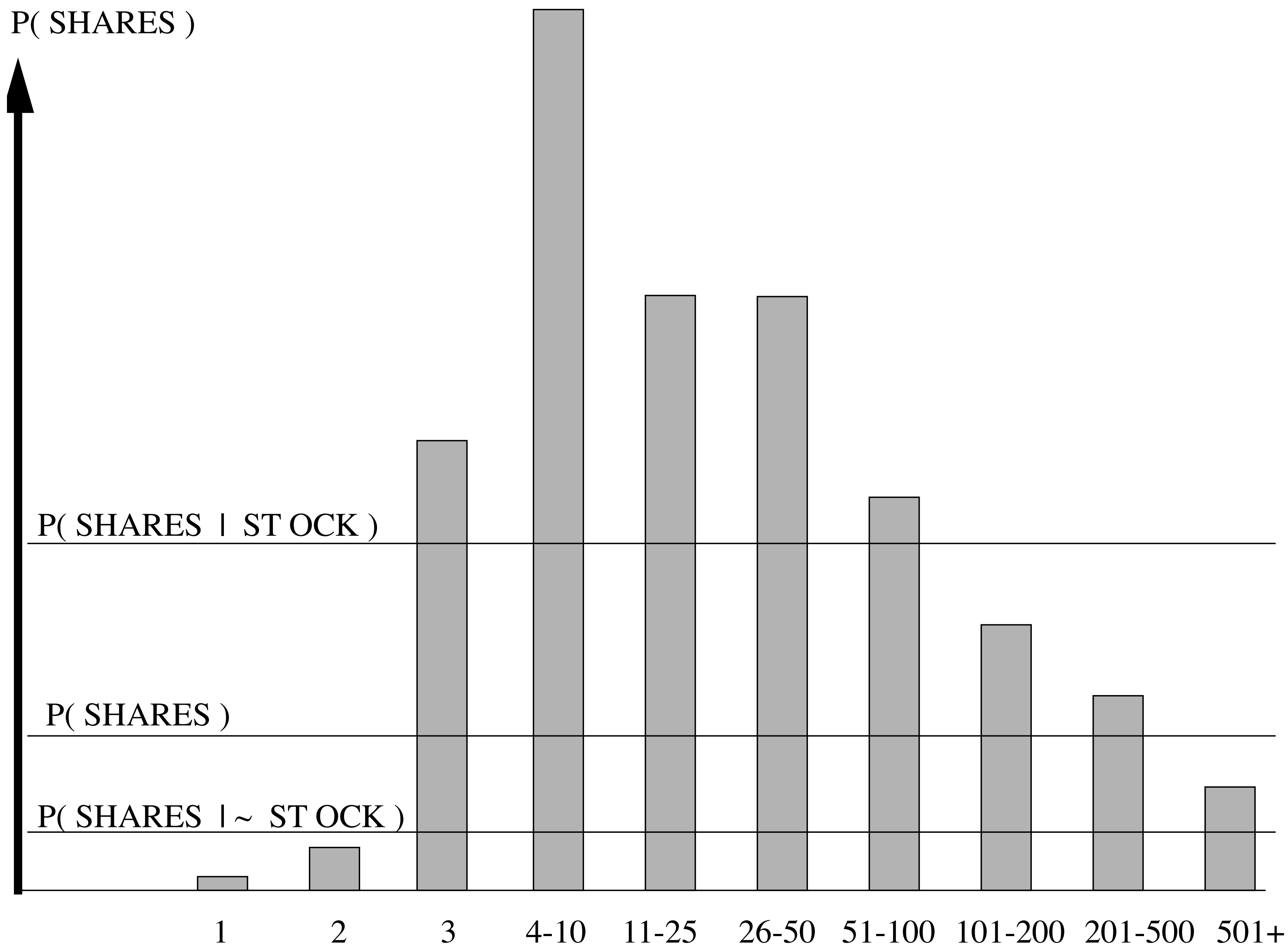

Figure 2: Probability of ’SHARES’ as a function of the distance from the last occurrence of ’STOCK’ in the same document. The middle horizontal line is the unconditional probability. The top (bottom) line is the probability of ’SHARES’ given that ’STOCK’ occurred (did not occur) before in the document.

