Project 2: Zipf Law and validation 2. Project members: Aapo Juutinen, Eetu Ervasti, Niklas Riikonen

***Abstract*— In this project we aim to test Zipf law on a large corpora, in this case we will use Wikipedia dump which contains [all articles from English Wikipedia?]**

# Introduction

Zipf’s law states that the frequency of n:th most used word in any natural language (or any large collection of texts, such as corpus) is inversely proportional to its rank in the frequency table (of the given corpus). This means that the most commonly occurring word will occur twice as often as the second most common word, three times more often than the third most common word and so on. This law is known to be true for most languages, but the reasons are not well known.

Mathematically, Zipf’s law can be expressed as:

Where **r** is the rank of the given word in the corpus (i.e. the most frequent word’s rank is 1, second most frequent word’s rank is 2 etc.). **Pr** is the probability (frequency) of the given word.

The most common methods for observing Zipf’s law is to plot log(rank) vs log(frequency) of the words in the given corpus and examine the resulting point-plot. If the points fall on a straight line, we can see this as evidence that our data supports Zipf’s law. We will demonstrate this and more statistical methods for Zipf’s law validation later on this paper.

In this paper, Wikipedia corpus is constructed and it will be used to validate Zipf’s law. There are few problems dealing with the size of Wikipedia corpus, there are more than 6 million articles as of today in English Wikipedia [Ei meillä varmaa oo kaikki?]. With enough time, we managed to make the corpus [containing xxx] and convert it into raw text using python. The final size of the resulting text file was [?].

Our first goal was to examine the good of fit of Zipf’s law associated with the whole corpus. We also used statistical methods to find the good of fit of Zipf's law.

Later on, we chose two contrasting topics of our choice, to further validate Zipf’s law. We chose topics [X] and [Y]. [These topics were ? out of our corpus using ??]. Later we focused mainly on the 1000 most frequent words in each query. We also analyzed similarities and differences between these two queries.

# Methodology

Before we can start analyzing our data and validating Zipf’s law, we have to obtain the Wikipedia corpus. This takes a lot of time considering the size of the corpus, but we will have to manage. Wikipedia keeps their own “dump files” online and they are free to download for everyone. We download the most recent dump file, in our case we took file named “xxx” [lähde: <https://dumps.wikimedia.org/enwiki/latest/>]

After the download is done, we will have a large file, which we will have to convert into plain text corpus. This is done using python and one of its many useful libraries, gensim. This step will also take several hours. After this, we will have a large text file [xx gb?] which we can use for our analyses.

For our first task, making a barplot of the 30 most frequent words in this corpus, we used pythons main plotting library xx. This allowed us to have a first glimpse of our data and see what kind of words were the most frequently used. [you can see the results in figure 1].

After this, we finally got to test if Zipf’s law is present in our data. To draw Zipf’s law associated to this corpus, we used pretty simple python coding. These results can be seen in figure 2.

The next step was to quantify the good of fit of the Zipf’s law using statistical confidence interval at 90%. [Tähän ehkä jotain sepostusta koodista yms.]. If all of the points are within these intervals, we can say the fit is statistically valid (at 90% confidence). [We also tried other intervals than 90% and the results of these fits can be seen at table 1].

The next step was to query our two contrasting topics, X and Y. [Tähän sitten selittää että miten se sitte oikeesti tehtiinkää]. To explore these two queries vocabulary and their possible differences and/or similarities, we extracted the 1000 most frequent words from each query and plotted graph of frequency vs. rank and fitted Zipf’s law. [The results can be seen from figure 2]. After, we calculated Jaccards similarity between these queries using python [xxx…]

To analyze the variation of common words with respect to frequency, we calculated the following things:

Where V1 is the number of words in query 1 whose frequency are between values of *i,* and likewise V2 is the number of words in query 2 whose frequency is between values of *i*. The intervals for word frequencies of *i* are the following:

So, just to be clear, for value of *i = 3* we calculate

We repeat this step for each *i* = 1…10 and we have ingredients to analyze the variation of the common words.

Then, we reproduce this step to get values for . This is simply done by replacing V1 from the numerator with V2.

Now we can draw curves for R1 and R2.