Statistical Methods in Computational Linguistics

Expectation, Information, and Entropy

Expectation

Car Prices

Let's reconsider the joint distribution of COLOR and MAKE in which the two variables are independent.

	Green	Red	Yellow	Total
VW	30	10	10	50
BMW	18	6	6	30
Jag	12	4	4	20
Total	60	20	20	100

Now let's consider prices:

	Green	Red	Yellow	Total
VW	10K	7K	5K	50
BMW	25K	20K	22K	30
Jag	48K	42K	40K	20
Total	60	20	20	100

What can we learn from this table besides the fact that you should immediately go out and paint your car green?

Expected Value

Suppose we want to get the average price of a car?

We dont just add the 9 prices together and divide by 9 because that ignores the relative frequencies of the prices.

A green VW price needs to be weighted more in the final average because green VWs are more common.

Weighting by relative frequency:

Make/Color	Price	Relative Frequency	Weighted Price

Green VW	10000	30/100=.3	3000
Red VW	7000	10/100=.1	700
Yellow VW	5000	10/100=.1	500
Green BMW	25000	18/100=.18	4500
Red BMW	20000	6/100=.06	1200
Yellow BMW	22000	6/100=.06	1320
Green Jag	48000	12/100=.12	5760
Red Jag	42000	4/100=.04	1640
Yellow Jag	40000	4/100=.04	1600
Expected Price			19720

This turns out just to be another way of taking an AVERAGE.

Make/Color	Price	Frequency	Costs
Green VW	10000	30	300000
Red VW	7000	10	70000
Yellow VW	5000	10	50000
Green BMW	25000	18	450000
Red BMW	20000	6	120000
Yellow BMW	22000	6	132000
Green Jag	48000	12	576000
Red Jag	42000	4	164000
Yellow Jag	40000	4	160000
Total Car Costs			1972000
Average			19720

General Expected Value

The general idea of expected value is that we have some **function** that assigns a number to every member of a sample space.

> Since we think the cost is independent of the color let's use MAKE as the sample space:
>
> MAKE = {VW,BMW,JAG}
>
> The number we want to assign to every member of the sample is the **cost**.

The expected value of the function is just the sum of its values for each member of the sample space **weighted by probability**

> We computed the **expected price** by adding all the prices **weighted by relative frequency**.
>
> General definition of expected value. (ps, pdf).

	Note that we defined random variables **formally** so that the notion of expectation always applies to them. A random variable is a function that returns a number. So for any random variable X defined over sample space $x_1, x_1, \ldots x_n$: $E(X) = p(x_1)X(x_1) + p(x_2)X(x_2) + \ldots p(x_n)X(x_n)$ $= \Sigma_{x_i} p(x_i) * X(x_i)$ But the COLOR, MAKE functions aren't very interesting random variables to take the expectation of. So we introduced the price function.

Information

We're interested in assigning a NUMBER to an event that characterizes the quantity of **information** it carries.

Next we'll be interested in the expected value of that number. **The expected quantity of information per event**. (Hint: that's the entropy)

Two Probabilistic Criteria	1. **Significance**: The more improbable an event is, the more information it carries. $P(x_1) > P(x_2) \implies I(x_1) < I(x_2)$ 2. **Additivity**: If x_1 and x_2 are independent events: $I(x_1x_2) = I(x_1) + I(x_2)$
Inverse Probabilities (Attempt 1)	Let's try: $I(x) = 1/p(x)$ Examples: $p(x_1) = 1/4 \implies I(x_1) = -\log(1/4) = 2$ $p(x_2) = 1/8 \implies I(x_2) = -\log(1/8) = 3$ Criterion 1, significance, is satisfied. Result: Contradiction of criterion 2, additivity. Consider two independent events such that $p(x_1) =1/4$ and $p(x_2)=1/8$. Then: 1. $p(x_1x_2)= 1/32 = 1/4 * 1/8$ 2. $I(x_1x_2)= 32$ But: 1. $I(x_1)= 2$ 2. $I(x_2)= 3$ 3. $I(x_1) + I(x_2) = 5$
Log probability	Observation: We know the probabilities of independent events multiply, but we want their combined information content to add:

