TP3 - Zipf's Law and Precision/Recall

Atul Sinha
e-mail: Atul.Sinha@unige.ch

March 29, 2021

1 Goal

1.1 Zipf's Law

In the first part of this exercise we study the statistics of word occurrences in texts. An observation made by George Kingsley Zipf (1902-1950) states that, in general:

- A small number of words occur very frequently
- Many words occur rarely

More formally, the above Zipf's law states that the *probability* of encountering the r-th most common word is inversely proportional to its rank r, that is,

$$P(r) \approx 0.1/r. \tag{1}$$

For example, according to the above equation, the most frequently occurred word w_1 with $r = 1$ has a probability of $P(1) \approx 0.1$, meaning that roughly one of every ten words is w_1. The second most frequent word has a probability $P(2) \approx 0.05$, meaning that one of every twenty word is w_2, and so on.

Terminologies

Frequency (f) number of occurrences of a word in a text

Corpus a collection of documents of a particular kind

Rank of a word (r) a word's ordinal number in a list sorted by decreasing frequency (f)

Vocabulary set of all unique words in a corpus

1.2 Evaluate your search Engine

In the previous lab, you built search engines able to rank documents d_j according to their relevance r_j w.r.t a query. The aim here is that you build the evaluation framework for assessing the quality of the retrieval.

For this you need a ranked list (permutation of the index list of the documents $\{1, ..., N\}$) and a relevance list (list of N binary values indicating if document d_j is relevant ($r_j = 1$) or not ($r_j = 0$)).

Precision wll assess the density of relevant documents when exploring the ranked list downwards. Recall indicates the proportion of relevant documents retrieved so far (among all relevant documents).

At one step n of such a scrolling of the ranked list one may place the threshold "positive above, negative below" and compute the related P_n and R_n value and therefore plot one (R_n, P_n) point of the P-R curve (for one query).

2 Tasks

2.1 Evaluate your search Engine

Your task here is to build functions computing and plotting the P-R curves in several settings. You may use:

- $N = 10$, the ranked list is $\{2, 4, 5, 3, 6, 1, 7, 8, 0, 9\}$ and the relevant list is $\{1, 1, 1, 1, 0, 1, 0, 0, 0, 0\}$ (you may also make up your own data - including perfect ranking)
- Random or manual: Choose N, generate a binary list, generate several permutations.
- Based on your previous labs using `nasa` data, generate ranking for queries "engine" and "analysis". For the relevance set use the command `grep -i <query> *.key` to select the relevant documents.

1. Create a function taking as input the ranked and relevance lists, level n and providing you with (R_n, p_n)
2. Given 2 queries (any data above), plot the P-R curves.
3. Average these two P-R curves using the 11-point technique
4. (optional) Compute nDCG on the above (see Learning to Rank). Discuss nDCG against P-R
5. (optional) Compute the mAP for the above queries
6. (optional) Get more queries as above and use micro and macro-averaging techniques
7. (optional) Create functions providing you with the Spearman correlation and Kendall Tau between the above lists.

2.2 Verify the Zipf's Law

1. Choose one or more French books as your corpus from the project Guttenberg web site `http://www.gutenberg.org/`, for example
   - "20000 Lieues Sous Les Mers" `http://www.gutenberg.org/ebooks/5097`
   - "Les Trois Mousquetaires" `http://www.gutenberg.org/ebooks/13951`
   - "L'homme Qui Rit" `http://www.gutenberg.org/ebooks/5423`
   - "L'Odyssée" `http://www.gutenberg.org/ebooks/14286`
   - "Histoire de la Révolution franaise" `http://www.gutenberg.org/ebooks/9945`
2. Verify the Zipf's Law. For this you need to:

- Identify all unique words in your corpus. One way to do this is to tokenize your corpus by splitting based on white space characters. If a token match a predefined regular expression, then memorize it as a valid word. This is for filtering non-word tokens like "****", "—", etc.
- Count the frequencies of all words in your corpus and arranges them in a list according to their rank. You may program in `Python`, `Matlab`, `Perl`, etc.
- Transform the frequencies into probabilities by normalizing each frequency using the total number of words in your corpus. On the same diagram, plot the obtained probabilities against their ranks, the theoretical relationship according to the formula of Zipf's law and a linear regression (least-squares fit) line. Please report the values of R-squared, p-value and attach a residual plot for the linear regression. Justify whether linear regression models can be used to explore the dependence between words' probabilities and their ranks. Comment on how the approximation fits the theoretical formula.

3. From the data you obtained, find 10 examples of extremely frequent, very rare, and averagely frequent words to fill out the following table

	Very Frequent Words	Averagely Frequent Words	Very Rare Words
1			
2			
⋮			
10			

Intuitively, which of the above three word categories could be more useful in information retrieval? Which of these categories is likely to have large `tf-idf` values? Why?

3 Assessment

The assessment is based on your report. It should include all experimental results, your answers to all questions, and your analysis and comments of the experimental results. It should be around 4 pages for this assignment. Please try to detail the report by giving examples and conclusions. Please archive your report and codes in "PrenomNomTP3.zip", and upload to `http://moodle.unige.ch` under TP3 before April 18, 2021, 23:59PM. Later submission will not be accepted.