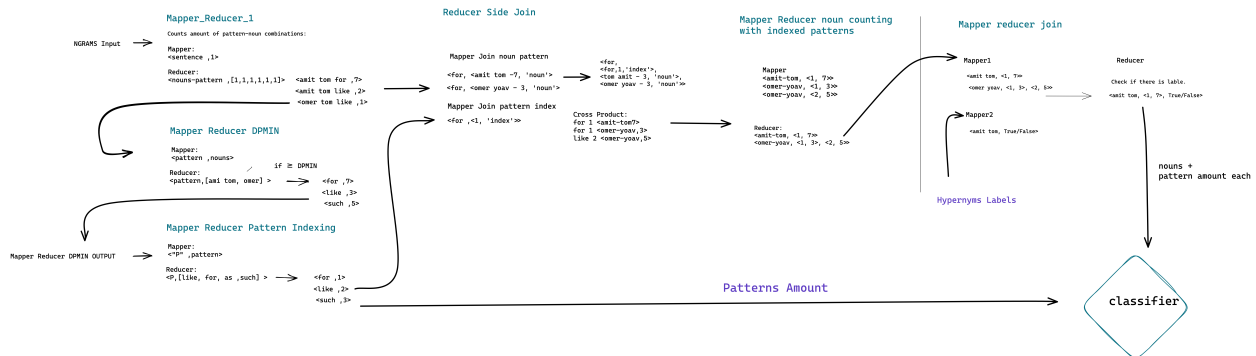


Summary Assignment 3



Design Overview :

The system is an Hadoop-based map-reduce system that analyzes a corpus of sentences and provides labeled vectors each represent the frequency of patterns that occur between pairs of nouns in the corpus. Those vectors are inserted into a classifier which trains on this labeled data and when given two nouns with pattern between them results a boolean answer rather those two nouns are hypernyms.

The input for the system is labeled hypernyms and google's Ngrams biarcs corpus.

The system is built using 6 Map-Reducer steps and a classifier DecisionTree algorithm, which we will describe below.

Manual running:

Map Reduce

1. Run mapper reducer 1.
2. Run DPMIN.
3. Run Pattern Indexer.
4. Run Mapper Reducer JOIN.

Classifier

1. CD to classifier
2. run pipenv shell
3. run python classifier.py

5. Run Mapper Reducer -Nouns.
6. Run Mapper Reducer- Labels.

Mapper Reducer 1

Given biarcs ngrams , output the amount of occurrence for each nouns pattern combination in the corpus.

Mapper Input	Mapper Output	Reducer Input	Reducer Output
Ngrams biarcs	<nouns pattern ,1 >	<nouns-pattern , [1,1,1,1,1,1]>	<nouns-pattern ,8>

Amount of key-value pairs: one for each sentence.

Reducer input key-value pairs: 2601458

Mapper input key-value pairs: 18695161

Mapper output key-value pairs: 2601458

Committed heap bytes: 43100667904

Mapper - Reducer 2

Given amount of occurrences for each nouns pattern combination ,outputs list of patterns which qualifies DPMIn (amount of unique nouns related to each pattern)

DPMIn = 2

Mapper Input	Mapper Output	Reducer Input	Reducer Output
<nouns-pattern ,amount>	<pattern , nouns >	<pattern ,[nouns1, nouns2, nouns3]>	<pattern ,3>

Amount of key-value pairs: $N_{patterns}$

Reducer input key-value pairs: 813577

Mapper input key-value pairs: 813577

Mapper output key-value pairs: 813577

Committed heap bytes: 687865856

Mapper - Reducer 3

Given the qualified patterns list, indexing the patterns using global context counter.

Mapper Input	Mapper Output	Reducer Input	Reducer Output
<pattern ,3>	<"P" , pattern>	<"P" ,[pattern1, pattern2, pattern,3]>	<pattern ,index>

Amount of key-value pairs: One for each pattern.

Reducer input key-value pairs: 17440

Mapper input key-value pairs: 17440

Mapper output key-value pairs: 17440

Committed heap bytes: 545259520

Mapper - Reducer 4

Two mappers each with it's own tag(index/noun) .The first one is getting the nouns-pattern combination with it's correlated amount with the specific pattern (Mapper-Reducer 1)

And the second is getting the patterns indexed.

This program outputs each nouns amount with it's correlated pattern index.

