Classifying data using CELOE Algorithm (DLLearner)

Atul Bhopalsing Pundir, Sowmya Kamath Ramesh
Tejas Devaraj.
Prof. Dr. Axel-Cyrille Ngonga Ngomo.
Supervisor:Alexander Bigerl.

July 2021

1 Introduction

Our project uses the CELOE algorithm which generates the class expressions and these expressions are used for classifying individuals based on the knowledge base. From each learning problem, we provide positive individuals to the algorithm, and it sees what it fits for the class expressions. Since the reasoner is based on a closed world assumption, the negative examples need not be specified explicitly. The Project is executed to generate positive and negative individuals for both training and grading data, which are then stored in different files. Furthermore, these stored files are given as input and the class expression then generated classifies individuals accordingly. Finally, we store these results in an output file.

2 Process

Step 1-Getting data from a given TTL file and generating lp files for positive and negative examples using class : ReadTTLDataUsingOWLAPI
1. Parsing the given training and grading files.
2. Getting individuals using includesResource and excludesResource properties from each learning problem.
3. Save the generated individuals in a directory. Example :
training files -$lpfiles/train/lp\_1\_p.txt, lpfiles/train/lp\_1\_n.txt$, etc.
grade files - $lpfiles/grade/lp\_26\_p.txt, lpfiles/grade/lp\_26\_n.txt$, etc.

Step 2- Classify data using CELOE and generate results using class : DL-LearnerCELOE
1. Initializing the CELOE algorithm.
2. Getting the individuals from lp files generated in step 1.

3. Classify data based on class expression generated by the algorithm.
4. Generated results are stored in a directory for both training and grading data.

3 Code description

3.1 ReadTTLDataUsingOWLAPI.java

This class reads data from given TTL file and creates file for each learning problem.

3.1.1 initializeParameters() method

This method helps initializing the parameters like train file name, train file path, grade file name and grade file path specified in the config.properties file. These parameters are configurable for processing different specifications by simply changing the properties in the config.properties file and we can get results for these new specifications.

3.1.2 readData() method

This particular method reads learning problem data from the input file and returns it in the form of a HashMap. The input to this method is an OWLOntology object which is used to walk through the file and get the individuals. The key in the output map is learning problem URI and the value is another Hashmap which contains details for individuals specific to include, exclude URIs. For the Value map, key is either include/exclude resource URI and value is a list of URIs specific to the resource.

3.1.3 deleteOldFiles() method

Whenever we generate individual files for learning problems, this method is called first to delete the previously generated files.

3.1.4 createNewFiles() method

As the name suggests, the method creates new files for learning problems based on the input data. In this method, we iterate over the Hashmap and generate files in a specific path as mentioned in the config.properties file for all LPs.

3.1.5 main() method

All the above methods are called the main method. Firstly, the initializeParameters () method is called which fetches all parameters from the config.properties file. Then, using OWLAPI, we load the training and grading data. For the grading and training data, the Hashmap is generated by the readData () method which is later used by the createNewFiles () method after calling the deleteOldFiles () method. After successful execution of the main method, lp files for positive and negative examples are generated for both the training and grading data. The exact input file name, path as well as output file directory are configurable and are specified in the config.properties file.

3.2 DLLearnerCELOE.java

In this class, the data generated from individual learning problems by the above mentioned class is read and then the CELOE algorithm is invoked. which performs the desired operation. Thus generating the output result file.

3.2.1 initializeParameters() method

Similar to the ReadTTLDataUsingOWLAPI.java class, here we use the same method. But in this class, we have certain additional parameters:

1. startlpnumber - this parameter exists for both training and grading data specifying the first learning problem for which the algorithm should generate the results.

2. totallps-total number of learning problems for training and grading data can be mentioned using this parameter.
3. execute-there is one parameter specific to training data (fokg.train.execute) which can be set to true or false. If false, the algorithm will not execute for training data, otherwise the algorithm will execute for both the training and grading data.