(Attempt 2)

$I(p(x,y)) = I(p(x)) + I(p(y))$
$I(p(x) * p(y)) = I(p(x)) + I(p(y))$

One function that does something VERY like what we want is *log*:

$\log(x * y) = \log x + \log y$

Thus we have two ideas:

1. Using inverse probabilities deals with our significance criterion
2. Using logs deals with additivity,

Combining our two ideas we get:

$I(x) = \log(1/p(x)) = - \log p(x)$

Examples:

1. $p(x_1) = 1/4 \implies I(x_1) = -\log (1/4) = - (- 2) = 2 = \log (4)$
2. $p(x_2) = 1/8 \implies I(x_2) = - \log (1/8) = - (- 3) = 3 = \log (8)$

As desired, we satisfy significance.

Now we satisfy additivity:

1. $p(x_1x_2) = 1/32 \implies I(x_1x_2) = - \log (1/32) = - (- 5) = 5$
2. $I(x_1x_2) = I(x_1) + I(x_2)$
   $5 \qquad = 2 \quad + 3$

The **unit** is bits. Think of bits as counting *binary choices*. Taking probabilities out of it:

With n bits, you have enough to specify 2^n choices.

Example (binary numbers). 3 bits. 8 choices.

1. 000 = 0
2. 001 = 1
3. 010 = 2
4. 011 = 3
5. 100 = 4
6. 101 = 5
7. 110 = 6
8. 111 = 7

Information Quantity Defined

Assume a random variable X with probability mass function (pmf) p. For each x in the sample space of X:

$I(x) = - \log p(x)$

Surprisal

A lot of people prefer the term **surprisal** to information. That is, what

$- \log p(x)$

	is measuring is the amount of **surprise** generated by the event x. The smaller the probability of x, the bigger the **surprisal** is. It's helpful to think about it this way, particularly for linguistics examples. But I will continue to use the term *information*, out of pure love for tradition.

Entropy

We next define entropy as the expected information quantity.

Expected Information Quantity (Entropy)	I (Information quantity) is a function that returns a number for each member of a sample space of possible events. We can compute the expected value of I, which we call H: H: expected value of I. (ps, pdf). Let Ω for p be $x_1 \ldots x_n$. Then. $H(p) = p(x_1) * - \log p(x_1) + p(x) * - \log p(x_2) + \ldots p(x_n) * - \log p(x_n)$ $= - \sum_{i=1} p(x_i) * \log p(x_i)$ H(p) is called the **entropy** or expected information value of p. It is one many numbers we can associate with a probability distribution, such as mean, standard deviation, or variance. Each of these tells us something significant about the distribution. You should think of entropy as a measurement you can take, in particular a measurement you can take of a probability distribution. Like measurements of physical quantities, like relative velocity, temperature, and density, it is significant, but there are many factors that conspire together to give us that one number, so it can be hard to interpret. Always remember it's an average (expected value). Thus distributions that are very different in nature can have the same entropy.
Coin tossing	Suppose the probability of a head is 1/2. Then, $H(X) = [1/2 * - \log (1/2)] + [1/2 * - \log (1/2)]$ $H(X) = [1/2 * 1] + [1/2 * 1]$ $H(X) = 1$ The unit is **bits**. The expected information value for a uniform coin-flipping distribution is 1 bit. Suppose the probability of a head is 3/4. Then, $H(X) = [1/4 * - \log (1/4)] + [3/4 * - \log (3/4)]$ $H(X) = [1/4 * 2] + [3/4 * .415]$

$H(X) = .5 + .311 = .811$

The expected information value for this biased coin-flipping distribution is .811 bits. <-p> Suppose the probability of a head is 7/8. Then,

