Estimating entropy of the source: the case of natural language

Anna Rumshisky

July 1, 2002

Abstract

We review several different approaches to estimating the entropy of a natural language (English). We then discuss the design of a language model that takes into account higher level linguistic dependencies typically ignored in computation of the entropy rate for natural languages.

1 Introduction

Estimating the entropy of a source is an interesting problem in itself, since it creates a framework for comparing relative complexity and predictability of different random processes. However, there are also more straighforward practical applications of it, as accurate estimation of the entropy of a source allows to establish the bounds on compression efficiency.

Briefly, just to set up the problem for natural language: we wish to establish how much information, in bits, is contained in a unit of language. What is taken as a unit depends on the model. Roughly, an approximation of per-letter (or per-word) entropy rate is obtained by dividing the entropy of the entire message string by its length. In this paper, we will review several approaches to estimating the entropy, and then attempt to describe a language model that takes into account higher-level linguistic dependencies. We will for the most part leave out the discussion of statistical techniques for estimating the probability distributions necessary for certain entropy estimates.

Shannon's original work modeled English as a stationary process with a given probability distribution over single letters (ranging over the set of 26 characters and the space), or blocks of letters of length k. We are going to discuss the k-gram entropy rate estimates, then look at techniques using the human speakers to estimate the entropy rate, a guessing technique due to Shannon (1951), and its modification by Cover & King (1978). Finally, we'll describe one of the more recently used computer-based methods and propose an alternative language model.

2 Estimates of the entropy of English

When the entropy of a natural language is estimated, the language is viewed as a stochastic process, i.e. a sequence of linguistic units $L = \{X_i\}$. For example, one can think in this manner of all linguistic utterances issued by some source, such as a particular newspaper.

It seems to have become a staple disclaimer/caveat in the literature that natural language is NOT a stationary ergodic process (Cover & Thomas:133, Manning & Schütze:76, Schürmann & Grassberger:424). However, likely it can be assumed to maintain stationarity (unchanging probability distributions over symbol n-grams for all n) for fairly long periods.

As a side remark, one might say that while it may be reasonable to model some non-stationary processes in nature as sequences of stationary segments interrupted by singular events violating or modifying the stationarity, this doesn't seem to describe well the change in language. What happens rather are the occasional non-stationary segments corresponding to the periods of intensive linguistic change.

2.1 K^{th} order *Markov approximation*

This method of estimating the entropy of English relies on modeling it as a stationary process, that is, on assuming that the probabilities of particular n-grams (blocks of n symbols) remain constant $\forall n \in \{1, 2, \ldots \infty\}$. The entropy rate of a random process $\{X_i\}$ is the average number of bits of information contained in one symbol of the $\{X_i\}$ sequence. It is defined (cf. Cover & Thomas:63) as:

$$H(\mathcal{X}) = \lim_{n \to \infty} \frac{1}{n} H(X_1, X_2, \ldots, X_n)$$

Note that the "symbol" need not be a character of the alphabet, as X may range over other linguistic units, for example, words.

A way to estimate the entropy rate so defined would be to get the probability distribution on blocks of symbols of a certain length and compute the corresponding entropy. However, a better empirical estimate of the entropy rate is given by what Shannon (1951) had termed the N-gram entropy, namely, the conditional entropy of the N-th symbol, given the preceding $N-1$ symbols. The N-gram entropy is defined as

$$\begin{aligned} F_N &= H(X_N | X_1^{N-1}) = -\sum_{x_N, x_1^{N-1}} p(x_N, x_1^{N-1}) \log p(x_N | x_1^{N-1}) \\ &= -\sum_{x_N, x_1^{N-1}} p(x_N, x_1^{N-1}) \log p(x_N, x_1^{N-1}) + \sum_{x_1^{N-1}} p(x_1^{N-1}) \log p(x_1^{N-1}) \end{aligned}$$

where x_1^{N-1} is a sequence of $N-1$ letters and x_N is the Nth letter[1] (cf. Shannon, 1951; Moradi et al.)

Thus, we have $F_1 = H(X_1)$, $F_2 = H(X_2|X_1)$, $F_3 = H(X_3|X_1X_2)$, and so on.[2] Conditionining reduces entropy, so

$$H(X_N|X_1,\ldots,X_{N-1}) \leq H(X_N|X_2,\ldots,X_{N-1})$$

and for a stationary process, we get

$$H(X_N|X_1,\ldots,X_{N-1}) \leq H(X_{N-1}|X_1,\ldots,X_{N-2})$$

.

