Statistical Language Modeling and N-grams

Roni Rosenfeld

January 1997

Supporting literature:

- Bell, Cleary & Witten, "Text Compression".
- Witten & Bell, "The Zero Frequency Problem: Estimating the probabilities of Novel Events in Adaptive Text Compression", *IEEE-IT-37*, No. 4, July 1991.
- Slava M. Katz, "Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer", *IEEE-ASSP-35* no. 3, pp. 400–401, 1987.
- Frederick Jelinek, "Self-Organized Language Modeling for Speech Recognition", in *Readings in Speech Recognition*, Alex Waibel and Kai-Fu Lee (eds.), Morgan Kaufmann Publishers, 1990.

1 Statistical Language modeling

In the "noisy channel" formulations discussed in the previous lecture, an inevitable common subproblem was the estimation of the prior probabilities of various events in the language. For example, in speech recognition, given an acoustic signal A, we seek the linguistic unit L (say, a sentence) that is most likely to have given rise to it, Namely, we seek the L that maximizes $\Pr(L|A)$. Using Bayes Law:

$$\arg\max_L \Pr(L|A) = \arg\max_L \frac{\Pr(A|L) \cdot \Pr(L)}{\Pr(A)} = \arg\max_L \Pr(A|L) \cdot \Pr(L)$$

$\Pr(L)$ is the prior probability of L in the language. “Prior” here means “before we know A”. From here on, and for the rest of this course, we will concentrate on estimating $P(L)$ as best we can, using statistical means.

2 Measuring Model Quality

2.1 Perplexity

The ultimate measure of the quality of a language model is its impact on the application. But this is usually hard to measure, and impossible to use for direct optimization. A common alternative is to judge a LM M by how well it predicts some hitherto unseen text T. This can be measured by the log-likelihood of M generating T, or, equivalently, by the *cross-entropy* of the distribution function $P_T(\mathbf{x})$ of the text, with regard to the probability function $P_M(\mathbf{x})$ of the model. Intuitively speaking, cross entropy is the entropy of T as “perceived” by the model M. Put another way, it is the amount of surprise in seeing T when using the model M to anticipate events. In an equation:

$$H(P_T; P_M) = -\sum_{\mathbf{x}} P_T(\mathbf{x}) \cdot \log P_M(\mathbf{x}) \tag{1}$$

$H(P_T; P_M)$ has also been called the *logprob*. Often, the *perplexity* of the text with regard to the model is reported. It is defined as:

$$\mathrm{PP}_M(T) = 2^{H(P_T;P_M)} \tag{2}$$

Perplexity can be roughly interpreted as the geometric mean of the branchout factor of the language: a language with perplexity X has roughly the same difficulty as another language in which every word can be followed by X different words with equal probabilities.

Perplexity is a function of both the model and the text. When comparing perplexities, vocabularies must be the same, or else the smaller vocabulary will paradoxically bias the model towards lower perplexity (because it typically excludes rare words). Even if vocabularies are identical, different texts, with different out-of-vocabulary word rates, will render the comparison problematic.

2.2 Alternatives to Perplexity

Perplexity does not take into account acoustic confusability, and does not pay special attention to outliers (tail of the distribution), where most recognition errors occur. Lower perplexity does not always result in lower error rates, although it often does, especially when the reduction in perplexity is significant.

Several alternatives to perplexity have been suggested in the literature:

- *acoustic perplexity* is like perplexity but it takes into account acoustic confusability. However, a study showed that it is proportional to "regular" perplexity.

- *Speech Decoder Entropy* measures the combined information provided by the acoustic and linguistic models together. This is typically less than the sum of their individual contributions, since some of the information overlaps. Using a sample text, a study found that the acoustic and linguistic sources are slightly more complementary than if they were completely orthogonal.

- Recognition errors are strongly correlated with low-probability language model predictions ("surprises") . To capture this effect, one can measure the fraction of surprises, namely the percentage of the text which was predicted with a probability below a certain threshold.

3 Dealing with Data Sparseness

Can we use a large corpus (say, 100M sentences) to derive Maximum Likelihood estimates of $\Pr(L)$?

The space of all sentences L is of size $(V+1)^S$, where S is the largest number of words possible in a sentence (assume we impose such a limit). For a modest vocabulary of, say, 20,000 words, this is a huge space.

