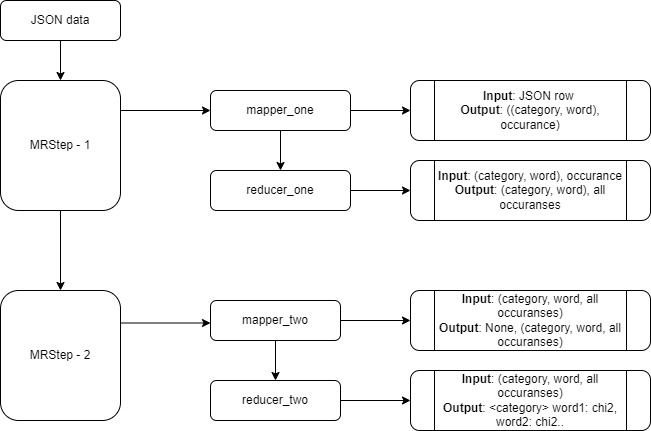
**Data intensive computing**

**Exercise 1: Text Processing Fundamentals using MapReduce**

**Teodor Chakarov N: 12141198**

I had to create a MapReducer job, which is used and powered by Hadoop for big data processing.

The big picture looks like so:



With the help of python and its library Mrjob, I can stream data into different Hadoop clusters and preprocess it.

Code explanation:

    def mapper\_one(self, \_, line):

        stops = set(i.strip() for i in open('stopwords.txt'))

        review = json.loads(line)

        for cat in WORD\_RE.findall(review['category']):

            for word in WORD\_RE.findall(review['reviewText']):

                if len(word) != 0 and word.lower() not in stops and any(x in word for x in delimiters) == False:

                    yield ((cat.lower(), word.lower()), 1)

I used 2 MrJob steps. In the first one I load line by line the data from the JSON file. I iterate for each category and each review to get rid of stopwords and then I allow only words which are more than 0 character long, transform it to lowercase and it’s not in stopwords and also don’t have a delimiter. That’s how in the end I mange to clean the text rand yield-ing my generator tuple (category, word) and 1 as occurrence.

    def reducer\_one(self, cat\_word, counts):

        cat, wrd  = cat\_word

        yield cat\_word, sum(counts)

After that I am calculating all occurrences of that tuple in the first reducer.

At the step two of MrJob,

    def mapper\_two(self, cat\_word, counts):

        cat, wrd = cat\_word

        yield (None, (cat, wrd, counts))

I have also the mapper making a tuple datatype containing the both key from the previous step and the value.

def reducer\_two(self, key, cat\_wrd\_counts):

for i, v in enumerate(cat\_wrd\_counts):

            self.generatorList.append(v)

            ct=self.generatorList[i][0]

            wrd=self.generatorList[i][1]

            wordCounts[wrd] += self.generatorList[i][2]

            CatCounts[ct]+=self.generatorList[i][2]

            total+=self.generatorList[i][2]

In the last reducer I am calculating the chi square values.

Here I am putting from the generator inside a list, also I am storing the current category and word as well as putting the word into new dictionary as key and the number of occurrence as value while I am summing all the words. Also I am putting the categories in another dictionary and storing for each category how many words they are.

for i in range(0,count):

          ct=self.generatorList[i][0]

          wrd=self.generatorList[i][1]

        # number of words in each category \* words in this category / sum of all words

          val=wordCounts[wrd]\*CatCounts[ct]/total

        #the value of the number in the category

          values=pow((self.generatorList[i][2]-val),2)/val

          self.generatorList[i][2] = values

        #sorting by category alphabetically and then the value by descending

        self.generatorList.sort(key=lambda row: (row[0], -row[2]))

In the second loop, I have already filled the list with all the values as well as the dictionaries. Here I am calculating the chi square values. And at the end sorting the list, firstly by category and then the highest chi score.

for i in range(0,count):

            cntr=cntr+1

            ct=self.generatorList[i][0]

            if oldct!=ct:

               cntr=0

               yield "<" + oldct +"> " + (s), ""

               s=''

               oldct=ct

            if cntr<76:

                s = s + self.generatorList[i][1] + ":" +  str(self.generatorList[i][2]) + " "

                if self.generatorList[i][1] not in self.finalWords:

                    self.finalWords.append(self.generatorList[i][1])

                self.finalWords.sort()

        if s!='':

            yield "<" + oldct +"> " + (s), ""

        s=""

        yield ' '.join(map(str, self.finalWords)), ""

At the end I iterate one more time from this list but this time I am counting until 75 words for each category. As well as building finalWords list where are going to be present all the words having presence in those 75 word for each category. Sorted alphabetically.