$H(X) = [1/8 * - \log (1/8)] + [7/8 * - \log (7/8)]$
$H(X) = [1/8 * 3] + [7/8 * .193]$
$H(X) = .375 + .144 = .519$

And finally, suppose we have the perfectly engineered cheater's coin such that the probability of a head is 1:

$H(X) = [0 * - \log (0)] + [1 * - \log (1)]$
$H(X) = [0 * - - \infty] + [1 * 0]$
$H(X) = [0 * \infty] + [1 * 0]$
$H(X) = 0 + 0 = 0$

Information theorists have defined $0 * \infty$ as 0 here [not generally true, but it is true that

$\mathrm{Lim}_{z \to 0}\ z * - \log z = 0,$]

so this is legitimate.]

Entropy as disorder

Consider the graph of all possible values of p(H)(pdf, ps).

> The more predictable the system becomes (the further p(H) moves from .5) the lower H is. **Entropy is a measure of disorder**.

General fact:

> For any random variable X, varying p and keeping Ω the same, the uniform distribution gives the highest value of H.

Entropy as Choice Number

Consider an 8-sided die:

Suppose the probability of each face is 1/8. Then,

$H(X) = 8([1/8 * - \log (1/8)])$
$H(X) = 8([1/8 * 3]$
$H(X) = 3$

Suppose the probability of one die is 1/4, and the others are all 3/28. Then,

$H(X) = [1/4 * - \log (1/4)] + 7([3/28 * - \log (3/28)])$
$H(X) = [1/4 * 2] + [7 * 3/28 * 3.22]$
$H(X) = .5 + 2.42 = 2.92$

Suppose the probability of one die is 7/8, and the others are all 1/56.

$H(X) = [1/8 * - \log (1/8)] + 7([1/56 * - \log (1/56)])$
$H(X) = [1/8 * 3] + [1/8 * 5.81]$
$H(X) = .375 + .726 = 1.101$

Same trend as the examples with heads, but the numbers are staying higher.

For a sample space of size m (=2^n), the entropy of a uniform distribution is log m (= n = log 2^n).

Reviewing:

Size of Ω	H(p); p uniform
2	1
4	2
8	3

So H(p) = log n for the uniform distribution over a sample space of size n.

in general, entropy goes up *slowly* as the number of events goes up, as the number of binary choices (bits) required to determine an event goes up.

Optimal Encoding

Consider the following horse race:

Distribution X for horses

H1	1/2
H2	1/4
H3	1/8
H4	1/16
H5	1/64
H6	1/64
H7	1/64
H8	1/64

H(X) = -1/2 log 1/2 - 1/4 log 1/4 - 1/16 log 1/16 - 4 (1/64 log -64)
= 2 bits

Encoding winners (first try)

Expected length of encoding

Horse	**Prob**	**Encoding**	**Expected Length Term**
H1	1/2	000	**3/2**
H2	1/4	001	**3/4**
H3	1/8	010	**3/8**
H4	1/16	011	**3/16**
H5	1/64	100	**3/64**
H6	1/64	101	**3/64**
H7	1/64	110	**3/64**

H8	1/64	111	**3/64**
Average length			3.0

Under this encoding, the length of the average message will be 3 bits. since

```
((3.0/64) * 4) + (3.0/16) + (3.0/8) + (3.0/4) + (3.0/2) = 3.0
```

Encoding winners (Optimal way)

Expected Length

Horse	**Prob**	**Encoding**	**Expected Length Term**
H1	1/2	0	**1/2**
H2	1/4	10	**1/2**
H3	1/8	110	**3/8**
H4	1/16	1110	**1/4**
H5	1/64	111100	**6/64**
H6	1/64	111101	**6/64**
H7	1/64	111110	**6/64**
H8	1/64	111111	**6/64**
Average length			2.0

Since:

```
((6.0/64) * 4) + (1.0/4) + (3.0/8) + (1.0/2) + (1.0/2) = 2.0
```

Shannon Weaver Encoding Theorem: A code averaging H(p) bits per horse always exists and is always the optimal encoding.