which gives us $H(X_1) \geq H(X_2|X_1) \geq H(X_3|X_1,X_2) \geq \ldots$. This being a non-increasing sequence of non-negative numbers, it must have a limit. Then, by chain rule,

$$\begin{aligned} H(X_1,X_2,\ldots,X_N) &= H(X_1)+H(X_2|X_1)+H(X_3|X_1,X_2)\ldots \\ &= \sum_{i=1}^{n} H(X_i|X_1^{i-1}) = \sum_{i=1}^{n} F_i \end{aligned}$$

dividing by n and taking a limit,

$$H(\mathcal{X}) = \lim_{n\to\infty} \frac{H(X_1^n)}{n} = \lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} H(X_i|X_1^{i-1})$$

and by *Cesáro mean*,

$$\lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} H(X_i|X_1^{i-1}) = \lim_{n\to\infty} H(X_i|X_1^{i-1}) = H(\mathcal{X})$$

(cf. Cover & Thomas:64-65). But since $\frac{1}{n}H(X_1^n) = \frac{1}{n}\sum_{i=1}^{n} H(X_i|X_1^{i-1})$, that is, it is an average of n non-increasing terms, we have $\frac{1}{n}H(X_1^n) \geq H(X_n|X_1^{n-1})$, and so $F_N = H(X_n|X_1^{n-1})$ gives a tighter upper bound on the entropy rate $H(\mathcal{X})$ than $\frac{1}{n}H(X_1^n)$.

Shannon's calculations of F_k are sometimes referred to as the k^{th} order approximations for the entropy rate of English. The n-gram probabilities necessary to calculate F_k's are obtained through standard statistical estimators such as the *maximum likelihood estimate* (cf. Manning & Schütze:197, Schürmann & Grassberger:416). For example, given the alphabet $\{a,b,c\}$, and a training corpus consisting of the string "abacaaba", the MLE estimates $p(b|a) = \frac{c(ab)}{c(a)} = \frac{2}{5}$

[1] This merely follows from the definition of conditional entropy:
$H(X|Y) = \sum_y p(y)H(X|y) = \sum_y p(y)(-\sum_x p(x|y)\log p(x|y))$
$= -\sum_{x,y} p(y)p(x|y)\log\frac{p(x,y)}{p(y)} = -(\sum_{x,y} p(x,y)\log p(x,y) - \sum_{x,y} p(x,y)\log p(y))$
$= -\sum_{x,y} p(x,y)\log p(x,y) + \sum_y p(y)\log p(y) = H(X,Y) - H(Y)$

[2] The following is merely a summary the relevant arguments from Cover & Thomas, mostly for my own use.

and $p(ab) = p(a)p(b|a) = \frac{c(a)}{n}\frac{c(ab)}{c(a)} = \frac{c(ab)}{n} = \frac{2}{8}$, where $c(s)$ is the number of times the string s occurs in the training corpus, and n is total the number of symbols in the training corpus.

Using the n-gram frequencies for the 27-letter alphabet (26 letters + space), Shannon obtained the upper bound of 4.03 for the 1^{st} order (letter unigram) approximation, 3.32 for the 2^{nd} order (letter bigram) approximation, and 3.1 for the 3^{rd} (letter trigram approximation). He also obtained the 2.14 per-letter entropy estimate using probability estimation from word unigram frequencies.

Note that with this model, the link between the entropy rate and achievable compression efficiency is easily demonstrated. Since $\{X_i\}$ is a stationary process, the expected per-symbol length of codeword for an optimal code for the blocks of n symbols $L_n^* \to H(\mathcal{X})$ as $n \to \infty$. We know that for instantaneous codes for a random variable X with D outcomes, $X \sim p$, the Kraft inequality holds $\sum D^{-l_i} \leq 1$, so the code that uses codeword lengths $l_i^* = \log_D \frac{1}{p_i}$ achieves optimality. The expected optimal codeword length is then

$$L^* = \sum p_i l_i^* = \sum p_i \log_D \frac{1}{p_i} = H_D(X)$$

.

but since $(\log_D \frac{1}{p_i})$ may be non-integer, the actual optimal codeword length merely satisfies the inequality $H_D(X) \leq L^* < H_D(X) + 1$. When considering the optimal code for blocks of n symbols, we get

$$\frac{H(X_1, X_2, \ldots, X_n)}{n} \leq L_n^* < \frac{H(X_1, X_2, \ldots, X_n)}{n} + \frac{1}{n}$$

For stationary processes, both lhs and rhs approach $H(\mathcal{X})$ as $n \to \infty$, so $L_n^* \to H(\mathcal{X})$ (cf. Cover & Thomas:87-89).

2.2 Shannon's letter prediction experiments

In his 1951 paper, Shannon suggested a method for estimating the entropy rate of English by taking advantage of the native speaker's knowledge of the language. He describes three experiments with human subjects which allow to model the redundancy of language. In the first experiment, a passage from a text is selected, and the subject is asked to guess the first letter of the passage. If he guesses correctly, he is asked to then guess the next letter. If he does not guess correctly, he is given the correct answer and then asked to guess the next letter. The space is considered a separate symbol. The result of this procedure is what Shannon calls the "reduced text", that is, the text where only the letters that were not guessed correctly (and their positions relative to the other text) are represented. Thus, the segment "THE ROOM WAS NOT VERY LIGHT" gets reduced to "- - - - R O O - - - - - - N O T - V - - - - - I - - -". Essentially, it's as though the letters absent from the reduced text are not necessary, for they can be guessed.

