3. Primero voy a transformar el texto crudo en un diccionario {palabra: frecuencia}. Asumo una representación en disco tal que los mappers reciben cada uno una línea del texto.

map(String line) {

```
//Hago un split del String
String[] chunks = split (line)

//Armo un diccionario {String, Int}, y cuento ocurrencias de cada palabra
wordCount = {}
for chunk in chunks
    wordCount[chunk] += 1

// emito a los reducers, itero sobre las keys del diccionario
for key in wordCount.keys():
     context.emit(key, wordCount[key])
```

}

En la próxima etapa, conceptualmente voy a tener tantos reducers como palabras distintas, cada reducer recibe una lista de conteos para una determinada palabra.

reduce(key, listCount(int)) {

```
suma = sumar todos los enteros de listCount
//emito un solo par (key,value)
context.emit(key, suma)
```

}

El próximo trabajo MapReduce arranca con un diccionario {palabra, conteo}. La salida que quiero obtener es {conteo, frecuencia del conteo}.

map(key, value) {

```
context.emit(value, 1)
```

}

reduce(key, list){

```
//list es una lista con unos, ejemplo, [1, 1 , 1]
context.emit(key, length(list))
```

}

En este punto tengo una lista de {r, Nr}, la cual se puede seguir procesando en un solo nodo. Tanto el cálculo del suavizado Zr = Nr/0.5(t-q) y el cálculo para r* son realizables en un solo nodo.