CS838-1 Advanced NLP: Language Modeling

Xiaojin Zhu

2007

Send comments to jerryzhu@cs.wisc.edu

1 The Need for Finding Likely Sentences

Say you're designing a speech recognizer. Why do you prefer

s_1 =It's hard to recognize speech

over

s_2 =It's hard to wreck a nice beach?

The two sentences have the same acoustic signal. We must have some intrinsic preference over sentences. Language modeling tries to capture the notion that some sentences are more likely than others by *density estimation* $P(s)$. In particular, we want the model to satisfy $P(s_1) > P(s_2)$ for our examples above.

Similarly, if you are building an Optical Character Recognition (OCR) system, there are plenty of ambiguity in input (e.g., 1 vs. l, 0 vs. O). But we know that certain letter sequences are more likely than others.

Given input a (be it acoustic sound wave or digitized pixels), the recognition process can be defined as finding the sentence s that maximizes, among all sentences, the conditional probability

$$s^* = \arg\max_{s \in \mathcal{S}} P(s|a). \tag{1}$$

By Bayes rule, we have

$$P(s|a) = \frac{P(a|s)P(s)}{P(a)}. \tag{2}$$

Note $P(a)$ is a constant w.r.t. the maximization over s, so that the problem reduces to

$$s^* = \arg\max_{s \in \mathcal{S}} P(a|s)P(s). \tag{3}$$

The term $P(a|s)$ describes the probability of sound a that can be produced from sentence s: this is known as the *acoustic model* in speech recognition, and is often a Hidden Markov Model (HMM), as we will discuss later in the class. The term $P(s)$ is the *language model*, which we discuss now.

2 Unigram Language Model

In the following we assume words are the basic units, i.e. $s = w_1 \dots w_n \equiv w_{1:n}$.

A *unigram* language model makes the strong *independence assumption* that words are generated independently from a multinomial distribution θ (of dimension V, the size of the vocabulary). That is,

$$P(w_{1:n}|\theta) = \prod_{i=1}^{n} P(w_i|\theta) = \prod_{w=1}^{V} \theta_w^{c_w}, \tag{4}$$

where c_w is the count of word w in s. This is the multinomial distribution over c, except that we do not have the combinatorial coefficient, because we know the particular sequence.

You might be alarmed that this seems to be a particularly bad model of language, as it ignores word order and many other information about language! You are right – more on this later. But it is *useful*. By the way, the exact *chain rule* is

$$P(w_{1:n}|\theta) = P(w_1|\theta)P(w_2|w_1,\theta)P(w_3|w_{1:2},\theta)\dots P(w_n|w_{1:n-1},\theta), \tag{5}$$

and you can see the assumption unigram makes.

Technically this is not a complete definition of $P(s)$: there should be a distribution over sentence length $P(n)$, and $P(s) = P(n)P(w_{1:n})$.

The question is: where do we get θ? We estimate it from a corpus of English sentences. We need a vocabulary of V word types, and counts (redefining the symbol c) $c_{1:V}$ of each word type in the corpus. The counts are observed data, and we want to assess the unknown quantity θ, which is naturally the distribution $P(\theta|c_{1:V})$. By Bayes rule,

$$P(\theta|c_{1:V}) = \frac{P(c_{1:V}|\theta)P(\theta)}{P(c_{1:V})}. \tag{6}$$

where $P(c_{1:V}|\theta)$ is the multinomial distribution (4).

2.1 The Maximum Likelihood (ML) Estimate

For now let's take the prior $P(\theta)$ to be uniform. This gives the *maximum likelihood estimate* (ML or MLE) of θ:

$$\theta^{ML} = \arg \max_{\theta \in V-\text{simplex}} P(c_{1:V}|\theta) \tag{7}$$

$$= \arg\max_{\theta} \prod_{w=1}^{V} \theta_w^{c_w} \quad \text{multinomial definition} \tag{8}$$

$$= \arg\max_{\theta} \sum_{w=1}^{V} c_w \log \theta_w \quad \log() \text{ monotonic} \tag{9}$$