One can present the subject in advance with all possible k-grams of letters (with k ranging from 0 to n) and ask him to predict $k + 1$st letter. Given such table of predictions, one can easily reconstruct any original text from its reduced form. So the reduced form shows the actual amount of information contained in a text segment.

In the second version of this experiment, the subject is again presented with a text segment and asked to guess the next letter. But if the guess is incorrect, the subject continues to guess until he guesses right, and the number of guesses is recorded. As a result, another version of the "reduced text" is obtained, the one which contains numbers from 1 to 27 in place of letters, where each number corresponds to the number of guesses that was required to get a given letter. Thus, the segment, "THERE IS NO REVERSE ON A MOTORCYCLE" gets reduced to "1 1 1 5 1 1 2 1 1 2 1 1 15 1 17 1 1 1 2 1 3 2 1 2 2 7 1 1 1 1 4 1 1 1 1". In the reduced text, the frequencies of "1" and the other lower numbers are clearly much higher than any of the letter frequencies, and so it can be encoded much more efficiently, yet contains the same information. Again, one can present the subject in advance with all possible k-grams, asking in each case to arrange the letters of the alphabet according to how likely they are to appear following a given k-gram.

Shannon uses the second experiment to estimate the bounds on the entropy rate in the following manner. The subject is asked to guess sequentially each of the 100 pre-selected 15-letter text segments. The number of times it takes to guess the correct letter each time is recorded. The result is 100 sample cases where the subject had to guess based on the context of 0, 1, 2, ..., and 14 letters, respectively. For each context length $N \in \{0, \ldots, 14\}$ and possible number of guesses $i \in \{1, \ldots 27\}$, he then records the number of instances, out of a 100, for which i guesses were required.

Ideally, the subject uses his knowledge of language dependencies to rank the letters according to their conditional probability, given the preceding context. Defining q_i^N to represent the number of times (out of a 100) that it took i guesses to get correctly the N^{th} letter, given the previous $N-1$ letters, Shannon derives the following upper bound for the entropy rate:

$$H(\mathcal{X}) \leq H(X_n | X_1^{N-1}) \leq -\sum_{i=1}^{27} q_i^N \log q_i^N$$

The upper bound for $N = 100$ (length of context = 99) that Shannon obtained in this experiment was 1.3 bits per symbol.[3]

2.3 Gambling estimates

Cover and King (1978) developed a modification of the letter prediction experiment using a gambling model. In their experiments, the subject is asked not merely to arrange the letters in the order of likelihood (i.e., rank the possible

[3]The data on the change in the relative frequencies q_i^N as the length of preceding context is increased was used by Shannon to derive the lower bound on the entropy rate. This lower bound estimate, however, is not deemed reliable and is for the most part ignored in the following literature.

outcomes according to their conditional probability given the preceding context), but essentially to guess the probability distribution on the next letter given the preceding sequence.

The subject acts as a gambler that places a bet on each letter of the alphabet before the next letter in the sequence is revealed. The conditions on betting are that the gambler has to bet all of his wealth at every trial, and a non-negative bet should be placed on every letter. That is, the bet is a vector $b = \langle b_1, \ldots, b_{27} \rangle$, where b_i is the fraction of his total assets that he bets on letter i, $\sum b_i = 1$, and $b_i \geq 0$. Initially, the gambler's total assets are $S_0 = 1$, and and he places a bet $b_i * S_0$ on each of the 27 letters of the alphabet. There is a corresponding vector of payoff odds, $o = \langle o_1, \ldots, o_{27} \rangle$, where o_i is the number by which the bet amount is multiplied if letter i is seen next. So when the letter i shows up, the gambler's total assets are multiplied by $b_i * o_i$. For example, if the first word of the text segment is "the", the total assets are

$S_1(t) = (b_t * S_0) * o_t$ after the first trial,

$S_2(th) = (b_h * S_1) * o_h$ after the second trial, and

$S_3(the) = (b_e * S_2) * o_e = S_0 * b_t o_t * b_h o_h * b_e o_e = 1 * b_t o_t * b_h o_h * b_e o_e$

- after the third trial. So after seeing the sequence of n letters, the gambler's total wealth is

$$S_n(X_1, \ldots, X_n) = \prod_{i=1}^{n} b(X_i) o(X_i)$$

where $b(X_i)$ is the bet placed on the letter that occurred at position X_i, and $o(X_i)$ is the corresponding payoff.

