# Knowledge base for Word Prediction

Authors: Zoharit Hadad, id:312211162

Or Hayat, id:312198377

knowledge-base for Hebrew word-prediction system, based on Google 3-Gram Hebrew .dataset, using Amazon Elastic Map-Reduce (EMR)

## Introduction

We will generate a knowledge-base for Hebrew word-prediction system, based on Google 3-Gram Hebrew dataset, using Amazon Elastic Map-Reduce (EMR). The produced knowledge-based indicates for each pair of words the probability of their possible next words. In addition, we will examine the quality of the algorithm according to statistic measures and .manual analysis

## Goal

Our goal is to build a map-reduce system for calculating the conditional probability of each trigram (w1, w2, w3) found in the corpus:  Hebrew 3-Gram dataset of [Google Books .Ngrams] (https://aws.amazon.com/datasets/google-books-ngrams/)

The output of the system is a list of word trigrams (w1, w2, w3) and their conditional .(probabilities (P (w3|w1, w2))

.The list will be ordered: (1) by w1w2, ascending; (2) by the probability for w3, descending

:For example

0.6 קפה נמס עלית

0.4 קפה נמס מגורען

0.6 קפה שחור חזק

0.3 קפה שחור טעים

0.1 קפה שחור חם

# Description

Our application has 5 steps:

**Step1**

Step 1 is the WordCount class, its takes as input the 3-gram corpus and parses it line by line

The Mapper class divide the corpus for two parts (with mod 2) and return as key the ngram and as value the occurrence of the ngram in specific part.

In the reducer, we combine all the occurrences in both parts to get the sum of all the words and also count N0 and N1,

**Step2**

Step 2 is the Calc class, this class takes as input the 3-gram as key and as value the two sum N0: N1 and send the reducer the key 1/0 + N0/N1 and the value Pair (N1/N0, 1).

In the reducer we calculate T and Nr0 Nr1, we combine all the occurrences by the value key

and send T and Nr0/Nr1.

**Step3**

Step 3 is the calcProb class, takes as input the Nr0/Nr1 and T and send the reducer Nr0/Nr1 as key and T: Nr1/Nr0 as value

The reducer gets Nr0/Nr1 and T, also from the first step get N and calculate the probability for specific r.

$$P_{del}(w_1...w_n) = \frac{T_r^{01} + T_r^{10}}{N(N_r^0 + N_r^1)}, \text{where } C(w_1...w_n) = r$$

- N is the number of n-grams in the whole corpus.
- $N_r^0$ is the number of n-grams occurring *r* times in the first part of the corpus.
- $N_r^1$ is the number of n-grams occurring *r* times in the second part of the corpus.
- $T_r^{01}$ is the total number of those n-grams from the first part in the second part of the corpus.
- $T_r^{10}$ is the total number of those n-grams from the second part in the first part of the corpus.

**Step4**

Step 4 is the joinClasses class, this class takes as input step 3 output and a corpus of ngrams we join the ngrams table and the probability of its r in the corpus that was calculated in step 3.

We use reducer side join that is based on sorting the keys, so we assume nothing on the numbers each word appears in each r.

We send as output from this step the P(del) of each trigram in the corpus.

**Step5**

Step 5 is the SortOutPut class, this class take step 4 as its input

The mapper sorts the ngrams as required (first sorted by the first 2 words and then by the probability)

And reducer just write the sorted output to file

## Main

The main includes all the steps and the arguments that we need. We create a cluster, give it all the steps and arguments and we let the application to run.

Link for EmrControl.jar: s3://ass2dsp191/EmrControl.jar