Statistical NLP: course notes

Çağrı Çöltekin — SfS / University of Tübingen

2020-05-27

These notes are prepared for the class *Statistical Natural Language Processing* taught in Seminar für Sprachwissenschaft, University of Tübingen.

3 *Information theory*

The field of information theory is concerned with measurement, storage and transmission of information. It has its roots in communication theory, but it is used in many different fields including machine learning and natural language processing. In this chapter, we briefly introduce some of the main ideas, and discuss a few important information theoretic measures that will be used in the rest of the course.

3.1 The noisy channel model

A basic motivation in information theory is to characterize the communication over a noisy channel. In a *noisy channel model*, as the one depicted in Figure 3.1, the sender encodes the given message, and sends it through the channel, but the receiver receives a possibly corrupted version of the coded message.

The task of the decoder is to recover the original message, even if there are some errors introduced in the noisy channel. There are two competing objectives within the noisy channel model. First, we want to use codes that make use of the *channel capacity* as *efficiently* as possible. We want coding schemes that result in short coded messages, to transmit or store. This is where the strong connection between the information theory and compression comes into the picture. Second, we want to be able to detect and correct the errors introduced by the noisy channel. An obvious way to detect and correct errors is to send multiple copies of the code. As we introduce more redundant copies, it is more likely to recover the original message. However, replication also wastes the bandwidth of the channel. Coding information efficiently, while allowing error detection and error correction is the fundamental motivation in information theory. The example code in Figure 3.1, is simply the ASCII code followed by an odd parity bit, which means the last bit is set to 1 if the number of 1s in the code is an odd number, otherwise, the last bit is set to 0. Note that with the given code at hand, the decoder can detect the error.[1] Here, we will not discuss error-correcting codes, neither most of the other fascinating topics in information theory. Interested readers are referred to the textbooks on information theory, such as MacKay (2003).

Clearly, the noisy channel model is useful in the study of computer networks. However, it has many other uses. Note that the channel does not have to be a network connection. For example, the model fits equally well to storing data to a permanent storage. Per-

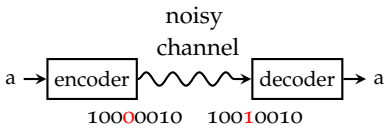

Figure 3.1: Schematic description of the noisy-channel model. The encoder codes the message and sends through a noisy channel to the decoder. The encoded message may possibly be corrupted during the transmission. The decoder's task is to reconstruct the original message, despite the potential noise introduced.

[1] Can the decoder *correct* the error in Figure 3.1?

manent storage systems (e.g., hard disks) are fairly accurate, however, they are not error free. As a result, error resilience and efficiency is a concern here too. The encoder encodes the information, and sends it to the disk, the decoder reads code (possibly after a long time) from the disk and has to make sure that the information decoded was not corrupted.

Beyond those obvious extensions, the noisy channel model found its use in many other applications. For an example close to home, we often model *speech recognition* with a noisy channel model. The information here is a linguistic message, e.g., a sentence. The speaker codes this information as an acoustic signal, and the recognizer's task is to decode the sentence from the acoustic signal (code). Another common use for the noisy channel model in NLP is in machine translation, which we will return later. We will not further discuss the applications of the noisy channel model here. However, we introduce some of the concepts, particularly some measures, that have very frequent uses in machine learning and NLP.

3.2 *Entropy and information*

In information theory, *entropy* is a measure of uncertainty. The measure is analogous to 'physical' entropy measure in statistical thermodynamics, but measures the uncertainty of an information system rather than a physical system. In ambiguous contexts, it is also called *information entropy* or *Shannon entropy* after Claude Shannon, the inventor of the measure and the founder of the field. Entropy and information are tightly connected concepts. More concretely, information in a message (e.g., in the noisy channel model described above) is the reduction of entropy after receiving the message.

Before introducing entropy properly, we will first introduce a related measure *surprisal*, which is also called *self information* or *information content*. The information theoretic measure of surprisal of an event x is defined as

$$\log \frac{1}{P(x)} = -\log P(x). \tag{3.1}$$

If the probability of the event is 1 (event occurs with certainty) the surprisal will be 0. For events with decreasing probabilities, we will get higher values for surprisal. The value of surprisal will approach to ∞ while the probability of the event approaches to 0.