Note that in these conditions, betting on a whole sequence $x_1, \ldots, x_n$ at once is essentially equivalent to sequentially betting on each letter as it is revealed.

$$\begin{aligned} S_n(X_1, \ldots, X_n) &= b(x_1) b(x_2|x_1) b(x_3|x_1 x_2) \ldots b(x_n|x_1^{n-1}) * \prod_{i=1}^{n} o(X_i) \\ &= b(x_1^n) * \prod_{i=1}^{n} o(X_i) \end{aligned}$$

In the gambling scheme used to derive bounds on the entropy rate, *uniform fair odds* are used, meaning that the payoff is the same on every letter, and $\sum \frac{1}{o_i} = 1$. With fair odds, if the gambler spreads his wealth amongst the letters proportionally to how much payoff each letter gives, he can always retain the initial total capital: if $b_i = \frac{1}{o_i}$, $S_i = S_{i-1} * o_i * \frac{1}{o_i} = S_{i-1}$. With uniform fair odds, the payoff vector on letters is $\underbrace{\langle 27, 27, \ldots, 27 \rangle}_{27}$.

Thus we get the following expression for the total accumulated wealth after sequential betting on n letters:

$$\begin{aligned} S_n(X_1,\ldots,X_n) &= 27^n b(x_1)b(x_2|x_1)b(x_3|x_1x_2)\ldots b(x_n|x_1^{n-1}) \\ &= 27^n b(x_1,\ldots,x_n) \end{aligned}$$

Taking a log of both sides of the equation and dividing by n, we get:

$$\begin{aligned} \frac{1}{n}\log[S_n(X_1,\ldots,X_n)] &= \frac{1}{n}\log[27^n b(x_1,\ldots,x_n)] \\ &= \frac{1}{n}[n\log 27+\log b(x_1,\ldots,x_n)] \\ &= \log 27+\frac{1}{n}\log b(x_1^n) \end{aligned}$$

Taking the expected value of of both sides of the equation,

$\frac{1}{n}E[\log S_n(X_1,\ldots,X_n)] =$

$$\begin{aligned} &= \log 27 + E(\frac{1}{n}\log b(x_1^n)) \\ &= \log 27 + \frac{1}{n}\sum_{(x_1^n)} p(x_1^n)(\log b(x_1^n)) \\ &= \log 27 + \frac{1}{n}\sum_{(x_1^n)} p(x_1^n)\log[\frac{b(x_1^n)}{p(x_1^n)}p(x_1^n)] \\ &= \log 27 + \frac{1}{n}[\sum_{(x_1^n)} p(x_1^n)\log\frac{b(x_1^n)}{p(x_1^n)} + \sum_{(x_1^n)} p(x_1^n)\log p(x_1^n)] \\ &= \log 27 - \frac{1}{n}\sum_{(x_1^n)} p(x_1^n)\log\frac{p(x_1^n)}{b(x_1^n)} - \frac{H(X_1^n)}{n} \\ &= \log 27 - \frac{1}{n}D(p(x_1^n)\|b(x_1^n)) - \frac{H(X_1^n)}{n} \end{aligned}$$

Since $D(p(x_1^n)\|b(x_1^n)) \geq 0$, this gives us

$$\frac{1}{n}E[\log S_n(X_1,\ldots,X_n)] \leq \log 27 - \frac{H(X_1^n)}{n}$$

that is,

$$\frac{H(X_1^n)}{n} \leq \log 27 - \frac{1}{n}E[\log S_n(X_1^n)]$$

As n increases, the sequence on the lefthand side of this inequality decreases and approaches $H(\mathcal{X})$, so at any given point, the following holds:

$$H(\mathcal{X}) \leq \log 27 - \frac{1}{n} E[\log S_n(X_1^n)]$$

If language is an ergodic process, it can be shown that[4] the gambler's total capital would double at the highest rate if he always bets amounts proportional to the conditional probability, given the preceding sequence (cf. Cover & King:416, Cover & Thomas, Ch. 15). That is, if at any point i of the sequential betting, $b(x_i|x_1^{i-1}) = p(x_i|x_1^{i-1})$ and so $b(x_1, \ldots, x_n) = p(x_1, \ldots, x_n)$. For an ergodic process, the expectation in the above formula can be removed, giving this upper bound for the entropy rate:

$$H(\mathcal{X}) \leq \log 27 - \frac{1}{n} \log S_n(X_1^n)$$

If S_n results from proportional betting, for a stationary ergodic process, this upper bound converges to the entropy rate with probability 1 (cf. Cover & King:415, Cover & Thomas:65,133,475):

$$\begin{aligned}
\log 27 - \frac{1}{n} \log S_n &= \\
&= \log 27 - \frac{1}{n} \log 27^n p(X_1, \ldots, X_n) \\
&= \log 27 - \frac{1}{n} n \log 27 - \frac{1}{n} \log 27^n p(X_1, \ldots, X_n) \\
&= -\frac{1}{n} \log p(X_1, \ldots, X_n) \stackrel{AEP}{\longrightarrow} H(\mathcal{X})
\end{aligned}$$