Corollary: For a uniform distribution, we can do no better than the code we started with, averaging 3 bits per horse.

Twenty Questions

Another way to look at entropy is that it measures the average number of yes/no questions it takes to determine an outcome, using an optimal questioning strategy:

Consider a variant of 20 questions in which your pay-off halves after each question. To maximize average payoff you want a strategy that guarantees you the least average number of questions.

1. Is it H1? (P = 1/2) [1 question]
2. Is it H2? (p = 1/4) [2 questions]
3. Is it H3? (p = 1/8) [3 questions]
4. Is it H4? (p = 1/16) [4 questions]
5. Is it H5 or H6? (p = 1/64) [6 questions]
6. Is it H7? (p = 1/64) [6 questions]

	It takes an average of 3 questions, given this strategy. A uniform strategy (Is it H1?.... Is it H7?) is once again much worse.
Entropy of Joint Distribution	Definition of H(X,Y). (ps, pdf).
Entropy of Conditional Distribution	Definition of H(XlY). (ps, pdf).
Mutual Information	Suppose we want to define the amount that knowing an event y reduces the surprisal of knowing x. The probability of y when x is known is P(xly). The reduction in the surprisal for x -- written I(x;y) -- is then: (1) $I(x;y) = I(x) - I(x\|y)$ $= - \log P(x) - (- \log P(x \mid y))$ $= - \log P(x) + \log P(x \mid y)$ $= \log (1/P(x)) + \log P(x \mid y)$ $= \log (1/P(x)) + \log P(x,y)/P(y)$ (2a) $= \log (P(x,y)/(P(x)*P(y)))$ (2b) $= \log (P(x \mid y)/P(x))$ I(x;y) is called the **Pointwise Mutual Information of x and y**. An interesting property, not obvious from (1), but clear from (2a), is that it is symmetric: (3) $I(x;y) = I(y;x)$ Pointwise mutual information, like surprisal, is a numerical function of events in a probability distribution, the distribution P(x,y). We can therefore take its expectation with respect to that distribution. This gives: (4) $I(X;Y) = \sum_{x,y} p(x,y) \log [p(x\|y) \div p(x)]$ It can be shown that under this definition: (5) $I(X;Y) = H(X) - H(X \mid Y)$ Summarizing the intuitions about (4): 1. The expected value of the ratio of p(xly) to p(x). 2. When x and y are independent, p(xly) = p(x). So $\log [p(x\|y) \div p(x)] = \log [1] = 0$ 3. Average amount of reduction in uncertainty, weighted by p(x,y) [from 1]
Interdisiplinary Nature of Information Theory	Fields where the notion of entropy (or something like it) plays a role: 1. Communication Theory Optimal encoding 2. Computer Science Kolomogorov Complexity: length of the shortest binary program for describing/computing a string 3. Physics

	Thermodynamics: 2nd law. Entropy always increases. The universe tends toward heat death (= the uniform distribution) 4. Probability Theory 5. Statistics 6. Economics 1. Portfolio Theory 2. Kelly Gambling
Cross entropy (Intuition)	Measure average surprisal assigned to a corpus by our model. 1. Sum up the surprisal of each new word, according to our model: $- \log m(w_1)$ $- \log m(w_2 \mid w_1) +$ $- \log m(w_3 \mid w_1 w_2)+$ … $- \log m(w_n \mid w_1, w_2, w_3, \ldots w_{n-1})$ 2. $= - \log(m(w_1) * m(w_2 \mid w_1) \ldots * m(w_n \mid w_1, \ldots w_{n-1}))$ 3. $= - \log m(w_1, \ldots w_n)$ [chain rule] 4. Divide by n to make this a per word measure: an average. 5. The smaller this number the better the model (the less surprised by the corpus). The bigger $m(w_1, \ldots w_n)$, the smaller this number. Of course our model will always treat some maximum chunk-size, say, n. And a good test corpus should naturally be much bigger than n. Let $W_{1,n}$ be a corpus of n-word chunks we write in lower-case, say $w_{1,n}$. We get the following definition of cross-entropy: $- 1/n \sum_{w_{1,n} \in W_{1,n}} \log m(w_{1,n})$ Now the summation here is just the log of the probability the model assigns to the whole corpus (the log of the product of the probabilities of each n-word chunk token). So retooling n to now be the length of the corpus, this is usually rewritten: $- 1/n * \log m(w_{1,n})$
Cross Entropy Definition	It is an interesting fact that this very simple definition of cross-entropy as per-word surprisal gives exactly the same results as a more mathematically motivated definition, given some natural assumptions. Cross-Entropy