The base of the logarithm in Equation 3.1 is not very important, since the logarithms in a different base can be obtained by multiplication with a constant (with a linear transformation). Most common choices include base 2 logarithms, which results in surprisal (or information) measured in *bit*s. If we use natural logarithm (with base e, Euler's number), than the unit is called *nat*. In this course we will always use base-2 logarithms, and measure the information in bits.

The same quantity having names 'surprisal' and 'self information' may not sound right very intuitive first sight. The intuition here is that we learn more from low-probability events. Low probability

events are surprising, but also have more information content. There is nothing surprising with a weather report that tells it will rain in a very rainy country. It also does not have much information content, we can already predict it. But if it predicts a sunny day in the middle of a rainy season, then it is surprising as well as news-worthy, it contains more information.

Entropy of a system is the average surprisal. We define entropy, H, of a random variable X as

$$H(X) = -\sum_{x \in X} P(x) \log P(x) \tag{3.2}$$

where, x ranges over all values of X. The above definition is for discrete variables, which is much more common in NLP. The notion of entropy can be extended to continuous random variables, by replacing the sum with an integral and $P(x)$ with the appropriate probability density function. The resulting definition (for continuous random variables) is called *differential entropy*.

To get a sense of what entropy does, consider the 'guess the number' game, where the first player picks a number between 1 and a larger number M, (say 32), and the task of the second player is to guess the number. After every guess, the first player tells whether the guess was larger or smaller than the number the second player predicted. Assuming the first player picks the number completely randomly (samples from the uniform distribution), and if the second player follows the optimum strategy (binary search), the first player will need to make $\log_2 M$ guesses at most.[2] Which is exactly what you will find if you use Equation 3.2, to calculate the entropy of this system. For $M = 32$,

$$H = -\sum_{1 \leqslant x \leqslant 32} \frac{1}{32} \log_2 \frac{1}{32} = -32 \frac{1}{32} \log_2 \frac{1}{32} = -\log_2 \frac{1}{32} = 5.$$

[2] If the number is not picked randomly, but follows a known non-uniform distribution, a different strategy may yield a faster solution. In fact, uniform distribution is the distribution with the highest entropy. If the first player samples the numbers from any other (known) distribution, there will always be a strategy that predicts the target number with fewer guesses on average.

Note that since the numbers are equally likely, probabilities for each number is the same (1/32). As a result, at the beginning we have 5 bits of entropy. You should also see that we reduce the entropy by 1 bit for every guess (unless we guess the correct number). Hence, information we gain with every guess is 1 bit.

It is important to realize that the entropy, or the information, increase or decrease proportional to the logarithm of the states (the numbers). If we double the range of numbers to pick from, the entropy will not double, but only increase with one bit. This logarithmic relationship is convenient, since states of many systems we are interested in grow exponentially.

To get a sense of why logarithm is a good idea, consider storage or transmission of a 8-letter alphabet on a binary medium. We can use as many bits as the number of letters (setting one of the bits to 1, and the rest to 0, similar to the one-hot representation discussed in Section 2.2), but this is wasteful. The optimum coding would not require 8 bits. We can easily represent 8 letters letters with 3 bits, which turns out to be exaclty $\log_2 8$. An example coding of 8-letter

alphabet is shown in Table 3.1. Note that if we double the number of letters, we would not need to double the number of bits used, all we need to do is add one more bit. Hence, the number of bits needed to represent M letters is $\log_2 M$. This is optimum if the letters we want to transmit are distributed uniformly. We will soon see that we can do better, if we have more structure in the distribution of the letters.

For simplicity, we assumed uniform distribution in the discussion/demonstration above. What if the distribution of the random variable is not uniform? To answer this question, consider a Bernoulli trial, for example, a coin-toss experiment or picking letters from a two-letter alphabet. If the letters were equally probable as in our earlier example, the entropy would be,

$$H = -\left(0.5 \times \log_2 0.5 + 0.5 \times \log_2 0.5\right) = -\log_2 0.5 = 1.$$

So, the entropy of a Bernoulli random variable with equal probabilities ($p = 0.5$), such as outcome of a fair coin toss, is 1 bit. We can also calculate the entropy with non-equal probability values. For example, entropy of a Bernoulli random variable with $p = 0.8$,

$$H = -\left(0.8 \times \log_2 0.8 + 0.2 \times \log_2 0.2\right) = 0.72.$$

Note that the entropy reduced from 1 bit to 0.72 bits. This should be intuitive, since knowing that some of the outcomes are more likely than others reduces the average surprisal. If we do the above calculations for the parameter of Bernoulli distribution p in range $[0, 1]$, we get the graph in Figure 3.2. If the outcome of the variable is certain with $p = 1$ or $p = 0$, then the entropy is 0, and the entropy peaks at 0.5, with a value of 1 bit, where the uncertainty is at its maximum.