The idea of the gambling experiments is then to obtain a good empirical estimate of the maximum total wealth achievable by betting over a long sequence of letters. Human gamblers try to maximize their winnings using their knowledge of language dependencies, thus giving their best estimates of the probability of a given letter sequence. It can then be assumed that the total wealth they achieve has resulted from proportional gambling and thus should give a good estimate for the entropy rate. Following this logic, Cover and King (1978) conducted an experiment where they asked 12 subjects to bet on a sequence of 75 symbols. Averaging the capital achieved by the subjects at the end of the sequence, they used the resulting value $\overline{S_n}$ in the above formula to obtain the estimate of 1.34 bits per letter for the entropy rate.

2.4 Cross-entropy based estimates

An obvious problem with computing the entropy rate $H(\mathcal{X})$ of a process is that the n-gram probability distribution through which it is defined is not known and $p(x_1, x_2, \ldots, x_n)$ can only be estimated. In the letter prediction experiments,

[4]This is easily demonstrated for the case when there are no dependencies between letters (i.e. when $X_1, \ldots, X_n$ are i.i.d.). See Appendix.

the information about the probability distribution is implicitly elicited from the human subjects. In the approach described in section 2.1, this difficulty could be circumvented by using k-gram approximations

$$p(x_1, x_2, \ldots, x_n) = p(x_1^k) \prod_{i=k+1}^{n} p(x_i | x_{i-k}, \ldots, x_{i-1})$$

Using the notion of cross-entropy allows to formalize the idea of the data model which is used to estimate the underlying probability distribution.

Cross-entropy of a random variable. Let X be a random variable with the probability distribution p and M be a model of that random variable that stipulates an estimate m of the probability distribution p. Cross-entropy of X, with respect to its model M is given by (cf. Manning & Schütze:74, Brown et al.)

$$H(X, m) = H(X) + D(p \| m) = -\sum_{x} p(x) \log m(x)$$

That is, cross-entropy is the average number of bits that is necessary to encode an outcome of X if one uses an estimate of, rather than the actual probability distribution. The relative entropy $D(p\|m) = \sum_x p(x) log \frac{p(x)}{m(x)}$, which is the average resulting loss in bits per outcome of X, is always positive (cf. Cover & Thomas:26), so

$$H(X) \leq H(X, m)$$

Thus, given a particular model of X that provides an estimate of the probability distribution, it is possible to obtain an upper bound on the actual entropy (cf. Brown et al.).

Cross-entropy of a stochastic process. The above definition is easily extended to define the cross-entropy of a stochastic process. Just as the entropy rate is defined as the limit of the entropy of a block of variables as the size of the clock approaches infinity

$$\begin{aligned} H(\mathcal{L}) &= \lim_{n\to\infty} \frac{1}{n} H(X_1, X_2, \ldots, X_n) \\ &= -\lim_{n\to\infty} \frac{1}{n} \sum_{(x_1, x_2, \ldots, x_n)} p(x_1, x_2, \ldots, x_n) \log p(x_1, x_2, \ldots, x_n) \end{aligned}$$

so the cross-entropy of a process is defined as

$$H(\mathcal{L}, m) = \lim_{n\to\infty} \frac{1}{n} H(X_1^n, m) = -\lim_{n\to\infty} \frac{1}{n} \sum_{(x_1, x_2, \ldots, x_n)} p(x_1^n) \log m(x_1^n)$$

where X_1^n denotes $X_1, X_2, \ldots, X_n$ and m is a probability distribution over letter sequences, estimated according to the specific language model.

This definition, however, still relies on the knowledge of $p(x_1^n)$ which we don't have. But if language $L = \{X_i\}$ is modeled as a stationary ergodic process, then the Assymptotic Equipartition Property should hold (cf. Cover & Thomas, Ch. 15). By AEP, [5]

$$
\begin{aligned}
-\frac{1}{n}\log m(x_1^n) \quad &\overset{n\to\infty}{\longrightarrow} \quad -\lim_{n\to\infty}\frac{1}{n}\sum_{(x_1,x_2,\ldots,x_n)} p(x_1^n)\log m(x_1^n) \quad \text{in probability} \\
&= \quad H(\mathcal{L}, m)
\end{aligned}
$$

so the cross-entropy $H(\mathcal{L}, m)$ can be approximated as $-\frac{1}{n}\log m(x_1^n)$ for sufficiently long sequences of text (i.e. for some sufficiently large n). Since the relation between the entropy and the cross-entropy of an RV obviously continues to hold for the entropy rate of a process,

$$H(\mathcal{L}) \leq H(\mathcal{L}, m)$$

by estimating $H(\mathcal{L}, m)$, we can derive the upper bound on $H(\mathcal{L})$.

