CS 4650/7650, Lecture 5
Language Models

Jacob Eisenstein

September 3, 2013

1 Language models: what and why

We'll now talk about approaches for estimating the probability of a sequence of text, like a sentence.

What is the probabilities of the sentence: *Computers are useless, they can only give you answers.*

Relative frequency estimator:

$$\frac{\text{count}(\textit{Computers are useless, they can only give you answers})}{\text{count}(\text{all sentences ever spoken})} \tag{1}$$

We'll assume the probability of a sentence is equal to the probability of the words (in order): $P(S) = P(s_1, s_2, \ldots, s_N)$. We can apply the chain rule:

$$\begin{aligned} P(S) =& P(s_1, s_2, \ldots, s_N) \\ =& P(s_1)P(s_2|s_1)P(s_3|s_2, s_1)\ldots P(s_N|s_{N-1}, \ldots, s_1) \end{aligned}$$

Each element in the product is the probability of a word given all its predecessors. We can think of this as a *word prediction* task: *Computers are [BLANK].* The relative frequency estimator:

$$P(\textit{useless}|\textit{computers are}) = \frac{\text{count}(\textit{computers are useless})}{\sum_x \text{count}(\textit{computers are } x)} = \frac{\text{count}(\textit{computers are useless})}{\text{count}(\textit{computers are})}$$

Note that we haven't made any approximations yet, and we could apply the chain rule in reverse order, $P(S) = P(s_N)P(s_{N-1}|s_N)\ldots$, or any other order.

1.1 Is sentence probability a meaningful concept?

What are the probabilities of the two sentences,

- *Colorless green ideas sleep furiously*
- *Furiously sleep ideas green colorless*

Noam Chomsky used this pair of examples to argue that the probability of a sentence is a meaningless concept:

- Any English speaker can tell that the first sentence is grammatical but the second sentence is not.
- Yet neither sentence, nor their substrings, had ever appeared at the time that Chomsky wrote this article (they've appeared lots since then).
- Thus, he argued, empirical probabilities can't distinguish grammatical from ungrammatical sentences.

Pereira showed that by identifying *classes* of words (e.g., noun, verb, adjective, adverb), it's easy to show that the first sentence is more probable than the second. Time permitting, we will talk about class-based language models later.

1.2 Is sentence probability useful? The noisy channel model

Suppose we want to translate a sentence from Spanish:

- *El cafe negro me gusta mucho.*
- Word-for-word: *The coffee black me pleases much.*
- But a good language model of English will tell us:

$$P(\textit{The coffee black me pleases much}) < P(\textit{I like black coffee a lot}) \quad (2)$$

- How can we use this fact?

Warren Weaver on translation as decoding:

> When I look at an article in Russian, I say: 'This is really written in English, but it has been coded in some strange symbols. I will now proceed to decode.'

This motivates a generative model (like Naive Bayes!):

- English sentence E generated from language model $P(E)$
- Spanish sentence S generated from noisy channel $P(S|E)$

(picture)
Then the **decoding** problem is: $\max_E P(E|S) \propto P(E,S) = P(E)P(S|E)$

- The translation model is $P(S|E)$. This ensures the **adequacy** of the translation.
- The language model is $P(E)$. This ensures the **fluency** of the translation.

What else can we model with a noisy channel?

- Speech recognition (original = words; encoded = sound)
- Spelling correction (original = well-spelled text; encoded = text with spelling mistakes)
- Part of speech tagging (original = tags; encoded = words)
- Parsing (original = parse tree; encoded = words)
- ...

The noisy channel model allows us to decompose NLP systems into two parts:

- "Translation" model $P(S|E)$. We need labeled data for this.
- Language model $P(E)$. We need only *unlabeled* data for this.

Since there is always more unlabeled data, this means we can improve NLP systems just by improving $P(E)$.

2 Estimating language models

How big are language models? A back of the envelope calculation:

$$V = 10^4$$
$$M = 10$$
$$\text{size}[P(s_M|s_{M-1}, \ldots, s_1)] = (10^4)^{10} = 10^{40}$$

To handle a ten-word sentence with a vocabulary of 10^4, we would need to estimate and store a probability distribution over 10^{40} events. This won't work.