In general, uniform distribution is the distribution with the maximum entropy for distributions with the same support (the set of values/events the distribution has non-zero probability). If the number of outcomes increase, the uncertainty and the entropy will increase. And, as the distribution diverges from the uniform distribution by assigning higher probabilities to a small set of events, the entropy will decrease. Figure 3.3 demonstrates these two factors with the categorical distribution.

Table 3.1: Example binary coding of an eight letter alphabet.

letter	code
a	000
b	001
c	010
d	011
e	100
f	101
g	110
h	111

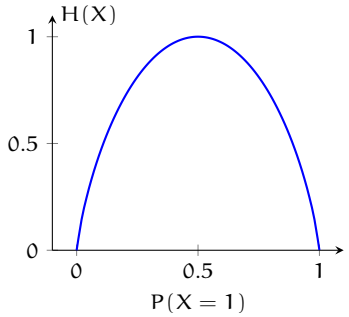

Figure 3.2: Entropy of Bernoulli random variables (in bits) as a function of the parameter p.

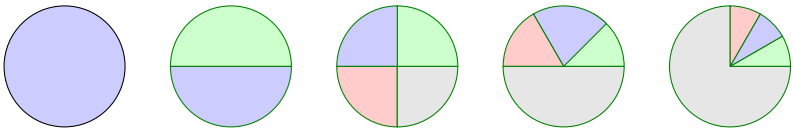

Figure 3.3: Demonstration of entropy values (in bits) with the categorical distribution. First three circles from the left represent uniform categorical distributions with increasing number of categories. Increasing possible outcomes increase the uncertainty, hence the entropy. Last three circles demonstrate categorical distributions with equal number of possible outcomes. As the distribution diverge from the uniform distribution, the uncertainty and entropy is reduced.

By now, you should be able to calculate the entropy of a categorical distribution over 8 symbols, just like the letters in Table 3.1. Given that these letters are distributed uniformly (each having probability of 1/8), the entropy of the distribution is 3 bits, which corresponds to how many bits we need for optimally coding the alphabet. We noted that we can do better if the distribution was not uniform. Let us assume that, the letter 'a' occurs with probability 1/2, the letter 'b'

occurs with probability 1/4, the letter 'c' occurs with probability 1/8, the letter 'd' occurs with probability 1/16, and the other letters occur with probability 1/64. Now, using three bits for all combinations is just wasteful. We can do much better with the codes in Table 3.2. The type of coding example here is called *Huffmann* coding. Although the number of bits are variable, we can unambiguously determine each code with its prefix. One can easily show that this coding requires less storage or channel bandwidth on average. In fact, this corresponds to the entropy of the distribution, which is 2 bits: we save 1 bit on average. This is exactly another way to interpret entropy, it is the average code-length for the best achievable code for a given distribution. In other words, entropy gives us the upper bound on an optimal encoder (in terms of efficient channel utilization) in a noisy channel model (Figure 3.1).

To give a more concrete example, we re-introduce our dialects with eight letters from Chapter 2. Table 3.3 lists the joint probabilities of letters and dialects (east, north and south), as well as marginal probabilities. From this table, we can easily calculate the entropy of the distributions of letters for the complete language, and the entropy of the letter distribution for each dialect. Now you should take a moment and try to phrase what high or low entropy means for letters or dialects. For the sake of making things more concrete, we will calculate the entropy of dialect distribution in our document set:

$$H(D) = -(0.7 \times \log_2 0.7 + 0.2 \times \log_2 0.2 + 0.1 \times \log_2 0.1) = 1.16.$$

The entropy is lower than a uniform distribution of letters over the dialects which is 1.58. In other words, by knowing $P(D)$, we gain 0.43 bits of information, in comparison to a setting where the dialects were equally represented in the data.

The distribution over individual letters is more interesting, you are encouraged to calculate the entropy of letters ($P(L)$) and compare it with a uniform distribution. Note that if you want to calculate entropy of letters within each dialect, you need conditional probabilities, e.g., $P(L \mid D = \text{east})$, for calculating letter-entropy values of individual dialects.