Mapper Input	Mapper Output	Reducer Input	Reducer Output
<nouns-pattern ,8>	<pattern , <nouns-amount, 'noun'>>	<pattern , [<pattern- index, 'index>, <nouns1- amount, 'noun'>,<nouns2- amount, 'noun'>]	<index ,<nouns1-amount>> <index , <nouns2-amount>> <index2 ,<nouns1-amount>> <index2 , <nouns3-amount>>
<pattern ,index>	<pattern ,<index, 'index'>>		

Amount of key-value pairs: $N_{patterns} * N_{correlated-nouns} + N_{Patterns}$

Reducer input key-value pairs: 831017

Mapper input key-value pairs: 831017

Mapper output key-value pairs: 831017

Committed heap bytes: 1350565888

Mapper -Reducer 5

Joining pattern indexed with a list of correlated nouns, each with their occurrence amount with this pattern

The mapper is given the index to nouns amount calculated in Reducer 4.

Mapper Input	Mapper Output	Reducer Input	Reducer Output
<index ,<nouns1-amount>>	<nouns1 , <index-amount> ,	<noun1 , [<index1-amount> , <index2-amount>]	<noun1, [<index1-amount> , <index2-amount>] <noun2, [<index1- amount> , <index3- amount>]

Amount of key-value pairs: N_{nouns}

Reducer input key-value pairs: 783456

Mapper input key-value pairs: 783456

Mapper output key-value pairs: 783456

Committed heap bytes: 889192448

Mapper -Reducer 6

Given a list of nouns with their indexed patterns and correlated amount, outputs every row nouns with it's label if exists in the labeled file.

Mapper Input	Mapper Output	Reducer Input	Reducer Output
<nouns2, [<index1-amount> , <index3-amount>]>	<nouns2 , [<index1-amount> , <index3-amount>]>	<noun2 , [<index1-amount> , <index2-amount> , label]	<noun2 , [<index1-amount> , <index2-amount> , label]
<nouns2, label>	<nouns1, label>		

Amount of key-value pairs: N_{nouns}

Reducer input key-value pairs: 691141

Mapper input key-value pairs: 691141

Mapper output key-value pairs: 691141

Committed heap bytes: 1004535808

Classifier

With the given output from Reducer 6 we created a vector for each noun where each index i of the vectors represents the amount of times the noun is appeared in the corpus with the pattern i . if the noun did not appear with pattern i the value will be 0.

$$1 \leq i \leq \text{amount of patterns}$$

We trained the model and tested it using 10X validations.:

Results

The model was trained on the file biarcs 23-of-99 from google's NGram.

There were 17440 patterns that crossed the DPMIn threshold, therefore each noun vector is of size 17440.

Attached are the results after 10 fold cross validation:

Accuracy: 0.82 (+/- 0.02)

Precision: 0.86

Recall: 0.99

F1 score: 0.92

Accuracy - It measures the percentage of correct predictions made by the model among all the predictions made (Correct predictions / All predictions).

Precision - It measures the percentage of positive predictions that are correct among all the positive predictions made by the model.(How many of positive prediction are actually positive).

Recall - Also known as sensitivity or true positive rate, it measures the percentage of positive instances that are correctly predicted by the model. (How many positive labeled are predicted positive by the model).

F1 score - It is the harmonic mean of precision and recall. It is a single metric that balances the trade-off between precision and recall.

Analysis:

Here are examples we wanted to analyze:

FN = ['activity capacity', 'apex region', 'balance difference', 'ball player', 'base control']

FP = ['abandonment suffering', 'ability character', 'ability resilience', 'ability structure', 'ability temperament']

FN:

activity capacity:

“for” -3 times,

“ “ - 1 time.

apex region

“over” - 1 time.

balance difference

“in” -1 time.

“between” - 1 time.

ball player:

“with” -1 time.

“ “ - 2 times.

base control:

“over” -1 time.

FP:

abandonment suffering:

” ” - 1 time

ability character:

“ ” - 16 times.

ability resilience:

“ ” - 2 times.

ability structure:

“ ” - 2 times

“of” - 1 time

ability temperament:

” ” -2 times

In most cases it seemed like FN and FP could be caused by lack of occurrences in the corpus or lack of diversity in patterns.

Nouns that were TP like “appetite desire” were rich with patterns like “for is” ,“satisfies” ,“gratify” .another example is “architect designer” which appeared with the patterns “like” and “is”. Even though those appeared with a lot of empty spaces as patterns. they had other patterns to divert the decision to one side.



