**Assignment 3 – Design Document**

We split the problem into **3 stages** of Map-Reduce:

**Job 1 + Job 2:**

Calculate the **4 association metrics** for each **<Lexeme,Feature>** pair.

**Job1**

We made a **custom key** class **WordAndTagKey** that holds **a** **word and a tag**, to be able to calculate the different count metrics: , in the same job.

Mapper

Input -> corpus

Output -> **<WordAndTagKey,LongWritable>**

There are **4 key options**:  
**<lexeme, 'Lex'>, count\_l  
<lexeme feature, 'Pair'>, count\_lf  
<\*, 'L'>, count\_L**  
**<\*, 'F'>, count\_F**

the sorting: F < L < Lex < Pair

Reducer:

F -> sum all the values and emit to **LFFile**  
L -> sum all the values and emit to **LFFile**  
for each Lexeme the reducer will get the Lex tag for the lexeme and sum all the values to get the count\_l, we will save it as field in the reducer and will get next all the **<lexeme,feature>** Pairs with that specific lexeme and emit the **<lexeme,feature>**,**count\_lf,count\_l**

Out1 line: **<lexeme,feature> count\_lf, count\_l**

outLF file: **L Count(L)**  
 **F Count(F)**

**Job2**

add the count\_f values to each line of out1

Mapper:

Input -> corpus + out1

If corpus line -> emit **<feature,“F”>, count**  
if out1 line -> emit **<feature,“out” >, line**

Reducer:

The sorting will send **same features** in the key to the **same reducer**, with the **“F”** tag keys **first**.

This will enable us to save local field with the total **count\_f** for a feature and immediately afterwards get all the **lexeme feature** pairs and

For each **<lexeme,feature>** pair in the reducer we will have all of the necessary data to calculate the **4 association metrics**.

Out2 line: **<lexeme,feature>, assoc1, assoc2, assoc3, assoc4**

**Job3:**

Mapper

Input -> out2

**Estimations:**

**Job1:**

Key-Value pairs:

Memory usage:

**Job2:**

Key-Value pairs:

Memory usage:

**Job3:**

Key-Value pairs:

Memory usage: