Lecture 6; Using Entropy for Evaluating and Comparing Probability Distributions

Readings: Jurafsky and Martin, section 6.7
Manning and Schutze, Section 2.2

So far we have used one method for evaluating probability distributions – based on the idea of maximizing the likelihood of the observed data. As we look at other application, we will need a richer set of tools. We might, for instance, have two probability functions and want to measure how "close" they are to each other. We also further develop the notion of measuring how well a probability distribution fits observed data and develop techniques that allow us to compare inherently different formulations (e.g., say a probability distribution base don words vs one based on letters). The key concept used here is called **entropy**, which is a measure of the inherent randomness in a probability distribution (or set of observed data). This is the key concept in a field of research called **information theory**. Today we'll look at the basic of Information Theory and see how the concepts can be applied to natural language processing.

1. Entropy: A Foundation of Information Theory

Information theory can be viewed as a way to measure and reason about the complexity of messages. In our case we will be interested in natural language messages, but information theory applies to any form of messages. Consider you are designing a system to transmit information as efficiently as possible, i.e., use the least number of bits. How many bits are needed clearly depends on what we know about the content. For instance, say we know a message is the answer to a yes/no question. We would just need 1 bit to encode this message.

If we needed to send a message that encoding what day of the week it is, we would need a message that can encode 7 values, which we can reduce to bits using the logarithm function. Thus we need $\log_2 7 = 2.8$ bits, which on current machines would mean we use 3 bits (000 – Monday, 001 – Tuesday, …, 110 – Sunday, 111- unused). But how do we know this? How do we know there isn't some better scheme that requires fewer bits on average?

To explore this question, consider another example. Say we want to encode the part of speech of a sequence of words in English using a tag set of size 40. It would appear that would need to use 6 bits ($\log_2 40 = 5.3$). But say we also know that 90% of the tags are one of four values, ART, P, N, V. I could design a new coding scheme that uses the first bit to indicate whether the tag is in this subset or not. If it is, we need just two bits more to encode the tag. If not, we need 6 more bits ($\log_2 36 = 5.16$). This coding scheme is shown in Table 1 for a probability distribution where ART, P, N, V are equally likely.

Tag	ART	P	N	V	ADJ	ADV	…
Coding	1 00	1 01	1 10	1 11	0 000000	0 000001	0 + 6 digit code
Prob.	.225	.225	.225	.225	.0028 (= .1/36)	.0028	.0028

Table 1: An efficient coding of parts of speech when four tags account for 90% of data

So the average length of message in this new coding scheme is coputed by observing that 90% of the data uses 3 bits, and the remaining 10% uses 7 bits.

$.9 * 3 + .1 * 7 = 2.7+.7 = 3.4$

almost half the size of the original scheme! Can we do better than this? Information theory provides an answer.

As mentioned before, **Entropy** is a measure of randomness in a probability distribution. A central theorem of information theory states that the entropy of p specifies the minimum number of bits needed to encode the values of a random variable X with probability function p

Defn of Entropy

Let X be a random variable, and p be the probability function such that $p(x_i) = P(X=x_i)$, then we define the entropy of X (or p) as

$H(X) = H(p) = - \sum_i p(x_i) \log(p(x_i))$ where x_i ranges over the vocabulary of X

Say in the above example, say we have a random variable TAG, which we know has the actual probability distribution shown in table 1.

The entropy of TAG would be

$H(TAG) = - (4 * (.225 * \log_2.225) + 36 * (.0028 * \log_2.0028))$

$= -(-1.04 + -.82 + -.85)$

$= 2.72$

Thus, we know we can in principle do better that the coding scheme designed above. While the entropy measure can tell us whether we have the best possible coding, it does not actually tell us how to find it, or whether it is actually physically possible to construct.

2. Entropy for more complex probability functions

Just like with probability functions, we can then define other forms of entropy. For joint distributions consisting of pairs of values from two or more distributions, we have Joint Entropy.

Defn of Joint Entropy

$H(<X,Y>) = - \sum_i \sum_j p(<x_i,y_i>)\log(p(<x_i,y_j>))$

Continuing the analogy, we also have conditional entropy, defined as follows:

Conditional Entropy

$H(Y \mid X) = \sum_i p(x_i) H(Y \mid X=x_i)$

$= - \sum_i \sum_j p(x_i) p(y_I \mid x_j) \log p(y_I \mid x_j)$

$= - \sum_i \sum_j p(<y_I, x_j>) \log p(y_I \mid x_j)$

Just like with probability functions, we have a **chain rule for entropy**, which sums the components rather than multiplying them because we are dealing with logarithms.

$$H(X,Y) = H(X) + H(Y \mid X)$$

Note that addition is used on the right hand side rather than multiplication because of the logarithm component of the entropy formula.

3. Evaluating probability models: Cross Entropy

Throughout this course, we will be trying to build good probability models to describe the behavior of languages (modeled as random variables producing the words, letters, etc). We sometimes will need ways to compare distributions to see how "close" they are. There is a variant of the entropy definition that allows us to compare two probability functions called **cross entropy** (of two probability functions p and m for a random variable X):

$$H(p, m) = -\sum_i p(x_i) \log(m(x_i))$$

Note that cross entropy is not a symmetric function, i.e., H(p,m) does not necessarily equal $H_X(m, p)$. Intuitively, we think of the first argument as the "target" probability distribution, and m is the estimated one we are trying to evaluate how well it "fits" the target. Note also that if p = m then the cross entropy is simply the entropy of X. It is a theorem that cross entropy is at its minimum when p = m. Thus we can use cross entropy to compare approximate models. The closer the cross entropy is to the entropy, the better m is an approximation of p.

Consider an example, the following table shows the values of a random variable X with its actual distribution p, and two approximations m1 and m2. m1 is simply the uniform distribution, which we will typically use when we have no information about the behavior of X. m2 is a distribution we might have guessed at after seeing some sequence of outputs of X in which A and D appeared fours times as frequently as B and C. Which one is better?

	A	B	C	D
p	.4	.1	.25	.25
m1	.25	.25	.25.	.25
M2	.4	.1	.1	.4

Table 2: Two approximations of probability distribution p

Using the distributions in table 3, the entropy of X (the entropy of p) is

$$H(p) = -\sum_i p(x_i) \log(p(x_i)) = 1.86$$

The cross-entropy for m1 is

$$H(p, m1) = -\sum_i p(x_i) \log(m1(x_i)) = 2$$

while the cross-entropy for m2 is

$$H(p, m2) = -\sum_i p(x_i) \log(m2(x_i)) = 2.02$$

This means that m1 is a slightly better model of X than m2. So we were unlucky with the sample of data that we observed!

4. Using Entropy Measures on Corpora

Most of the time in language applications, we actually don't know the target distribution, but have a corpus. So how can we tell whether one distribution is better than another given all we have is a corpora? Last week we used a measure that the probability function that maximized the likelihood of the corpus is the best. Here we develop an equivalent formulation, but in terms of entropy measures. Specifically, we introduce a notion of the **corpus cross entropy** (or **log probability**). Given a corpus C of size N consisting of tokens $c_1, \ldots, c_N$, the log probability of a probability function m on this corpus is defined as:

$$H_C(m) = - (1/N) * \sum_i \log(m(c_i))$$

Note that we are summing over *tokens* in the corpora, whereas previously we were summing over *types* in the vocabulary.

It can be proven that as N goes to infinity, the corpus cross entropy becomes the cross entropy measure for the true distribution that generated the corpus. Why is this so? It depends on the fact that the Maximum Likelihood Estimator goes to the true probability distribution as n goes to infinity. Thus as the corpus grows in size, the MLE estimate starts to generate good approximations of the true distribution. To see how this applies, we simply need to rewrite the definition of LP in terms of MLE estimates. Let's assume the vocabulary of the corpus is $V = \{v_1, \ldots, v_T)$. Consider one vocabulary item v_i, which occurs n_i times in the corpus. Thus the MLE estimate for v_i occurring is n_i/N. We could rewrite the sum in the LP definition so that instead of summing over all the corpus values c_i, we gather all the values of the same type together and sum over the vocabulary items:

$$H_C(m) = - (1/N) * \sum_i n_i * \log(m(v_i))$$

This is equivalent since there are n_i tokens in the corpus of type v_i. This can be rewritten as

$$= - \sum_i (n_i / N) * \log(m(v_i))$$

But this is just the cross entropy formula using p_{MLE} as the target probability function, i.e.,

$$= - \sum_i p_{MLE}(v_i) * \log(m(v_i))$$

which of course is the same value as in $H(p_{MLE}, m)$. Since the MLE estimate goes to the true distribution as the corpus size grows to infinity, the log probability thus goes to the cross entropy of m with the true probability distribution underlying the corpus.

Perplexity

Another measure used in the literature is equivalent to the corpus cross entropy and is called **perplexity**:

The corpus cross entropy is another way of evaluating how well a probability function describes a corpus. It is fairly simple to show that the H_C measure always agree with the maximum likelihood technique we developed earlier. Specifically, if a probability function p maximizes the likelihood of a corpus C, the p also has the minimum log probability measure. This follows from three basic properties of logarithms:

1. $\text{Log}(\prod_i x_i) = \sum_i \log(x_i)$
2. for all x and y, x > y implies log(x) > log(y)
3. for all x between 0 and 1, log(x) is negative

$$\text{Perplexity}(C, p) = 2^{Hc(p)}$$

With used for sociological and historical reasons, it add no new capabilities beyind using the entropy measures.

4. Mutual Information

Another common concept is called mutual information, which intuitively is the amount of additional information that we need to encode about X once we know Y. We define this by comparing the entropy of X with the entropy of X given Y:

Defn of Mutual Information

$$I(X; Y) = H(X) - H(X \mid Y)$$

We can show that mutual information is always non-negative – in other words, you can't lose information by knowing about another variable, and ranges from 0 to H(X). If X and Y are independent, then the mutual information measure is 0. If Y totally determined X, then the mutual information would be the entropy of X. (in other words, H(X) = I(X;X)). It is also easy to show that mutual information is a symmetric function, I(X;Y) = I(Y;X).

Consider the horse racing example from lecture two, where we have a joint distribution with the race results R and the weather W:

	R=win	R=lose
W=rain	.15	.15
W=shine	.05	..65

Table 3: The joint probability for P(R , W)

The marginal probability distribution for R has P(win) = .2 and P(lose) = .8. The conditional probability P(R|W) is show in the following table:

	R=win	R=lose
W=rain	.5	.5
W=shine	.07	.93

Table 4: the Conditional probability P(R | W)
With this,

$$H(R) = -(.2 * -1.32 + .8 * -.32) = .72$$

$$H(R|W) = -\sum_i \sum_j p(r_I, w_j) \log p(r_I \mid w_j)$$
$$= -(.15 * \log(.5) + .05 * \log(.07) + .15 * \log(.5) + .65 * \log(.93))$$
$$= .56$$

Thus, the mutual information

$$I(R, W) = .72 - .56 = .16$$

This indicates that R is affected by the value of the weather, which is intuitively the case. In other words, once we know W we can guess the value of R much more easily.

On the other hand, consider I(R; F) where F indicates whether Harry was fed or not less than two hours before the race. The joint distribution and conditional probabilities are shown in table 4 and 5. Note that feeding Harry seems to slightly improve his changes of winning.

	R=win	R=lose
F=fed	.21	.79
F=not fed	.19	.81

Table 5: The conditional probability for P(R | F)

	F=fed	F=not
R=win	.12	.45
R=lose	.08	.35

Table 6: The joint probability for P(R , F)

With this distribution, H(R) = .722 (as before), H(R | F) = .721, and thus

$$I(R; W) = .722 - .721 = .001$$

The mutual information is near zero, as expected since the variables are close to independent.

5. Entropy Rate and Sequences

So far we have only considered corpora that have little structure – i.e., they consist of a sequence of values. But when we talk about natural language, a corpora might be a sequences of values are themselves sequences. For example, in a corpus of sentences, each value is itself a sequence of words. Or a corpus of words can be considered as a set of sequences of letters.

Let's consider the latter case. Say we are interested in building a probabilistic of letter transitions in a language. One idea is to view the corpus simply as a sequence of letters, which we do by inserting a "blank" character (call it *) between words. Thus a corpus

THIS IS A TEXT

Would become a new corpus

* T H I S * I S * A * T E X T *

We could now use whatever estimation technique we think best to estimate the probability function underlying the stochastic process that generates a sequence of letters.

For instance, to estimate a bigram model we'd estimate P(T | *), P(H | T), and so on for all pairs of 27 letters (the alphabet plus *).

But there is another option. We might build a probabilistic model of the words themselves, and of course once we know the words, we know the sequence of letters. In this, we need to estimate a distribution P(IS | THIS), P(A | IS), and so on for all pairs of words (plus maybe * for the start symbol). If we want to know which of these is a better model, we'd need to be able to compare them in some way. But the straight forward entropy models would produce very different values – one measuring the entropy of one letter given the previous one, while the other measuring the entropy of one word given the previous word. To normalize this, we often introduce a notion of Entropy Rate.

Let us assume that we have a language L whose elements V (= v_1, …, v_N) are individual sequences from another vocabulary W of varying length l_1 … l_N. The **Entropy Rate** of L is defined as:

$$H_{rate(W)}(L) = - \sum_i 1/l_i * p(v_i) \log(p(v_i))$$

In other words, a word of length 3 such of ONE has a probability .01, the contribution of this word to the entropy calculation would be p(ONE) * log(p(ONE)) = -.33. The per-letter entropy rate would then be -.33/3 = -.11.