Language modeling

n-grams, smoothing

Xiaojin Zhu

jerryzhu@cs.wisc.edu

Computer Science Department

University of Wisconsin, Madison

The need of language modeling

- estimate $P(s)$
- speech recognition, optical character recognition

$$s^* = \arg\max_s P(s|a) = \arg\max_s P(a|s)P(s)$$

- document classification

$$c^* = \arg\max_s P(c|d) = \arg\max_c P(d|c)P(c)$$

- the chain rule
- n-gram assumption
- trigram the best in performance and simplicity so far

Shakespeare unigram [JM p.203]

- To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have
- Every enter now severally so, let
- Hill he late speaks; or! a more to leg less first you enter
- Will rash been and by I the me loves gentle me not slavish page, the and hour; ill let
- Are where exeunt and sighs have rise excellency took of .. sleep knave we. near; vile like

Shakespeare bigram [JM p.203]

- What means, sir. I confess she? then all sorts, he is trim, captain.
- Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow.
- What we, hath got so she that I rest and sent to scold and nature bankrupt, nor the first gentleman?
- Enter Menenius, if it so many good direction found'st thou art a strong upon command of fear not a liberal largess given away, Falstaff! Exeunt
- Thou whoreson chops. Consumption catch your dearest friend, wekll, and I know where many mouths upon my undoing all but be, how soon, then; we'll execute upon my love's bonds and we do you will?

Shakespeare trigram [JM p.203]

- Sweet prince, Falstaff shall die. Harry of Monmouth's grave.
- This shall forbid it should be branded, if renown made it empty.
- What is't that cried?
- Indeed the duke; and had a very good friend.
- Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, 'tis done.

Shakespeare 4-gram [JM p.203]

- King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv'd in;
- Will you not tell me who I am?
- It cannot be but so.
- Indeed the short and the long. Marry, 'tis a noble Lepidus.
- They say all lovers swear more performance than they are wont to keep obliged faith unforfeited!

Perplexity

- average log likelihood of data D under model M

$$\frac{1}{n}\sum_i \log P_M(D_i)$$

- approximates the negative cross-entropy between P and P_M

$$-\sum_D P(D)\log P_M(D_i)$$

- perplexity: number of equally likely words that follows

$$2^{-\sum_D P(D)\log P_M(D_i)}$$

- perplexity (or likelihood) loosely correlated with accuracy

The need for smoothing

- big problem: data sparseness
- many smoothing ideas
  - additive
  - Good-Turing
  - Jelinek-Mercer interpolated
  - Katz
  - Whitten-Bell
  - Absolute discounting
  - Kneser-Ney

Good-Turing

The intuition:

- n_r: the number of word types with count r
- N: corpus length
- assume independence: taking a word out at a random position is the same as taking the last word out
- with probability $\frac{rn_r}{N}$ the word taken out occured r times in the corpus
- this is a word occured $r-1$ times in the remaining corpus
- obtain Good-Turing by reversing the time

Jelinek-Mercer Interpolation

The general problem:
Observing $x_1 \ldots x_n$, expert k predicts $p_k(x_1) \ldots p_k(x_n)$.
We linearly combine experts:

$$p(x) = \sum_{k=1}^{K} \lambda_k p_k(x)$$

How to determine the optimal weights λ?
Constraints:

$$\sum_{k=1}^{K} \lambda_k = 1$$

$$\lambda_k \geq 0, \forall k$$

The optimization problem

$$\max_{\lambda} \quad \prod_{i=1}^{n} p(x_i)$$

$$\text{s.t.} \quad \sum_{k=1}^{K} \lambda_k = 1$$

$$\lambda_k \geq 0, \forall k$$

where

$$\prod_{i=1}^{n} p(x_i) = \prod_{i=1}^{n} \sum_{k=1}^{K} \lambda_k p_k(x_i)$$

If you try to compute the gradient w.r.t. λ_k, you'll get a coupled expression – can't optimize it directly.

Expectation Maximization

Your first taste of EM, and variational optimization ...

$$\max_{\lambda} \prod_{i=1}^{n} p(x_i)$$

is equivalent to ($\log$ monotonic)

$$\max_{\lambda} \log \left(\prod_{i=1}^{n} p(x_i) \right) = \max_{\lambda} \sum_{i=1}^{n} \log p(x_i)$$

Why? We need the *concavity* of $\log$.

The variational trick

Introducing variational distributions $q_k(i)$, one distribution for each i

$$
\begin{aligned}
\sum_{i=1}^{n} \log p(x_i) &= \sum_{i=1}^{n} \log \sum_{k=1}^{K} \lambda_k p_k(x_i) \\
&= \sum_{i=1}^{n} \log \sum_{k=1}^{K} q_k(i) \left(\lambda_k p_k(x_i) / q_k(i)\right) \\
&\geq \sum_{i=1}^{n} \sum_{k=1}^{K} q_k(i) \log \left(\lambda_k p_k(x_i) / q_k(i)\right) \equiv L(\lambda, q)
\end{aligned}
$$

Jensen's inequality on concave $\log()$. Significance: λ no longer coupled.

Iterative optimization

- optimize q: form the Lagrangian for $\sum_k q_k(i) = 1$

$$\frac{\partial L(\lambda, q) - \sum_i \alpha_i \left(\sum_k q_k(i) - 1\right)}{\partial q_k(i)} = 0$$

$$q_k(i) \propto \lambda_k p_k(x_i)$$

- optimize λ: form the Lagrangian for $\sum_k \lambda_k = 1$

$$\frac{\partial L(\lambda, q) - \beta \left(\sum_k \lambda_k - 1\right)}{\partial \lambda_k} = 0$$

$$\lambda_k \propto \sum_i q_k(i)$$

Adaptation

- the true source $P(s)$ often is non-stationary
- LM built on one kind of text, used on another
- cache LM: a dynamic LM build on the current history

$$P(w|h) = \lambda P_{\text{static}}(w|h) + (1 - \lambda) P_{\text{cache}}(w|h)$$

Other techniques

- class-based LM, tree LM
- grammar-based LM
- Maximum entropy LM, whole sentence LM