$$H(p,m) = - \sum_x p(x) \log m(x)$$

NB: This is always HIGHER than:

$$H(p) = - \sum_x p(x) \log p(x)$$

(discussion below).

(Per Word) Cross-Entropy of Model for a corpus of size n

$$H(p,m) = - 1/n \sum_{i=1} p(w_i) \log m(w_i)$$

(Per Word) Cross-Entropy of Model for the Language

$$H(p,m) = - \text{Lim}_{n\to\infty} 1/n \sum_{i=1} p(w_i) \log m(w_i)$$

By a magical theorem, this turns into the definition of cross-entropy as per-word surprisal in a corpus, if we use a large enough corpus.

Suppose we have a lot samples of text of size n which we know to be independent of each other, large enough so that for all strings of length n, all $w_{1,n}$, we have a **valid** estimate of the probability of $w_{1,n}$. Then we can show:

(1a) H(p,m)	$= - \text{Lim}_{n\to\infty} 1/n \sum_{w_{1,n}} p(w_{1,n}) * \log m(w_1, w_n)$	Computed using **true** model of English
(1b)	$= - \text{Lim}_{n\to\infty} 1/n \log m(w_1 \ldots w_n)$	Computed using **corpus** of English

The virtue of formulation (1b) is that it can be used without a **true** model of English.

The idea is that a large enough corpus will have frequencies that faithfully reflect the true model of English.

Following Charniak (1997), 2.7, consider an example.

Suppose for the sake of concreteness that all 20 word sequences of English were independent of one another and suppose m was a complete model of such sequences. Assume there is one 20-word sequence M1 whose probability according to the true model of English is .05. Then (1a) tells us the contribution M1 makes to the overall cross-entropy of m is:

(2)	1/20 * 5/100	*	log m(M1)	according to (1a)
	weighting		M1's contrib to model	

Now what does (1b) tell us? For that we need a corpus. Suppose again for concreteness, a 2000 word corpus (100 20 word sequences) was sufficient to adequately represent the distribution of all possible 20 -word sequences. If the corpus gives a representative sampling of M1, there will be 5 occurrences of M1 in the 100 sequence corpus. So according to (1b) the overall contribution of M1 to the cross-entropy of m will be:

(3)	5 * 1/2000	*	log m(M1)	according to (1b)
	weighting		M1's contrib to model	

Now the weightings in (2) and (3) are equal, and therefore (2) and (3) are equal.

And in general for any message, they will be equal, because:

(4) Count(M1)/SequenceCount = Prob(M1)

for any representative corpus. In our example:

5/100 = Prob(M1)

If we now divide both sides of (4) by the length of M1 we get:

$$\underbrace{\text{(5) Count(M1)/(SequenceCount*Len(M1))}}_{(3)} = \underbrace{\text{Prob(M1) * (1/Len(M1))}}_{(2)}$$

The LHS is the weighting of M1's contribution according to (3) and the RHS is the weighting according to (2). They are equal.

Perplexity Perplexity is 2 to the cross-entropy:

$$\text{Perplexity(p,m)} = 2^{H(p,m)}$$

If H(p,m) = 3, Perp(p,m)= 8.

If H(p,m) = 4, Perp(p,m)= 16

Speech recognition tradition uses perplexity.