3.2.2 invokeAlgo() method

This method invokes the algorithm. It initializes the knowledge source, reasoner and learning problem, and starts the algorithm for every learning problem. The maximum execution time for every learning problem is set to 30 seconds. After 30 seconds, the algorithm terminates and the class expression generated by the algorithm is captured. Class expression generated is used for the classification of individuals in learning problems and the result is saved for later use. Note that the algorithm will be triggered in multiple iterations for every learning problem present in training and grading data.

3.2.3 createOutputFile() method

This method generates the output file based on an input Hashmap which contains the classification result for each learning problem. The output file is generated in a path specified in the config.properties file.

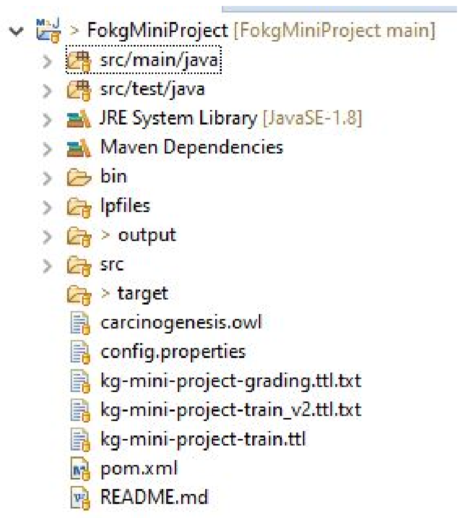
3.2.4 main() method

Firstly, the initializeParameters () method is called and the parameters specified in the config.properties file are fetched. If the execute flag is true in config.properties, the algorithm is then invoked for training data by calling the invokeAlgo () method and the output file will be generated for it by the createOutputFile () method. Lastly, for grading data, an algorithm is executed and generates the corresponding result file.

4 Execution

4.1 Project structure

It is a maven project which uses OWLAPI and DLLearner. Thus, all the required dependencies are downloaded automatically by maven. Apart from the usual maven project files, the project consists of config.properties, carcinogenesis.owl, grading, and training files. And it also includes directories like lp files and output, which store input and result files.



```
> FokgMiniProject [FokgMiniProject main]
    src/main/java
    src/test/java
    JRE System Library [JavaSE-1.8]
    Maven Dependencies
    bin
    lpfiles
    > output
    src
    > target
    carcinogenesis.owl
    config.properties
    kg-mini-project-grading.ttl.txt
    kg-mini-project-train_v2.ttl.txt
    kg-mini-project-train.ttl
    pom.xml
    README.md
```

Figure 1: Project Structure

4.2 Configure config.properties

This configuration for the given training and grading datasets already exists in the file, thus can be skipped. It consists of the following parameters.

4.2.1 fokg.train.filename

File name of the training data. For this mini-project, it's configured as kg-mini project-trainv2.ttl.txt. We can change this name for processing some other training data files.

4.2.2 fokg.train.filepath

It specifies the file path for LP files generated for training data. The directory specified here is lpfiles/train, and it's present in the project home directory. Under this path, all generated files are present which are specific to LPs for training data.

4.2.3 fokg.train.startlpnumber

First lp number for training data. It's configured to 1. Since for training data, the first learning problem is 1. However, if we want to execute from a specific lp, it can be changed.

4.2.4 fokg.train.totallps

Parameter for specifying total number of LPs for training data. This parameter is set to 25 since there are 25 LPs in training data. It can be changed, along with the fokg.train.startlpnumber parameter for debug purposes. For instance, if there is some problem with lp number - 8 and we want to re-run only this specific lp. We can set fokg.train.startlpnumber = 8 and fokg.train.totallps = 1, and the algorithm is invoked only for lp 8.

4.2.5 fokg.train.execute

Flag to execute the algorithm for training data. For training data, the configuration of execution of the algorithm is possible with this flag. If it's true, then the algorithm is executed for training data as well. However, if it's false, then the execution of the algorithm is skipped for training data.

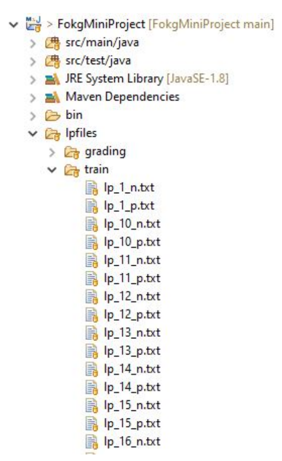
4.2.6 fokg.train.output.filedetail

Output file for training data. This parameter is set to output/trainresult.ttl, and this complete path is present in the project home directory.