2.1 N-gram models

N-gram models make a simple approximation: condition on only the past $n-1$ words.

$$P(s_m|s_{m-1} \ldots s_1) \approx P(s_m|s_{m-1}, ..., s_{m-n+1})$$

In this model, we have to estimate and store the probability of only V^n events.

To compute the probability of a whole sentence, it's convenient to pad the beginning and end with special symbols $\langle$s$\rangle$and $\langle$/s$\rangle$. Then the bigram approximation to the probability of *I like black coffee* is:

$$P(I|\langle\text{s}\rangle)P(like|I)P(black|like)P(coffee|black)P(\langle/\text{s}\rangle|coffee) \quad (3)$$

Do you think this is a good approximation? Can you think of examples where a trigram model (n=2) fails?

Here are a few:

- *Gorillas always like to groom THEIR friends.*
- *The computer that's on the 3rd floor of our office building CRASHED.*

There is a bias variance tradeoff in language models, just like in classification. Can you see how it works?

- The higher n, the less bias in our model.
  We can see this by looking at example sentences generated from ngram models. They look more and more like real text.
- However, higher n means more variance.
  - Big n-grams have small counts, which are inaccurate and have more zeros.
  - We need *smoothing* to control this variance (by introducing bias).

We'll talk about smoothing in a minute, but first I need to tell you a little about how we can evaluate LMs.

2.2 Evaluating language models

- We prefer **extrinsic evaluation**: does the LM help the task (translation or whatever). But this is often hard to do.
- **Intrinsic evaluation** is task-neutral.

2.2.1 Perplexity

A popular intrinsic metric is perplexity (PP).

$$\begin{aligned} PP(s) =& P(s)^{-\frac{1}{N}} \\ =& \sqrt[N]{\frac{1}{P(s)}} \end{aligned}$$

- Assume a uniform, unigram model in which $P(s_i) = \frac{1}{V}$ for all V words in the vocabulary.

$$\begin{aligned} PP(s) &= \left[\left(\frac{1}{V}\right)^N\right]^{-\frac{1}{N}} \\ &= \left(\frac{1}{V}\right)^{-1} = V \end{aligned}$$

- We can think of perplexity as the *weighted branching factor* at each word in the sentence.
  - If we have solved the word prediction problem perfectly, $PP(S) = 1$, because there is only one possible choice.
  - If we have only a uniform model that assigns equal probability to every word, $PP(S) = V$.
  - Most models fall somewhere in between.
  - Here's how you remember: lower perplexity is better, because you are less perplexed.

2.2.2 Entropy

Perplexity is very closely related to the concept of entropy, the expected value of the information contained in each word.

$$H(P) = -\sum_s P(s) \log P(s) \tag{4}$$

The true entropy of English (or any real language) is unknown. Claude Shannon, one of the founders of information theory, wanted to compute upper and lower bounds. He would read passages of 15 characters to his wife, and ask her to guess the next character, recording the number of guesses it took for her to get the correct answer. As a fluent speaker of English, his wife could provide a reasonably tight bound on the number of guesses needed per character. **Question: is this an upper bound or a lower bound?**

Cross-entropy is a relationship between two probability distributions, the true one $P(s)$ and an estimate $Q(s)$.

$$\begin{aligned} H(p,q) &= -\sum_s p(S) \log q(S) \\ &= -\lim_{n\to\infty} \frac{1}{n} \log q(S) \approx -\frac{1}{N} \log q(S) \\ PP(S) &= 2^{-\frac{1}{N} \log q(s)} \end{aligned}$$

A good language model has low cross-entropy with $P(s)$, and thus low perplexity.

Aside : A current topic in psycholinguistics is the "constant entropy rate hypothesis," also called the "uniform information density hypothesis." The hypothesis is that "optimal speakers prefer choices that keep that amount of information conveyed per time uniform"[Jae07]. Some evidence:

- Speakers shorten predictable words, lengthen unpredictable ones
- High-entropy sentences take longer to read
- Good segmentations of text tend to have low entropy in each segment

2.3 Example

On 38M tokens of WSJ, $V \approx 20K$, Jurafsky and Martin (page 97) obtain these perplexities on a 1.5M token test set.