2.4.1 An example of a language model M

Now all that is left is to come up with a model that would allow to estimate the n-gram probability $m(x_1, x_2, \ldots, x_n)$. For example, Brown et al. (1992) propose a model based on word trigrams. They estimate the probability of the string $x_1, x_2, \ldots, x_n$ as the sum of probabilities of all possible splits of this string into words (or rather, tokens t_i, since they include punctuation). The probability of the string of tokens is approximated using Markovian assumption on token trigram probabilities. So we have

$$
\begin{aligned}
&m(x_1, x_2, \ldots, x_n) = \\
&\qquad = \sum_{t_1^k} m(t_1, t_2, \ldots, t_k)\ m(\text{some other model variables}|t_1, t_2 \ldots, t_k)
\end{aligned}
$$

where other model variables include things like varied capitalization of the words, and $m(t_1, t_2, \ldots, t_k)$ is computed as

$$m(t_1, t_2, \ldots, t_k) = m(t_1, t_2)\prod_{i=3}^{k} m(t_i|t_{i-2}, t_{i-1})$$

[5] AEP states that $-\frac{1}{n}\log p(x_1^n) \to H(\mathcal{X}) = \lim_{n\to\infty}\frac{1}{n}\sum_{x_1^n} p(x_1^n)\log p(x_1^n)$ in probability. That is, $-\frac{1}{n}\log p(x_1^n) \to -\lim_{n\to\infty}\frac{1}{n}E_p \log p(x_1^n)$ in probability.

The trigram probabilities $m(t_i|t_{i-2}, t_{i-1})$ are estimated from a training corpus using a variation of the *maximum likelihood estimate* with smoothing for the sparsity.[6]

This model, trained on a corpus of about 600 million words, was used to compute the probability of the one million word Brown Corpus. The resulting estimate gave the upper bound of 1.75 on the per-character entropy of English.

3 Extending entropy estimates to include higher level linguistic constructs

Of the above models, the computer-based ones only consider the level of expression in language, that is, the actual representation of concepts. The underlying complexity of meanings represented by linguistic utterances is entirely ignored. This has been pointed out, for example, by Kontoyiannis (1997) who obtained a much smaller estimate of the entropy based on the text of the Bible than on a couple of novels by James Joyce. The Bible is known to contain only a few hundred roots, and James Joyce is notoriously complex reading, which seems to be confirmed by the entropy estimates. However, Kontoyiannis remarks that this may bear little on the relative complexity conceptual content of the two texts.

Ideally, we would like to create a language model that would allow to estimate the probability of a particular string of letters using higher order linguistic dependencies, i.e. the dependencies of the kind that native speakers would use in order to predict the next letter or word in the language "sequence", such as the knowledge of non-local dependencies in grammar (1), the knowledge of the topic of the text (2), the knowledge of selectional preferences of the verbs (3), etc.

(1) P("I will gladly and without hesitation" + "sacrifice") > P("I will gladly and without hesitation" + "computer")

(2) P("estimate the" + "entropy") > P("estimate the" + "impact")

(3) P("drink this very nice" + "coffee") > P("drink this very nice" + "chair")
P("spill the hot coffee" + "on") > P("spill the hot coffee" + "with")

3.1 An abstract specification of a language model

Thus, we want our model to allow assigning higher probabilities to grammatical constructions, to those that are well-formed from the point of view of semantic restrictions, and so on.[7] A single word does not contain this information so

[6]MLE estimates trigram probabilities as $p(t_i|t_{i-2}, t_{i-1}) = \frac{c(t_{i-2}, t_{i-1}, t_i)}{c(t_{i-2}, t_{i-1})}$, where $c(s)$ is the number of times the string s occurs in the training corpus.

[7]Ideally, the computation should allow for the fact that the specific words used need not have been encountered in the training text.

the logical solution is for the model to compute probabilities over higher order linguistic units, such as syntactic phrases or whole sentences.[8]

In order to allow long-range dependencies to be factored in, and also to avoid giving special treatment to languages with fixed word order, the probability of a sentence would need to be computed as follows. We define sentence (our linguistic unit of choice) as a collection of tokens (words) with superimposed ordering.

$$Pr(\text{sentence}_i) = Pr(\{t_1, t_2, \ldots, t_k\})Pr(\text{ordering}|t_1^k)$$

- where the probability of a set of tokens would be computed as the probability of the most probable partition of that set into subsets.[9] That is, defining partition π of a set S as a collection of non-empty subsets $C_i \subseteq S$ (constituents) such that $S = \bigcup C_i$ and $C_i \cap C_j = \emptyset$, we would compute the probability of a set of tokens as:

$$Pr(\{t_1, t_2, \ldots, t_k\}) = \max_{\pi} Pr(\pi(\{t_1, t_2, \ldots, t_k\}))$$

If we assume independence of each constituent so defined, the probability of a particular partition should just be:

$$Pr(\pi(\{t_1, t_2, \ldots, t_k\})) = \prod_i Pr(C_i)$$

This might seem a rather frivolous approximation, especially in view of the previous remarks in regard to semantic restrictions imposed by the constituents of a sentence on one another. However, we must remind ourselves that what we are dealing with here are not constituents in the traditional linguistic sense, but rather sets of words likely to occur together in a sentence.