Similarly, we have parameters for grading data. They have the same descriptions as it was for training data. These parameters are fokg.grade.filename, fokg.grade.filepath,fokg.grade.startlpnumber, fokg.grade.totallps and fokg.grade.output.filedetail.

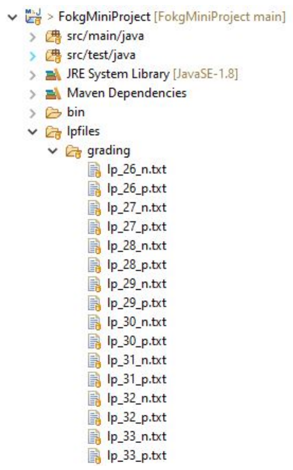
4.3 Execute ReadTTLDataUsingOWLAPI

For the generation of the LP files, this class is used. The training data under lpfiles/train path and grading data under lpfiles/grade path is generated as depicted in the figure. 2 and 3.



```
FokgMiniProject [FokgMiniProject main]
  src/main/java
  src/test/java
  JRE System Library [JavaSE-1.8]
  Maven Dependencies
  bin
  lpfiles
    grading
    train
      lp_1_n.txt
      lp_1_p.txt
      lp_10_n.txt
      lp_10_p.txt
      lp_11_n.txt
      lp_11_p.txt
      lp_12_n.txt
      lp_12_p.txt
      lp_13_n.txt
      lp_13_p.txt
      lp_14_n.txt
      lp_14_p.txt
      lp_15_n.txt
      lp_15_p.txt
      lp_16_n.txt
```

Figure 2: LP files for training data



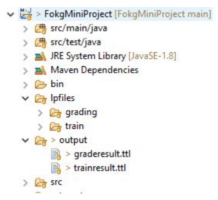
```
FokgMiniProject [FokgMiniProject main]
  src/main/java
  src/test/java
  JRE System Library [JavaSE-1.8]
  Maven Dependencies
  bin
  lpfiles
    grading
      lp_26_n.txt
      lp_26_p.txt
      lp_27_n.txt
      lp_27_p.txt
      lp_28_n.txt
      lp_28_p.txt
      lp_29_n.txt
      lp_29_p.txt
      lp_30_n.txt
      lp_30_p.txt
      lp_31_n.txt
      lp_31_p.txt
      lp_32_n.txt
      lp_32_p.txt
      lp_33_n.txt
      lp_33_p.txt
```

Figure 3: LP files for grading data

4.4 Execute DLLearnerCELOE

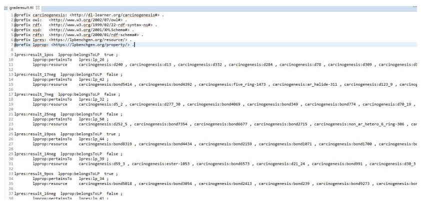
Invokes the CELOE algorithm for each lp of training (if fokg.train.execute flag is true) and grading data. After execution completion, the output file is created, which can be found under the output directory as shown in the figure. 4.

For training data, trainresult.ttl file is generated, and for grading data, we have graderesult.ttl as depicted under the figure. 5 and 6.



```
FokgMiniProject [FokgMiniProject main]
  src/main/java
  src/test/java
  JRE System Library [JavaSE-1.8]
  Maven Dependencies
  bin
  lpfiles
    grading
    train
  output
    graderesult.ttl
    trainresult.ttl
  src
```