- Unigram: 962
- Bigram: 170
- Trigram: 109

Will it keep going down?
(see slides from [MS99])

3 Smoothing

We want to make estimate from sparse statistics.

$P(s|denied\ the)$

word	counts c	effective counts c^*
allegations	3	2.5
reports	2	1.5
claims	1	0.5
request	1	0.5
charges	0	$2/V_0$
benefits	0	$2/V_0$
...		

3.1 Laplace smoothing

Idea: just add "pseudo-counts"

$$P_{\text{Laplace}}(s_i) = \frac{c_i + \alpha}{N + V\alpha} \tag{5}$$

Anything that we add to the numerator (α) must also appear in the dominator ($V\alpha$). We can capture this with the concept of **effective counts**:

$$c_i^* = (c_i + \alpha)\frac{N}{N + V\alpha}$$

The **discount** for each word is:

$$d_i = \frac{c_i^*}{c_i} = \frac{(c_i + \alpha)}{c_i}\frac{N}{(N + \alpha)}$$

- How much probability mass should we assign to unseen words (in other words, how to choose α?)
- Should we discount all words by the same amount?
- **Good-Turing** smoothing provides one set of answers to these questions.

4 Backoff and interpolation

What order N-gram should we use?

- So far we have treated all N-grams the same.
- Suppose we are using trigams, and count(*to both sisters*) = 0
- The bigram count(*both sisters*) is still informative!
- How can we use it?

4.1 (Katz) Backoff

Idea: if you have trigrams, use trigrams. If not, use bigrams.

$$\hat{P}_{\text{Katz}} = \begin{cases} P^*(s_i|s_{i-1}, s_{i-2}), & \text{if count}(s_i, s_{i-1}, s_{i-2}) > 0 \\ \alpha(s_{i-1}, s_{i-2}) P_{\text{Katz}}(s_i|s_{i-1}), & \text{otherwise} \end{cases}$$

- $P*$ is the **discounted** probability, leaving some mass for unseen events.
- If we don't have bigrams, we backoff to unigrams.
- The values $\alpha(s_{i-1}, s_{i-2})$ are set to ensure that the probabilities sum to 1 for each context $\langle s_{i-1}, s_{i-2} \rangle$.

4.2 Interpolation

Instead of choosing one n-gram order, we can take the weighted average:

$$\begin{aligned} \hat{P}_{\text{Interpolation}}(s_i|s_{i-1}, s_{i-2}) = \lambda_1 P^*(s_i|s_{i-1}, s_{i-2}) \\ + \lambda_2 P^*(s_i|s_{i-1}) \\ + \lambda_3 P^*(s_i) \end{aligned}$$

- P^* is the maximum likelihood estimate (MLE)
- Constraint: $\sum_z \lambda_z = 1$
- We can tune λ on heldout data.
- Or we can use expectation maximization, $q_i(z)$ is the probability that word n was generated from a n-gram of order z.

We can add a latent variable z_i, indicating the order of the n-gram that generated word s_i. Generative story:

- For each word n
  - Draw $z_i \sim \text{Categorical}(\lambda)$
  - Draw $s_i \sim P(s_i | s_{i-1}, \ldots s_{i-z_i})$

EM algorithm:

- **E-step**: $q_i(z) = P(z|s_{1:n}) = \frac{P(s_n|s_{i-1},\ldots s_{i-z})}{\sum_{z'} P(s_i|s_{i-1},\ldots s_{i-z'})} P(z|\lambda)$
- **M-step**: $\lambda_z = \frac{E_q[\text{count}(Z=z)]}{\sum_{z'} E_q[\text{count}(Z=z')]}$

By running the EM algorithm, we can obtain a good estimate of λ, which we can then use for unseen data. The reading describes how we can condition λ on the identity of the words in the context, $\lambda(s_{n-1}, s_{n-2})$.

5 Kneser-Ney

I just bought a new titanium [BLANK]

- *Francisco*?
- *bicycle*?

Key idea: some words are more **versatile** than others.

- Suppose $P^*(\textit{Francisco}) > P^*(\textit{bicycle})$
- We would still guess that $P(\textit{titanium bicycle}) > P(\textit{titanium Francisco})$ because *bicycle* is a more versatile word.