> Note that entropy is about complete probability distributions, while surprisal is about probabilities of individual events.

3.3 *Mutual Information and conditional entropy*

Another very important quantity from information theory used widely in computational and corpus linguistics is *mutual information*. Mutual information measures the amount of information obtained (reduced entropy) about a random variable, by knowledge of another random variable. Mutual information is a measure of dependence between two variables. If the variables are dependent (provide information about each other), then the mutual information will be high. If the variables are independent the mutual information will be 0.

Table 3.2: Example Huffman coding of an eight letter alphabet.

letter	prob	code
a	1/2	0
b	1/4	10
c	1/8	110
d	1/16	1 110
e	1/64	111 100
f	1/64	111 101
g	1/64	111 110
h	1/64	111 111

Table 3.3: Joint probability table for letters and dialects with marginal probabilities. $P(D)$ is the (marginal) probability of dialects, and $P(L)$ is the probability of letters in the corpus. This is the same as Table 2.7, repeated here for convenience.

let.	east	north	south	$P(L)$
a	0.20	0.01	0.03	0.23
b	0.03	0.01	0.00	0.04
c	0.03	0.01	0.01	0.05
d	0.06	0.01	0.01	0.08
e	0.17	0.10	0.02	0.29
f	0.02	0.00	0.01	0.03
g	0.05	0.01	0.00	0.06
h	0.15	0.04	0.03	0.22
$P(D)$	0.70	0.20	0.10	1.00

As we did with introduction to entropy, we will first introduce another relevant measure, *pointwise mutual information* (PMI), which measures the (in)dependence of two events. The PMI value for two events $X = x$ (or simply x) and $Y = y$ (or y) is calculated by

$$\text{PMI}(x, y) = \log \frac{P(x, y)}{P(x)P(y)} \tag{3.3}$$

A quick study of the formula indicates that pointwise mutual information is high if the joint probability of the two events involved is high. However, since the high-probability events can cooccur by chance frequently, the denominator of the term in Equation 3.3 includes (marginal) probabilities of both events. Hence, it discounts the by-chance cooccurrences of the events due to high marginal probability. Remember that the joint probability of two independent events is the product of their probabilities ($P(x, y) = P(x)P(y)$, for independent events x and y). As a result, the division within the logarithm in Equation 3.3 will result in 1, and PMI will be 0, for independent events. If two events are highly positively associated (they occur together), then PMI will be positive, and if events are highly negatively associated (one does not occur if the other event occurs), then the PMI value will be negative.

Just for the sake of example, we will calculate the PMI of letter 'e' occurring in the 'east' dialect in the distribution presented in Table 3.3.

$$\text{PMI}(\text{east}, e) = \log \frac{P(\text{east}, e)}{P(\text{east})P(e)} = \log \frac{0.17}{0.70 \times 0.29} = -0.22$$

Despite the fact that the joint probability of the two events is rather high, we find a negative association between them, which indicates that the letter 'e' occurs less than chance in the eastern dialect.

One of the common uses of PMI in corpus linguistics is to find *collocations*, groups of words that occur frequently together. Using PMI, it is likely to find linguistically plausible collocations like, 'corpus linguistics', even though it is much less frequent than non-interesting frequent bigrams like 'that the'.

The *mutual information* (MI) of two random variables is a measure of dependence of the variables. Mutual information is the expected value of (average) PMI. The mutual information of two discrete random variables X and Y is

$$\text{MI}(X, Y) = \sum_{x \in X} \sum_{y \in Y} P(x, y) \log \frac{P(x, y)}{P(x)P(y)}. \tag{3.4}$$

Similar to PMI, a positive MI value indicates a positive association between the random variables, zero indicates independence (no association), and negative values indicate negative association. Like *correlation* (Section 2.10), mutual information measures dependence between two random variables. However, unlike correlation, it is not limited to linear dependence.

Mutual information is related to entropy through a measure known as *conditional entropy*. We note the conditional entropy of a random

variable X given another random variable Y takes the value y as $H(X \mid Y = y)$. Intuitively, if the event $Y = y$ gives information about the random variable X, we expect low conditional entropy. The conditional entropy of X given Y, noted $H(X \mid Y)$, is the average entropy of X given Y. The conditional entropy of X given Y can be calculated by[3]

$$\begin{aligned} H(X \mid Y) &= \sum_{y \in Y} P(y) H(X \mid Y = y) \\ &= -\sum_{x \in X, y \in Y} P(x, y) \log P(x \mid y). \end{aligned} \tag{3.5}$$

[3] You are recommended to derive this equation. Meanwhile, you may also want to show that

$$H(X \mid Y) = \sum_{x \in X, y \in Y} P(x, y) \log \frac{P(y)}{P(x, y)}$$

is also correct.