We are faced with the constrained optimization problem of finding $\theta_{1:V}$:

$$\max_{\theta_{1:V}} \quad \sum_{w=1}^{V} c_w \log \theta_w \tag{10}$$

$$\text{subject to} \quad \sum_{w=1}^{V} \theta_w = 1. \tag{11}$$

The general procedure to solve equality constrained optimization problems is the following: We introduce a scalar β called a *Lagrange multiplier* (one for each constraint), rewrite the equality constraint as $E(x) = 0$, and define a new Lagrangian function of the form $G(x, \beta) = F(x) - \beta E(x)$, where $F(x)$ is the original objective. Solve for the *unconstrained* optimization problem on G.

In our case, the Lagrangian is

$$\sum_{w=1}^{V} c_w \log \theta_w - \beta \left(\sum_{w=1}^{V} \theta_w - 1 \right) \tag{12}$$

After verifying that this is a concave function, we set the gradient (w.r.t. $\theta_{1:V}$ and β) to zero:

$$\frac{\partial}{\partial \theta_w} = \frac{c_w}{\theta_w} - \beta = 0 \tag{13}$$

$$\frac{\partial}{\partial \beta} = \sum_{w=1}^{V} \theta_w - 1 = 0, \tag{14}$$

which gives

$$\theta_w^{ML} = \frac{c_w}{\sum_{w=1}^{V} c_w} = \frac{c_w}{|C|}, \tag{15}$$

where $|C| = \sum_{w=1}^{V} c_w$ is the length of the corpus. It can be seen that the purpose of β is normalization. Therefore, in this case the MLE is simply the frequency estimate!

In practice, however, there is a huge problem with the MLE: if a word type w is in the vocabulary but not in the corpus (hence $c_w = 0$), the MLE is $\theta_w = 0$. Any new sentence with w in it will thus have zero probability! The problem is also known as data sparseness problem. This can be avoided by constructing the vocabulary from the corpus. But a more principled and satisfying solution is known as smoothing in the language modeling community, which is related to shrinkage in statistics, or, in certain cases, arises from an estimate other than MLE as we show next.

2.2 The Maximum A Posteriori (MAP) Estimate

In (6), the most likely θ given the corpus $c_{1:V}$ and a prior $P(\theta)$ is

$$\theta^{MAP} = \arg \max_{\theta \in V-\text{simplex}} P(c_{1:V}|\theta) P(\theta). \tag{16}$$

This is known as the *maximum a posteriori* (MAP) estimate. It is the same as the ML estimate if the prior is uniform, but otherwise is different in general.

One can choose any prior to encode domain knowledge. But the Dirichlet prior is particularly simple, since it is the conjugate prior of multinomial. A Dirichlet prior $P(\theta|\alpha)$ has positive *hyper-parameters* $\alpha_{1:V}$. It can be shown that the MAP estimate is

$$\theta_w^{MAP} = \frac{c_w + \alpha_w - 1}{|C| + \sum_{i=1}^{V}(\alpha_i - 1)}. \tag{17}$$

The hyperparameters thus act as "pseudo counts", i.e. $(\alpha_w - 1)$ can be viewed as the count of word w we collected before seeing the corpus. Clearly this avoids the zero probability problem (caution: α_w has to be no less than 1). A particular choice of hyperparameters $\alpha_i = 2,\ \forall i$ results in the so-called "add-one smoothing":

$$\theta_w^{MAP} = \frac{c_w + 1}{|C| + V}, \tag{18}$$

which is also known as Laplace smoothing, with which Laplace allegedly compute the probability that the Sun will rise again tomorrow. A simple variation brings "add-ϵ smoothing". These smoothing methods are not the best in terms of performance, but are often used because of their simplicity.

3 Evaluating Language Models

How do we know a language model (or a density estimator in general) is good or bad? The MLE is the 'best' (most likely) among all estimates on the training corpus $C = c_{1:V}$ by definition, and MAP seems to be even a bit inferior to MLE.

The answer lies in our interest in *future data*, not the corpus we trained on. Any model should give high probability to the kind of text we are interested in, but not seen in the training corpus. Therefore we assume we have a *test* corpus C', and we hope our θ gives high probability on any such C'.