We define the Kneser-Ney bigram probability as

$$P_{KN}(s_i|s_{i-1}) = \begin{cases} \frac{\text{count}(s_i,s_{i-1})-d}{\text{count}(s_{i-1})}, & \text{count}(s_i, s_{i-1}) > 0 \\ \alpha(s_i)P_{\text{continuation}}(s_i), & \text{otherwise} \end{cases}$$

$$P_{\text{continuation}}(s_i) = \frac{\#|s_{i-1} : \text{count}(s_n, s_{i-1}) > 0|}{\sum_{s'} \#|s_{i-1} : \text{count}(s', s_{i-1}) > 0|}$$

- We reserve probability mass using absolute discounting d.
- The *continuation probability* $P_{\text{continuation}}(s_i)$ is proportional to the number of observed contexts in which s_i appears.
- As in Katz backoff, $\alpha(s_{i-1})$ makes the probabilities sum to 1
- In practice, interpolation works a little better than backoff

$$P_{KN}(s_i|s_{i-1}) = \frac{\text{count}(s_i, s_{i-1}) - d}{\text{count}(s_{i-1})} + \beta(s_{i-1})P_{\text{continuation}}(s_i) \tag{6}$$

- This idea of counting contexts may seem heuristic, but actually there is a cool justification from Bayesian nonparametrics [Teh06].

6 Other types of Language Models

Interpolated Kneser-Ney is pretty close to state-of-the-art. But there are some interesting other types of language models, and they apply ideas that we have already learned.

6.1 Class-based language models

The reason we need smoothing is because the trigram probability model $P(s_i|s_{i-1}, s_{i-2})$ has a huge number of parameters. Let's simplify:

$$P_{\text{class}}(s_i|s_{i-1}) = \sum_z P(s_i|z;\theta)P(z|s_{i-1};\phi),$$

where $z \in [1, K], K \ll V$.

We get a bigram probability using $2VK$ parameters instead of V^2.

We could use EM to estimate θ and ϕ.

- E-step: update $q_i(z)$
- M-step: update θ and ϕ

But this is usually too slow, so there are approximate algorithms, like "exchange clustering" (Brown et al 1992), which assigns each word type to a class.

6.2 Discriminative language models

- Or we could just train a model to predict $P(s_i|s_{i-1}, s_{i-2}, \ldots)$ directly.
- We might be able to use arbitrary features of the history to model long-range dependencies.
- Algorithms such as perceptron and logistic regression have been considered.
- Currently, "neural probabilistic language models" are attracting a lot of interest. The log-bilinear model (Mnih and Teh 2012) looks like this:

$$P_\theta^h(w) = \frac{\exp(s_\theta(w,h))}{\sum_{w'} \exp(s_\theta(w',h))}$$
$$s_\theta(w,h) = \hat{\boldsymbol{q}}_h^\mathsf{T} \boldsymbol{q}_w + b_w,$$

  where h is the history context, $\hat{\boldsymbol{q}}_h$ is a latent description of the history, $\boldsymbol{q}_w$ is a latent description of the word, and b_w is an offset.

6.3 Other details

Datasets Dataset genre is really important. An LM learned from Shakespeare is a poor match for the Wall Street Journal (WSJ). An LM learned from the WSJ is a poor choice for predictive text entry in cellphones.

Unknown words Test data will usually have words which haven't been seen in the training data. What can we do about this?

- **Closed vocabulary**. Decide the dictionary before you start. Throw out everything else, or mark it with a special token $\langle$unk$\rangle$.
- **Open vocabulary**. Base the vocabulary on the training data. When you see a word for the first time in the training data, mark it with $\langle$unk$\rangle$, but add it to the dictionary.

References

[Jae07] T. Florian Jaeger. Optimal language production: Uniform information density, 2007.

[MS99] Christopher D. Manning and Hinrich Schütze. *Foundations of statistical natural language processing*. MIT Press, Cambridge, MA, USA, 1999.

[Teh06] Yee Whye Teh. A hierarchical bayesian language model based on pitman-yor processes. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 985–992. Association for Computational Linguistics, 2006.