3.1 The Chain Rule

Let $L = w_1^n \stackrel{\text{def}}{=} w_1, w_2, \ldots w_n$, where the w_i's are the words that make up the sentence. One way to estimate $\Pr(L)$ is to use the chain rule:

$$\Pr(L) = \prod_{i=1}^{n} \Pr(w_i | w_1^{i-1})$$

Indeed, most statistical language models try to estimate expressions of the form $\Pr(w_i | w_1^{i-1})$. The latter is often written as $\Pr(w|h)$, where $h \stackrel{\text{def}}{=} w_1^{i-1}$ is called the *history*.

3.2 Partitioning the history space

The event space (h, w) is very large, and no reasonable amount of data would be sufficient to span it. Some simplifying assumptions must be made. Typically, these come in the form of *clustering*: a partition of the event space h is defined, and histories that fall into the same equivalence class are assumed to have a similar effect on the probability distribution of the next word w.

For example, in the N-gram model, the partition is based on the last $N - 1$ words of the history, and the underlying assumption is:

$$\Pr(w_i|w_1^{i-1}) = \Pr(w_i|w_{i-n+1}, \cdots, w_{i-1})$$

4 Word Ngram behavior

A few definitions:

a word ***type*****:** a sequence of letters.

a word ***token*****:** a particular occurrence of a word type in text.

So, for example, the single word type "THE" has many tokens in any English text.

On the other extreme, there are many word types that occur only once in a given text corpus (and many more that don't occur at all...).

Some graphs:

- Word Frequency List (head+tail) for broadcast news (BN) and and telephone conversations (SWB) (compare!)

- type vs. token graphs. Compare BN & SWB.
- type vs. token graphs. Compare unigrams, bigrams, trigrams.
- Zipf's distribution, in linear & log scale.

  Zipf's law: rank * frequency are constant.

  Zipf attributed to "Principle of least effort".

  George Miller showed it can be derived analytically.
- f-o-f graph for BN unigrams (Height measures frequency of words in that category. Area shows total tokens in that category).
- f-o-f graph for trigrams (compare to unigrams!)
- These 5 sample texts were generated using Ngram statistics collected from the Brown Corpus, using N=-1 (uniform), N=1 (unigram), N=2 (bigram), N=3 (trigram) and N=5 (pentagram). Can you guess which is which?

5 Smoothing

Consider the relatively simple case of constructing a unigram model. We only need to estimate the context-independent probability of $P(w_i)$, for every word w_i in the vocabulary. We do that by using Maximum Likelihood estimates (i.e., relative frequencies) based on a large corpus.

But we saw that as we collect more training data, we run into more and more words, and the process doesn't seem to end.

Even if we were given a fixed vocabulary, it is quite likely many words in it will not occur at all in a reasonable size training set. If we estimate these with their ML estimate, we get $P(w) = \frac{0}{N} = 0$.

If we later run into this word and predict it with zero probability, we get $\log(0) = -\infty$, and our perplexity measure blows up, regardless of how well we predict other words.

The moral: *never predict anything with zero probability*.

Fix: we must *smooth* our probability estimates.

Smoothing is the process of making the model more robust to phenomena that were not observed in the training data. It is usually accomplished by "loosening" the match between the model and the training data.

One simple smoothing algorithm:

1. Assign a *floor*. Make sure each word is predicted with a probability which is no less than some threshold, regardless of its frequency in the training data.

2. Renormalize all the probabilities to sum to 1.

What floor value is reasonable? Assume a vocabulary of size $V = 100,000$. If we used a floor or 10^{-6}, this may "cost" up to 0.1 of the total probability mass, which means all non-smoothed probabilities will have to be reduced proportionately. If we used 10^{-7} instead, we would affect at most 0.01 of the total mass.

How different are 10^{-6} and 10^{-7} ? Remember our figure of merit is based on $\log P$, not on P.

Can we do better than the floor method?

5.1 The Zero Frequency Problem

If we toss a die 10 times and observe that it never landed on face '4'. Will we conclude that $P_4 = 0$? If we use cross-entropy to measure the quality of our model, the answer is "of course not". But how should we estimate P_4 in that case?