Perplexity is the measure of the average number of choices a random variable has to make. For a uniform distribution over horses, it's 8. For a skewed distribution like the horse race above, it's 4.

Cross-entropy: Intuitions

For any model m differing from the true model p, cross-entropy(p,m) > entropy(p).

Now in general any incorrect probability model m is going to do some underestimating of the probabilities in the event space of p (adding to H(p)), as well as some overestimating (subtracting from H(p)). But it turns out these two kinds of errors don't balance out.

Suppose we underestimate:

Model		**Truth**
m(x)	<	p(x)
log m(x)	<	log p(x)
p(x) * log m(x)	<	p(x) * log p(x)
- p(x) * log m(x)	>	- p(x) * log p(x)

So in general underestimating the probability of an event pushes the cross-entropy upward.

Symmetrically, overestimating pushes it downward:

Model		**Truth**
m(x)	>	p(x)
log m(x)	>	log p(x)
p(x) * log m(x)	>	p(x) * log p(x)
- p(x) * log m(x)	<	- p(x) * log p(x)

But the effect of underestimating always outweighs the effect of overestimating. How does this work? This is a property of the log function.

Consider a subdistribution of p that only contains 1/2 of the prob mass.

Entropy of a subdistribution

Event (x)	**p(x)**	**Info (- log p(x))**	**$H_p(e)$**
e_1	0.25	2	0.5
e_2	0.25	2	0.5

entropy of subdist	1.0

Now consider a misguided model m:

	p	m	H(p)	H(p,m)	Δ_p	Δ_H
e_1	0.2500	0.1250	0.5000	0.7500	−0.1250	+0.2500
e_2	0.2500	0.3750	0.5000	0.3538	+0.1250	−0.1462
Totals:	0.5000	0.5000	1.0000	1.1038	+0.0000	+0.1038

So underestimating $p(e_1)$ made $H_{p,m}(e_1)$ exceed $H(e_1)$ (.5199 > .3466), and the overestimation of e_2 wasn't enough to make up for it.

How much do we have to overestimate e_2 to make up for underestimating e_1?

	p	m	H(p)	H(p,m)	Δ_p	Δ_H
e_1	0.2500	0.1250	0.5000	0.7500	−0.1250	+0.2500
e_2	0.2500	0.5000	0.5000	0.2500	+0.2500	−0.2500
Totals:	0.5000	0.6250	1.0000	1.0000	+0.1250	+0.0000

So now the Δ_H balances, but the Δ_p does not. But this means the rest of m (.375) is going to underestimate the probability of the rest of p (.5), which will make the cross-entropy for that set of events exceed the entropy.

Varying the amount of the overestimations and breaking them up always leads to a net positive Δ H. For example:

	p	m	H(p)	H(p,m)	&Delta$_p$	Δ_H
	0.0500	0.0600	0.2161	0.2029	+0.0100	−0.0132
	0.0400	0.0500	0.1858	0.1729	+0.0100	−0.0129
	0.0300	0.0400	0.1518	0.1393	+0.0100	−0.0125
	0.0200	0.0300	0.1129	0.1012	+0.0100	−0.0117
	0.0100	0.0200	0.0664	0.0564	+0.0100	−0.0100
	0.0600	0.0500	0.2435	0.2593	−0.0100	+0.0158
	0.0500	0.0400	0.2161	0.2322	−0.0100	+0.0161
	0.0400	0.0300	0.1858	0.2024	−0.0100	+0.0166
	0.0300	0.0200	0.1518	0.1693	−0.0100	+0.0175
	0.0200	0.0100	0.1129	0.1329	−0.0100	+0.0200
Totals:	0.3500	0.3500	1.6430	1.6688	−0.0000	+0.0258

Here the probabilities balance, but the entropy contributions do not, and the cross-entropy once again is greater.

In general it takes more than p+Δ probability mass to make up for an underestimation of Δ.

	To do your own cross-entropy trials, try out cross_entropy computing mdule, with some help from the documentation.