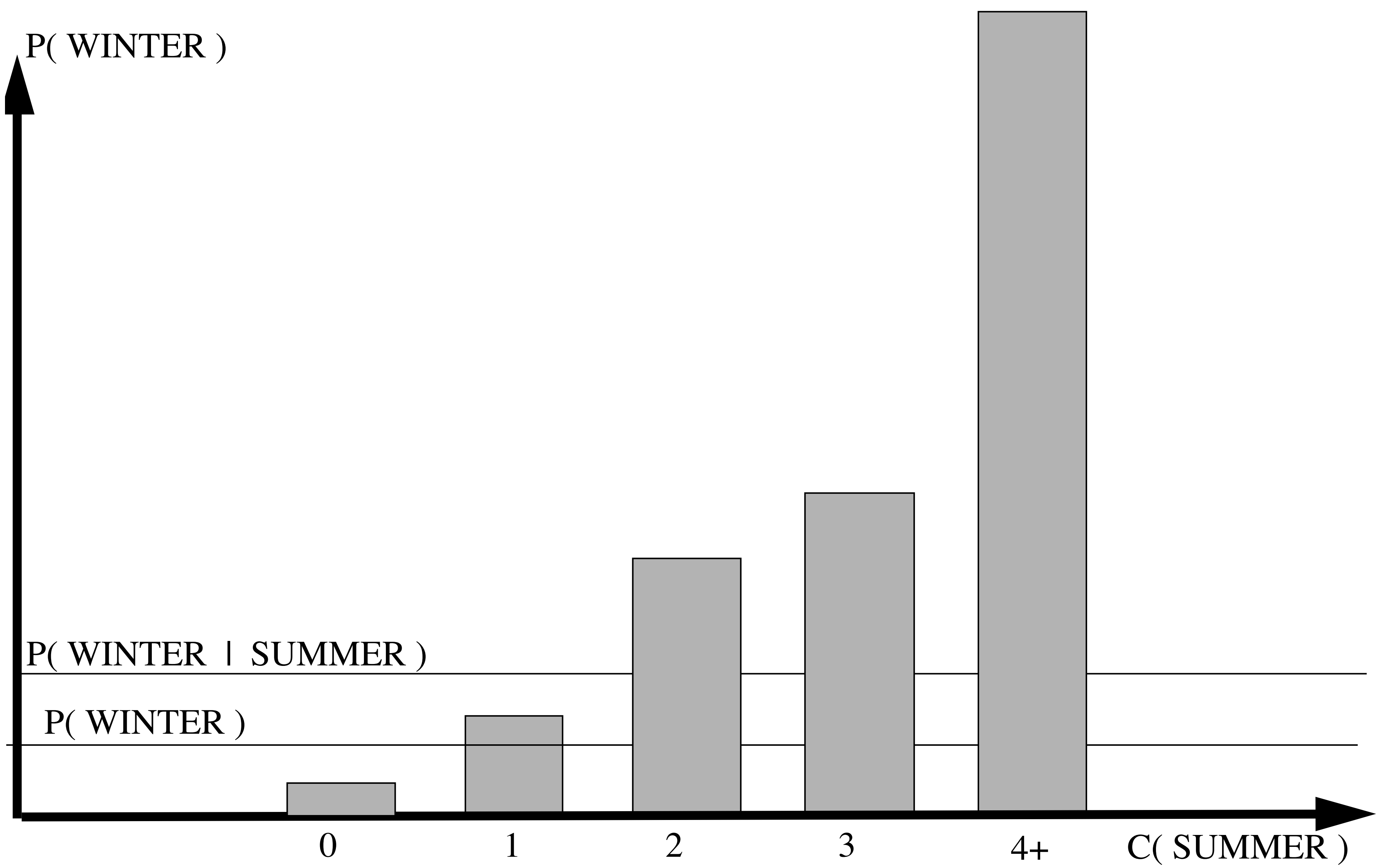

Figure 3: Probability of 'WINTER' as a function of the number of times 'SUMMER' occurred before it in the same document. Horizontal lines are as in fig. 2.

Feature selection

- Want to filter many possible (A,B) trigger pairs
- Mutual information to score them:

$$
\begin{aligned}
I(A_0 : B) \quad = \quad & P(A_0, B) \log \frac{P(B|A_0)}{P(B)} \quad + \quad P(A_0, \overline{B}) \log \frac{P(\overline{B}|A_0)}{P(\overline{B})} \\
& + \; P(\overline{A_0}, B) \log \frac{P(B|\overline{A_0})}{P(B)} \quad + \quad P(\overline{A_0}, \overline{B}) \log \frac{P(\overline{B}|\overline{A_0})}{P(\overline{B})}
\end{aligned}
$$

- Self-triggers (A->A): 90% of words, self-trigger among top 6
  - "Burstiness" or overdispersion in language
  - Same-root triggers mentioned also

How to combine cues?

- Linear interpolation (k models, λ≥0, $\sum_i$λ=1)

$$P_{\text{COMBINED}}(w|h) \stackrel{\text{def}}{=} \sum_{i=1}^{k} \lambda_i P_i(w|h)$$

  - + Very general
  - + Easy (train λ with EM: latent "switching" variable)
  - – Doesn't best combine information optimally
    - Can't increase "sharpness"
  - – Too many submodels

Example
(Collins notes)

$$\begin{aligned}
&p(\text{model}|w_1, \dots w_{i-1}) = \\
&\lambda_1 \times q_{ML}(\text{model}|w_{i-2} = \text{any}, w_{i-1} = \text{statistical}) + \\
&\lambda_2 \times q_{ML}(\text{model}|w_{i-1} = \text{statistical}) + \\
&\lambda_3 \times q_{ML}(\text{model}) + \\
&\lambda_4 \times q_{ML}(\text{model}|w_{i-2} = \text{any}) + \\
&\lambda_5 \times q_{ML}(\text{model}|w_{i-1} \text{ is an adjective}) + \\
&\lambda_6 \times q_{ML}(\text{model}|w_{i-1} \text{ ends in ``ical''}) + \\
&\lambda_7 \times q_{ML}(\text{model}|\text{``model'' does not occur somewhere in } w_1, \dots w_{i-1}) + \\
&\lambda_8 \times q_{ML}(\text{model}|\text{``grammatical'' occurs somewhere in } w_1, \dots w_{i-1}) + \\
&\dots
\end{aligned}$$

MaxEnt / Log-Linear models

- **x**: input (all previous words)
- **y**: output (next word)
- **f(x,y)** => R^d feature function [[domain knowledge here!]]
- **v**: R^d parameter vector (weights)

$$p(y|x;v) = \frac{\exp\left(v \cdot f(x,y)\right)}{\sum_{y' \in \mathcal{Y}} \exp\left(v \cdot f(x,y')\right)}$$

Application to history-based LM:

$$\begin{aligned} P(w_1..w_T) &= \prod_t P(w_t \mid w_1..w_{t-1}) \\ &= \prod_t \frac{\exp(v \cdot f(w_1..w_{t-1}, w_t))}{\sum_{w \in \mathcal{V}} \exp(v \cdot f(w_1..w_{t-1}, w))} \end{aligned}$$

$$f_1(x,y) = \begin{cases} 1 & \text{if } y = \texttt{model} \\ 0 & \text{otherwise} \end{cases}$$

$$f_2(x,y) = \begin{cases} 1 & \text{if } y = \texttt{model} \text{ and } w_{i-1} = \texttt{statistical} \\ 0 & \text{otherwise} \end{cases}$$

$$f_3(x,y) = \begin{cases} 1 & \text{if } y = \texttt{model},\ w_{i-2} = \texttt{any},\ w_{i-1} = \texttt{statistical} \\ 0 & \text{otherwise} \end{cases}$$

$$f_4(x,y) = \begin{cases} 1 & \text{if } y = \texttt{model},\ w_{i-2} = \texttt{any} \\ 0 & \text{otherwise} \end{cases}$$

$$f_5(x,y) = \begin{cases} 1 & \text{if } y = \texttt{model},\ w_{i-1} \text{ is an adjective} \\ 0 & \text{otherwise} \end{cases}$$

$$f_6(x,y) = \begin{cases} 1 & \text{if } y = \texttt{model},\ w_{i-1} \text{ ends in ``ical''} \\ 0 & \text{otherwise} \end{cases}$$

$$f_7(x,y) = \begin{cases} 1 & \text{if } y = \texttt{model}, \text{ ``model'' is not in } w_1, \ldots w_{i-1} \\ 0 & \text{otherwise} \end{cases}$$

$$f_8(x,y) = \begin{cases} 1 & \text{if } y = \texttt{model}, \text{ ``grammatical'' is in } w_1, \ldots w_{i-1} \\ 0 & \text{otherwise} \end{cases}$$

Figure 1: Example features for the language modeling problem, where the input x is a sequence of words $w_1 w_2 \ldots w_{i-1}$, and the label y is a word.

Feature templates

- Generate large collection of features from single template
  - Not part of (standard) log-linear mathematics, but how you actually build these things
- e.g. Trigram feature template:
  For every (u,v,w) trigram in training data, create feature

$$f_{N(u,v,w)}(x, y) = \begin{cases} 1 & \textit{if } y = w,\ w_{i-2} = u,\ w_{i-1} = v \\ 0 & \textit{otherwise} \end{cases}$$

where $N(u, v, w)$ is a function that maps each trigram in the training data to a unique integer.

- Feature template for long-distance triggers?