The conditional entropy has also a straightforward interpretation. The conditional entropy $H(X \mid Y)$ equals to the entropy of the variable X ($H(X)$), if the variables are independent. As the dependence between variables increase, the conditional entropy will decrease.

Note that entropy, conditional entropy, and mutual information are all related. Figure 3.4 shows the relationship between these measures schematically using a Venn diagram. For example, we can see from the figure that $H(X, Y) = H(X) + H(Y) - MI(X, Y)$. The total entropy associated with two random variables is reduced if the mutual information (dependence) between them is high. If one of the variables (Y) completely determines the other (Y), the conditional entropy ($H(X|Y)$) will be 0, the joint entropy will be equal to the entropy of one one of the variables ($H(Y)$), and mutual information will be equal to entropy of the other variable $H(X)$. The mutual information is symmetric ($MI(X, Y) = MI(Y, X)$), while conditional entropy is not ($H(X \mid Y) \neq H(Y \mid X)$). Also note that the relationships between the information theoretic measures are additive, while the relationship between the corresponding probabilities are multiplicative.

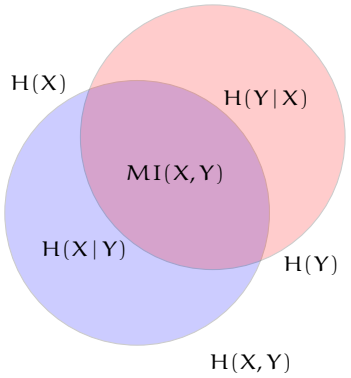

Figure 3.4: Relation between conditional entropy entropy and mutual information. The total shaded area is the joint entropy of two variables $H(X, Y)$.

3.4 *Cross entropy*

Cross entropy is another important concept from information theory that has wide usage, particularly in machine learning. Remember that entropy gives us the best achievable compression given a distribution. In many practical cases we do not know the true distribution of the data, instead we use an approximation. If we do not use the true distribution, inevitably, our code will be less optimum. This means that we will get higher entropy. If the true distribution is $P(x)$ and the its approximation is $\hat{P}(x)$, the cross entropy is defined as

$$H(P, \hat{P}) = -\sum_{x} P(x) \log \hat{P}(x). \tag{3.6}$$

The hat-notation used here is used for estimated objects. If P was the true distribution, we would note an estimation of it as $\hat{P}$. Although this is most common context we will see cross entropy used, the distribution $\hat{P}$ does not have to be an estimate of P. Note that the notation $H(X, Y)$ is also used for joint entropy, but for joint entropy the arguments are random variables noted like X and Y, and for cross entropy distribution functions noted with letters like P and Q.

The formula above makes sense if you think about it in terms of the noisy channel model. Since we are coding the data using $\hat{P}$, the code length is determined by the approximate model. However, the average is taken over the true distribution, hence, we multiply by $P(x)$ not $\hat{P}(x)$. Cross entropy is always larger than entropy. Using a wrong (approximate) distribution to code the data results in a longer code length. As $\hat{P}$ gets closer to the P, the cross entropy will approach the entropy of P. Note that cross entropy is not symmetric ($H(P, Q) \neq H(Q, P)$).

A very common use of cross entropy is as a *loss function* in some machine learning methods. In many machine learning methods, training a model is equivalent to minimizing a loss function. Hence, as we minimize cross entropy of true distribution that comes from the training data and the approximate distribution the machine learning system produces, we get closer to the true distribution of the answers we are seeking. That way, the output of the machine learning model becomes more similar to the expected output. We will see example uses of cross entropy later in this course.

For a concrete example, we return to the hypothetical dialects with 8-letter alphabets in Table 3.3. Assume that we do not know the distribution of the letters in the eastern dialect, and we are using the total distribution for coding (e.g., compressing) the data from the eastern dialect. Both distributions are summarized in Table 3.4. We simply calculate cross entropy by

$$H\left(P(L \mid \text{east}), P(L)\right) = -\sum_{x \in \text{letters}} P(x \mid \text{east}) \log_2 P(x) = 2.577.$$

As expected, this is larger than entropy of $P(L \mid \text{east})$, which is 2.562 bits.

Table 3.4: Probability distributions of letters in our hypothetical dialect data. $P(L \mid D = \text{east})$ is the distribution of letters given the dialect is east dialect, $P(L)$ is the marginal distribution of letters (for all dialects).

let.	$P(L \mid \text{east})$	$P(L)$
a	0.28	0.23
b	0.04	0.04
c	0.04	0.05
d	0.09	0.08
e	0.25	0.29
f	0.02	0.03
g	0.07	0.06
h	0.21	0.22

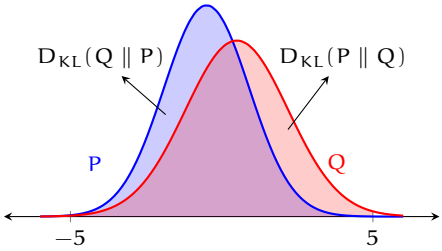

Figure 3.5: Visualization of KL divergences between two distributions. Note, however, this is a demonstration to show the asymmetry of measure, the areas indicated are not exactly what KL-divergence measures.

3.5 *Perplexity*

A measure related to entropy, used often in computational linguistic literature, is *perplexity* (PP). Perplexity is simply the exponentiated version of entropy. If we measure entropy in bits, then the perplexity is

$$PP(X) = 2^{H(X)}.$$

Like entropy, perplexity measures uncertainty. However, since it measures it in a different scale, sometime it offers a more intuitive interpretation. Its main use in NLP is evaluating language models, which are conditional distributions on words given a history of earlier words. In this context, the intuitive interpretation is the average number of words expected after each word.

3.6 *Kullback–Leibler divergence*

Another important quantity we often see during this course is *Kullback–Leibler divergence* which is also known as *relative entropy*. It is

often abbreviated as *KL divergence*. Similar to cross entropy, KL divergence is a about two probability distributions with the same support. It measures the amount of information lost, or extra bits of entropy, while using a distribution Q instead of using the true distribution P. It is defined as,

$$D_{KL}(P \parallel Q) = -\sum_x P(x) \log \frac{P(x)}{Q(x)}. \tag{3.7}$$

It is the average logarithmic difference between distributions P and Q,[4] where average calculated according to P. KL divergence is often used as a measure of difference between two distributions. However, it is not symmetric (as a result, it is not a proper distance measure). Figure **??** visualizes the KL divergence between two continuous distributions. For continuous distributions, as usual, you can simply replace probability mass functions with probability density functions, and the summation with the integral in Equation 3.7.

[4] Remember that
$$\log \frac{a}{b} = \log(a) - \log(b).$$

For a concrete example, we will return to the question of using an approximate distribution, the marginal letter distribution, instead of one of the conditional distributions in our letter distribution example in Table 3.4. As in our cross entropy example, assume that we do not know the distribution of the eastern dialect, and approximating it with the marginal distribution. If we do the calculation,

$$D_{KL}\left(P(L \mid \text{east}) \parallel P(L)\right) = -\sum_x P(x|\text{east}) \log \frac{P(x|east)}{P(x)} = 0.015$$

which is the amount of information we lose by using the marginal distribution instead of the dialect-specific distribution in bits.

Conceptually, you should already be expecting a link between the cross entropy and the KL divergence divergence. Cross entropy measures the entropy of a distribution (P) under another distribution (Q), while KL divergence measures the amount of additional entropy if one uses one distribution (Q) instead of another (P). As a result,

$$H(P, Q) = H(P) + D_{KL}(P \parallel Q).$$

You can easily verify this with our running example of dialect letter distributions as well (maybe with some rounding error).

Summary

This lecture is another quick and informal refresher on one of the important background fields that we use in this course. For more information you should consult textbooks on the subject such as MacKay (2003). Also, the original paper by Shannon (1948) which introduced some of the important concepts and in a way started the filed is accessible, and relevant to NLP.

Bibliography

MacKay, David J. C. (2003). *Information Theory, Inference and Learning Algorithms*. Cambridge University Press. ISBN: 978-05-2164-298-9. URL: http://www.inference.phy.cam.ac.uk/itprnn/book.html.

Shannon, Claude E. (1948). "A mathematical theory of communication". In: *Bell Systems Technical Journal* 27, pp. 379–423, 623–656.