5.1.1 Laplace's law of Succession

Consider an urn containing N balls, some black and some white. Suppose we draw n balls and observe that c of them are black. The ML estimate of Pr(black) is c/n. For large n and c, this is a fairly good estimate (we know it is unbiased, too). But what about small values of c, n? And what if $c = 0$?

Again, if cross-entropy is our figure-of-merit, we don't want to use the MLE in this case. What Laplace suggested in 1812 is:

$$\Pr(\text{black}) = \frac{c+1}{n+2}$$

This seems intuitively reasonable (think what happens when $n = 0$ and when $n = 1$ or $c = 0$ or 1). Better yet, it can be derived from Bayes theorem if we denote the total number of black balls by H_j, and assume the uniform prior:

$$\Pr(H_j) = \frac{1}{N+1} \quad (j = 0, 1, \ldots, N)$$

The "balls in the urn" example is really about a language with a vocaulary of 2 words ("black" and "white"). But this approach can be generalized to a language of any fixed and known vocabulary, say of size q. Assume we have a sample of N word tokens from that language, with C_i tokens from word type i ($\sum_i C_i = N$). If we assume a prior which assigns the same probability to any legal combination of the C_i's, then using Bayes Theorem, the posterior prediction given the sample $\{c_1, c_2, \ldots, c_q\}$ is:

$$\Pr(\text{word}i) = \frac{c_i + 1}{n + q}$$

This is sometimes called the "add 1" method.

5.1.2 When the vocabulary size is unknown

In this case, we cannot predict which specific new words we are going to run into. But at least we can try to estimate the probability that the next word we observe will be new. How do we estimate Pr(new)?

One possibility is:

$$\Pr(\text{new}) = \frac{1}{n + 1}$$

This means that estimates for all observe types are $\frac{c+i}{n+1}$.

Alternatively, one might consider any word type "new" until it occurred at least twice (makes sense if you want to exclude typos, for example), in whic hcase we get:

$$\Pr(\text{word}i) = \frac{c_i - 1}{n}$$

and therefore:

$$\Pr(\text{new}) = \frac{r\prime}{n}$$

Where $r\prime$ is the number of types that occurred more than once in the sample.

Yet a third possibility, which lies somewhere between the above two, is:

$$\Pr(\text{word}i) = \frac{c_i}{n+r}$$

and therefore:

$$\Pr(\text{new}) = \frac{r}{n+r}$$

An interpretation: Consider the first token of each word type to consist of two events: the event declaring "I am new", followed by the event defining that particular word type. Then there are really $n + r$ events, and the event "I am new" is estimated using ML.

(Challenge: we know that MLE of the expectation of a binomial distribution is unbiased. But this method still overestimates the probability of the next token being a new type. Why?)

5.1.3 A better estimate for Pr(new)

The following is a more sophisticated solution, proposed by I. I. Good in the context of estimating the population frequencies of animal species. His solution was in turn based on an observation by A. M. Turing.

The Good-Turing formula estimates that the probability of the next word token being novel is:

$$\Pr(\text{new}) = \frac{t_1}{n}$$

According to the same formula, the probability that the next word token will be one that has previously occurred exactly once is:

$$\Pr(\text{next token has occurred exactly once}) = \frac{2 \cdot t_2}{t_1}$$

and in general, the probability that the next word token will be one which has previously occurred exactly k times is:

$$\Pr(\text{next token has occurred exactly once}) = \frac{(k+1) \cdot t_{k+1}}{t_k}$$

There are many ways to derive the Good-Turing formula. Can you think of one?

Hint: Consider the frequency-of-frequencies (f-o-f) information. Let t_k be *the number of types which occurred exactly k times in the sample*. Specifically, consider the set of all words which occurred:

- not at all (zero times)
- exactly once each
- exactly twice each
- ·

Now consider the combined frequency of each of these sets, and consider what was the situation before the last word was added to the sample.

[Note that the MLE of each word is unbiased, as long as the word is chosen before we draw the sample. However, if we first draw the sample and then choose a word based on its frequency, then there is good reason to suspect that the estimate may be biased. Compare with the “regression to the mean” phenomenon.]

5.1.4 More sophisticated approaches

The Good-Turing formula can be viewed as an approximation (specifically, the first term in a Taylor expansion) of a yet more sophisticated estimate, one which is based on the assumption of a Poisson prior.

All these and other estimates can be compared with each other empirically and rated!