We would certainly still get some elements identified by the traditional syntactic analysis. For example, "to" will be much more likely to be lumped into one subset with a non-inflected verb than any of its arguments. But the hope is that other combinatory tendencies of words could also be captured. Thus, in (3) above, we would hope that "drink" and "coffee" would end up in the same subset.

This also makes it clear why we should be able to assume that constituents occur in a sentence independently of one another. Since we chose the probability-maximizing partition, the dependencies existing between lexical items within the sentence would force them into one subset, thus making the assumption of independence of the resulting constiuents justified.

[8]If we assume the approximation that discounts the discourse-level dependencies, such as the anaphoric links, register or topic restrictions, and so on, the sentences should be considered independent, in which case the probability of a string would be computed as sentence-unigram approximation.

[9]The probability of a particular ordering of a given set of tokens is, perhaps, better estimated by considering the ordering of the corresponding parts of speech, given a set of part of speech tags.

The actual "meat" of the model would then lie with the estimation of the probability $Pr(C_i)$ of a given constituent. This would be done by using the cooccurrence statistics of the tokens comprising it, obtained from the training corpus. It would also inevitably require generalization across lexical and grammatical classes of tokens to make up for the sparsity of data when the frequency counts are very low.

4 Conclusion

We have reviewed just a few of the approaches to the problem of estimating the entropy rate of a natural language. There is number of other approaches that we have not covered. For example, Teahan and Cleary (1996) use an estimate technique based on the PPM (prediction with partial matching) scheme. Kontoyiannis (1997) proposes some entropy estimators based on Lempel-Ziv coding scheme, thereby purportedly accounting for the complexity due to long-range dependencies in the text. The latter estimates are based on Wyner and Ziv's result that given a stationary process X_i, the length Λ_i^N of the shortest string starting at position i that does not appear amongst the preceding N symbols grows as $\log N$ with the slope $\frac{1}{H}$: $\frac{\Lambda_i^N}{\log N} \stackrel{N \to \infty}{\to} \frac{1}{H}$. Kontoyiannis estimates Λ_i^N by taking an average of its value at M positions, for a fixed large N.

For a practical application of the language model outlined in section 3.1, we would need to specify how the estimates of the probability distribution on the constituents C_i should be obtained. We did not discuss here the techniques used to estimate from the training data the probabilities necessary to obtain a particular entropy estimate. For example, n-gram probabilities $p(x_1, x_2, \ldots, x_n)$ are often assessed using extensions of the *maximum likelihood estimate* which are designed to compensate for the sparsity of data typical for NLP. Examples are Laplace's estimate (adjusting the frequency counts of n-grams by 1), Lidstone's estimate, ELE (*expected likelihood estimation*), the Good-Turing estimate (assumes binomial distribution on the occurrence of n-grams) and other methods of allocating some probability space to events that had not been encountered in the training set. Various mechanisms of cross-validation and held-out estimation are used to assess exactly how much of the probability space should be allocated to the "unseen" events (cf. Manning & Schütze:202-213; Church & Gale, 1991).

And finally, there are methods for combining estimators in such a way that in the absence of data, the model defaults to less specific probability estimators. In such methods (for example, *deleted interpolation*, or the *back-off models*), the contribution of each estimator is assigned a weight, so that the resulting weighted sum still defines a valid probability distribution (cf. Manning & Schütze:217-221; Church & Gale; Jelinek & Mercer). For example, for estimating token trigram probabilities, Brown et al. use a variation of the formula $p(t_3|t_1t_2) = \lambda_3 f(t_3|t_1t_2) + \lambda_2 f(t_3|t_2) + \lambda_1 f(t_3)$, where f_i are the conditional frequencies in the training corpus and $\sum \lambda_i = 1$. The values of λ_i are determined by the bigram and unigram counts in a held-out segment of training data.

If $c(t_1 t_2)$ is large, λ_3 is set close to 1, as the trigram count is then a reliable predictor of the probability $p(t_3|t_1 t_2) = \frac{c(t_1 t_2 t_3)}{c(t_1 t_2)}$, and so on.

