CS 5180 Project 1 Lijie Xia

1. Exceptionality of words within a single DNA sequence

Setting:

Only letters and words from complete_book_of_cheese_nows were used

P(letter) = the probability of letter occurres in the complete_book_of_cheese_nows

P(word) = P(w_1) * P(w_2) * ... * P(w_k), where w_i are letters in the word

E(word) = (l - k + 1) * P(word), where l is the length of the book, k is the length of the word

Outputs:

part1_result_top_bott_50O_by_length.csv

Words are separated regards to the word length and then sorted on the observation of the occurrence.

The top 50 most occurred and the top 50 least occurred words are saved

part1_result_top_bott_50OE.csv

Words are sorted on the O/E scores

The top 50 highest O/E scores words and the top 50 lowest O/E score words are saved

part1_result_top_bott_50OE_by_length.csv

Words are separated regards to the word length and then sorted on the O/E score.

The top 50 highest O/E scores words and the top 50 lowest O/E score words are saved

Discussion:

In part1_result_top_bott_50OE.csv, we can see that the 50 most over-represented words are long sequences, while the 50 most under-represented are all short words.

The reason why we get the result is that by independence, P(word) is defined to be the product of P(letter) for all letters in the word. When the word gets longer, the P(word) vanishes.

The part1_result_top_bott_50O_by_length.csv and part1_result_top_bott_50OE_by_length.csv are created to compare with the results in part1_result_top_bott_50OE.csv.

It's embarrassing that even part1_result_top_bott_50O_by_length.csv contains more actual words than part1_result_top_bott_50OE.csv

Problem of this approach:

Letters are considered to be independent, and context are not considered.

Result of the problem:

The O/E score is overwhelmed by the probability of a word

Illustrate the prolbem:

The O/E score of the most occurred word, "THE", in the complete_book_of_cheese_nows

Observation("THE") = 3868

P(letter = 'T') = 0.0795812905

P(letter = 'H') = 0.0469924117

P(letter = 'E') = 0.1286709453

P(word = "THE") = 0.0795812905 × 0.0469924117 × 0.1286709453 = 0.000481193

E(word = "THE") = (326905 - 3 + 1) * 0.000481193 = 157.303399822

O/E = 3868÷157.303399822 = 24.589424033

The O/E score is not outstanding even comparing to other words with length 3, which can be seen in the file part1_result_top_bott_50OE_by_length.csv

2. Exceptionality of words across a number of different DNA sequences

Setting:

all eight books are used to compute the probability of letters

P(letter = 'j') = the probability of 'j' occurres in all books = the_number_of_'j'_occurs_in_all_books / total_letters_in_all_books observation & expectation: number of books that contain W (word)

Define:

l = len(seq_i)

k = len(W)

y_i # of occurrence of W in seq_i

z_i if seq_i contains W

$Z = \sum(z_i)$ number of books that contain W

Then:

$p(W) = \prod(P(w_j))$, where w_j are letters in the word

$p(z_i) = p(y_i >= 0) = 1 - p(y_i == 0) = 1 - (1 - p(W))$^(l-k+1)

$\underbrace{}$ 1 - probability that book i doesn't contain word W

$E(Z) = E[sum(z_i)] = sum[E(z_i)] = sum[p(z_i)]$

Outputs:

part2_result_top_bott_50O_by_length.csv

Words are separated regards to the word length and then sorted on the number of books contains the word.

The top 50 most contained and the top 50 least contained words are saved

part2_result_top_bott_50OE.csv

Words are sorted on the O/E scores

The top 50 highest O/E scores words and the top 50 lowest O/E score words are saved

part2_result_top_bott_50OE_by_length.csv

Words are separated regards to the word length and then sorted on the O/E score.

The top 50 highest O/E scores words and the top 50 lowest O/E score words are saved

Discussion:

In part2_result_top_bott_50OE.csv, all of the top 50 words get infinite O/E score, which is because python underflowed when computing the expectation.

In the equation, p(z_i) = 1 - (1 - p(W))^(l-k+1), when W becomes very long, p(W) becomes very small, and python underflows. Hence, for those long words E(Z) becomes zero and O/E score becomes infinity

part2_result_top_bott_50OE_by_length.csv was created to analyze the performance of the model on different word length. In this file, we can see that start from words longer than 8 characters, python start to fail computing the O/E scores.

In addition, this model cannot generate better result comparing to the model in part 1. This is because the observation and expectation are both number of books, and it's easy for meaningless words to achieve same observation as English words; however, the expected value of meaningless words.

part2_result_top_bott_50O_by_length.csv was created to examine if the observation is a better parameter comparing to O/E score here. Because the observation is the number of books in this model, it's easy for both English words and meaningless sequences occurs in all books. Thus, this sorting on observation cannot provide us any meaningful result.

3. Towards a "perfect word dictionary"

There's another problem with the models used in part 1 and part 2, which is the order of letters doesn't matter. This problem is not reflected in the part 1 and part 2 output file because the model is so bad that even without this problem, we would still get similar bad result.

In order to get a better dictionary-like result, considering the dependency between letters could be a better way comparing to the approach we have in part 1 and 2, where letters were considered as a Bernoulli variables. We just covered Markov Chains in Wednesday's class which takes the form $P(w_i \mid w_{i-1})$.