Therefore test-set likelihood seems like a good quality measure for θ: $P(C'|\theta)$. But it is an extremely small number, and depends on the length of C'. A more useful measure is per-word test-set log likelihood:

$$\frac{1}{|C'|}\sum_{w=1}^{V} c'_w \log \theta_w. \tag{19}$$

If we call the actual distribution from which words are drawn $p(w)$, the above quantity is actually a stochastic approximation to

$$\sum_{w=1}^{V} p(w) \log \theta_w, \tag{20}$$

which becomes exact when $|C'| \to \infty$. We will come back to this in the next lecture.

A derived quantity, known as the *perplexity*, is widely used in language modeling:

$$PP(C';\theta) = 2^{-\frac{1}{|C'|}\sum_{w=1}^{V} c'_w \log_2 \theta_w} = P(C'|\theta)^{-\frac{1}{|C'|}}. \quad (21)$$

Note log base 2. Perplexity measures on average, how many 'equally likely' words we must choose from for each word position – the smaller the number, the more certain we are, and the better the model θ.

4 N-gram Language Models

We clearly view the order of words as important in the English language. One can partially incorporate such order into a language model by making weaker independence assumptions. In particular, for n-gram language model the assumption is

$$P(w_i|w_{1:i-1}) = P(w_i|w_{i-n+1:i-1}). \quad (22)$$

The conditioning part $w_{i-n+1:i-1}$ is called 'history', which has $n-1$ previous words. Compared to the chain rule (5), the n-gram language model dictates that

$$P(w_{1:n}|\theta) = \prod_{i=1}^{n} P(w_i|w_{i-n+1:i-1}, \theta). \quad (23)$$

Unigram is a special case when $n = 1$. Common names include bigram ($n = 2$) and trigram ($n = 3$). People use the special symbol `<s>` to denote start-of-sentence. N-grams normally does not run across sentences.

It is worth noting that the number of parameters in θ grows rapidly as $O(V^n)$. Another way to look at it is that for a given history h, $P(w|h)$ is a multinomial of size V, but there are V^{n-1} such possible histories, and θ consists of all these multinomials. For $V = 10,000$ which is typical, in theory there are 10,000-1 unigram parameters, 10^8 bigram parameters (compared to US population 3×10^8), and 10^{12} trigrams (about 150 trigrams per person in the world). In practice, the number of n-grams is bounded by corpus length. But it should be clear that the data sparseness problem is much more severe with a larger n, because the histories 'fragment' the corpus. Therefore smoothing is of great importance to n-gram language models.

We have seen MAP estimate as one way of smoothing. Other smoothing techniques interpolate MLE or MAP for $P(.|h)$ with lower order estimates $P(.|h')$. There are a large number of specific smoothing methods for language modeling, e.g., Good-Turing, Jelinek-Mercer interpolated, Katz, Whitten-Bell, Absolute discounting, and Kneser-Ney.

For document classification, people found that unigram is often sufficient. Trigrams are the standard for speech recognition. Google recently released "Web 1T 5-gram" which consists of n-gram counts (up to $n = 5$) generated from approximately 1 trillion word tokens of text from publicly accessible Web pages.

Beyond n-gram type language models, a variety of models have been proposed to try to "put language back into language model", including class-based LM, tree LM, grammar-based LM, Maximum entropy LM, whole sentence LM and so on. However, trigram language model is surprisingly hard to beat.

5 Sampling from a Distribution

We can generate sentences by sampling from the distribution $P(s)$ described by a language model. We write $s \sim P(s)$ if we select a sentence s at randomly according to the probability $P(s)$. For a unigram model, the sampling process is as simple as generating random numbers (word index) from the multinomial $P(w|\theta)$. For n-gram models, we sample from the conditional probability $P(w_i|w_{i-n+1:i-1}, \theta)$ where $w_{i-n+1:i-1}$ are the most recent $n-1$ words we just generated.

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

Bigram:

- 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?

Trigram:

- 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.

4-gram:

- 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!