5 Appendix

5.1 Proportional gambling maximizes the rate at which total capital doubles

Let S_n to be the cumulative wealth achieved after seeing n letters. Let $S(X_i) = b(X_i)o(X_i)$ be the factor by which the current wealth is multiplied after seeing letter i.

$$S_n = 2^{n(\frac{1}{n}\log S_n)}$$

$$S_n = \prod_{i=1}^{n} S(X_i)$$

$$\frac{1}{n}\log S_n = \frac{1}{n}\log\prod_{i=1}^{n} S(X_i) = \frac{1}{n}\sum_{i=1}^{n}\log S(X_i)$$

In case when there are no dependencies between letters (i.e. when $X_1, \ldots, X_n$ are i.i.d.), the betting scheme (vector $b = \langle b_i \rangle$) and the corresponding odds remain the same, so

$$\frac{1}{n}\sum_{i=1}^{n}\log S(X_i) \to E(\log S(X)) \quad \text{in probability}$$

Now we can maximize $E(\log S(X))$ differentiating it with a Lagrangian multiplier:

$$\max E(\log S(X)) = \max_{b_i}\sum_{i=1}^{27} p_i \log b_i o_i = \frac{1}{\ln 2}\max\sum_{i=1}^{27} p_i \ln b_i o_i$$

$$f(b) = \sum_{i=1}^{n} p_i \ln b_i o_i + \lambda\sum_{i=1}^{27} b_i$$

$$(p_i \ln b_i o_i)'_{b_i} + (\lambda b_i)'_{b_i} = (p_i\frac{1}{b_i} + p_i \ln o_i)'_{b_i} + \lambda = \frac{p_i}{b_i} + \lambda$$

At the extremum, all partial derivatives should be 0, so:

$$\frac{\partial f}{\partial b_i} = \frac{p_i}{b_i} + \lambda = 0$$

$$b_i = -\frac{p_i}{\lambda}$$

with the constraint $\sum b_i = 1$

$$-\sum_{i=1}^{27} \frac{p_i}{\lambda} = 1$$

$\sum p_i = 1$, so

$$\lambda = -1$$

$\Rightarrow$

$$b_i = -\frac{p_i}{\lambda} = p_i$$

That is, $\langle b_i \rangle = \langle p_i \rangle$ is an extremum, which is easily shown to be a maximum.

References

[1] T. Schürmann and P. Grassberger. Entropy estimation of symbol sequences. *Chaos* **6** (3), 1996.

[2] C. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, (27):379-423, 623-656. 1948.

[3] C. Shannon. Prediction and Entropy of Printed English. *Bell System Technical Journal*, (30):50-64. 1951.

[4] H. Moradi, J. Grzymala-Busse, and J. Roberts. Entropy of English Text: Experiments with Humans and a Machine Learning System Based on Rough Sets. *Information Sciences* **104**, 31-47, 1998.

[5] C. Manning and H. Schütze. Foundations of Statistical Natural Language Processing. *The MIT Press. Cambridge, MA*, 1999.

[6] T. Cover and J. Thomas. Elements of Information Theory. New York: John Wiley and Sons, 1991.

[7] T.M. Cover and R.C. King. A convergent gambling estimate of the entropy of English. IEEE Transactions on Information Theory, 24(4):413-421, 1978.

[8] F. Brown, S. Della Pietra, V. Della Pietra, J. Lai, and R. Mercer. An Estimate of an Upper Bound for the Entropy of English. *Computational Linguistics* **19**(1):31-40, 1992.

[9] P. Chen and F. Alajaji. Lecture Notes in Information Theory. Volume 2. http://citeseer.nj.nec.com/433033.html

[10] W. Teahan and J. Cleary. The entropy of English using PPM based models. *Proc. Data Compression Conference. IEEE Society Press*, 53-62. 1996. http://citeseer.nj.nec.com/cleary96entropy.html

[11] I. Kontoyiannis. The Complexity and Entropy of Literary Styles. 1997. NSF Technical Report No. 97. Department of Statistics, Stanford University. http://citeseer.nj.nec.com/kontoyiannis97complexity.html

[12] C. Nevill-Manning, I. Witten, and D. Maulsby. 1994 Compression by induction of hierarchical grammars. *Proc. Data Compression Conference*, 244-253.

[13] T. Bell, J. Cleary, and I. Witten. Text Compression. *Prentice Hall*. 1990.

[14] K. Sayood. Introduction to Data Compression. *Morgan Kaufmann Publishers*, San Francisco. 1996.

[15] K. Church and W. Gale. 1991. A comparison of the enhanced Good-Turing and deleted estimation methods for estimating probabilities of English bigrams. *Computer Speech and Language* 5:19-54.

[16] F. Jelinek and R. Mercer. 1980. Interpolated estimation of Markov source parameters from sparse data. In *Proceedings, Workshop on Pattern Recognition in Practice*, pg. 381-397, Amsterdam.

[17] F. Jelinek and R. Mercer, 1985. Probability Distribution Estimation from Sparse Data", IBM Technical Disclosure Bulletin 28, 2591-2594.