README

DIRT – Discovery of Inference Rules from Text

Zoharit Hadad 312211162 Or Hayat 312198377

Step 1:

In the first step we filter all the valid paths in the corpus, by building a tree for each line in the corpus.

Also, we calculate the parameter * slot * with counter.

Mapper:

Input: corpus.

Output:

Key:<p, s, w>

Value: n -number of occurrences.

Reducer:

Input:

Key:<p, s, w>

Output: n

Output:

Key:<p, s, w>

Value: sum of n's

Step 2:

The second step calculate the parameter ps* and pass it with n from the first step.

Mapper:

Input:

Key:<p, s ,w>

Value: n

Output:

Key:<p, s ,w>

Key:<p, s , *>

Value: n

Reducer:

Input:

Key:<p , s, *>

Key:<p , s , w>

Value: n

Output:

Key:<p, s , w>

n*= sum of ps*

Value: <n , n*>

Step 3:

The third step calculate the parameter *sw and pass it with n and n* from the first and second step.

Also, we calculate MI with n n* and n** from this step.

Mapper:

Input:

Key:<p ,s , w>

Value:<n , n*>

Output:

Key:<p , s , w>

Key:<* , s, w>

Value:<n , n*>

Reducer:

Input:

Key:<*, s, w>

Key:<p , s, w>

Value:< n , n*>

Output:

Key:<p , s ,w>

n**=sum of *sw

calculate mi= log (psw × *s*)/(ps* × *sw)

Value: mi

Step4:

From this step we start to calculate the similarity using test set.

Mapper:

Setup: load test set and build multimap with the paths <p1, p2> & <p2, p1>

Input:

Key:<p, s, w>

Value: mi

Output:

If the multimap contain path p as key then for each path pi in value:

Key:<p_pi, s, w>

Value:<p, mi>

Reducer:

Input: <p1_p2,s,w>

Output: nothing update the HashMap with numerator and denominator of (p1,p2)

Setup: load test set and build HashMap with the paths <p1, (p1,p2)> & <p2, (p1,p2)>

Cleanup: emit the similarity of the paths in the test sets calculated from the HashMap

Step 5:

In This step we sum the similarity of same keys to make them into one key
Each key got at most one non zero value so we get that value as the new value of the key.

Mapper:

Input:

Key<p1_p2,s>

Value: similarity of the key

Output:

Key<p1_p2,s>

Value: similarity of the key

Reducer:

Input:

Key<p1_p2,s>

Value: similarity of the key

Output:

Key<p1_p2,s>

Value: sum of the similarity calculation of step 4,to remove duplicate zeroes keys.

Step6:

In this step we calculate the final results the similarity of the two paths in each pair in the test set.

Mapper:

Input:

Key:<p1_p2, s>

Value: similarity

Output:

Key:<p1_p2, s>

Value: similarity

Reducer:

Input:

Key:<p1_p2, s>

Value: similarity

Output:

Key:<p1_p2>

Value: S(p1_p2)

Memory usage:

In order to calculate the mi table, we don't have many memory assumptions we assume we can hold small number of counters. But to archive that assumption we send in step 2 p s * as an extra key from the mapper to the reducer and in step 3 we send * s w in addition to sending the key p s w to count the required parameter.
In order to calculate Sim we assume we can hold the test set in the memory of the mappers and the reducers this allow us to filter the required pairs of mi's that we need to use them in order to calculate sim.