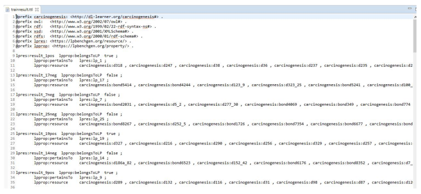
Figure 4: Output files generated in output directory



graderesult.ttl

```
@prefix carcinogenesis: <http://dl-learner.org/carcinogenesis#> .
@prefix owl:   <http://www.w3.org/2002/07/owl#> .
@prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd:   <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix lpres: <https://lpbenchgen.org/resource/> .
@prefix lpprop: <https://lpbenchgen.org/property/> .

lpres:result_1pos  lpprop:belongsToLP  true ;
        lpprop:pertainsTo   lpres:lp_26 ;
        lpprop:resource     carcinogenesis:d240 , carcinogenesis:d13 , carcinogenesis:d332 , carcinogenesis:d284 , carcinogenesis:d78 , carcinogenesis:d309 , carcinogenesis:d1

lpres:result_17neg  lpprop:belongsToLP  false ;
        lpprop:pertainsTo   lpres:lp_42 ;
        lpprop:resource     carcinogenesis:bond5414 , carcinogenesis:bond4392 , carcinogenesis:five_ring-1473 , carcinogenesis:ar_halide-311 , carcinogenesis:d123_9 , carcino

lpres:result_7neg  lpprop:belongsToLP  false ;
        lpprop:pertainsTo   lpres:lp_32 ;
        lpprop:resource     carcinogenesis:d5_2 , carcinogenesis:d277_30 , carcinogenesis:bond4069 , carcinogenesis:bond349 , carcinogenesis:bond774 , carcinogenesis:d70_19 ,

lpres:result_25neg  lpprop:belongsToLP  false ;
        lpprop:pertainsTo   lpres:lp_50 ;
        lpprop:resource     carcinogenesis:d252_5 , carcinogenesis:bond7354 , carcinogenesis:bond6677 , carcinogenesis:bond2715 , carcinogenesis:non_ar_hetero_6_ring-306 , car

lpres:result_19pos  lpprop:belongsToLP  true ;
        lpprop:pertainsTo   lpres:lp_44 ;
        lpprop:resource     carcinogenesis:bond8319 , carcinogenesis:bond4434 , carcinogenesis:bond2159 , carcinogenesis:bond1071 , carcinogenesis:bond1700 , carcinogenesis:bo

lpres:result_14neg  lpprop:belongsToLP  false ;
        lpprop:pertainsTo   lpres:lp_39 ;
        lpprop:resource     carcinogenesis:d59_3 , carcinogenesis:ester-1053 , carcinogenesis:bond6573 , carcinogenesis:d21_24 , carcinogenesis:bond991 , carcinogenesis:d30_3

lpres:result_9pos  lpprop:belongsToLP  true ;
        lpprop:pertainsTo   lpres:lp_34 ;
        lpprop:resource     carcinogenesis:bond5018 , carcinogenesis:bond3054 , carcinogenesis:bond2413 , carcinogenesis:bond239 , carcinogenesis:bond9273 , carcinogenesis:bor

lpres:result_16neg  lpprop:belongsToLP  false ;
        lpprop:pertainsTo   lpres:lp_41 ;
```

Figure 5: Result file for grading data



trainresult.ttl

```
@prefix carcinogenesis: <http://dl-learner.org/carcinogenesis#> .
@prefix owl:   <http://www.w3.org/2002/07/owl#> .
@prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd:   <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix lpres: <https://lpbenchgen.org/resource/> .
@prefix lpprop: <https://lpbenchgen.org/property/> .

lpres:result_1pos  lpprop:belongsToLP  true ;
        lpprop:pertainsTo   lpres:lp_1 ;
        lpprop:resource     carcinogenesis:d318 , carcinogenesis:d247 , carcinogenesis:d38 , carcinogenesis:d36 , carcinogenesis:d237 , carcinogenesis:d235 , carcinogenesis:d2

lpres:result_17neg  lpprop:belongsToLP  false ;
        lpprop:pertainsTo   lpres:lp_17 ;
        lpprop:resource     carcinogenesis:bond5414 , carcinogenesis:bond4244 , carcinogenesis:d123_9 , carcinogenesis:d323_25 , carcinogenesis:bond5241 , carcinogenesis:d100_

lpres:result_7neg  lpprop:belongsToLP  false ;
        lpprop:pertainsTo   lpres:lp_7 ;
        lpprop:resource     carcinogenesis:bond2031 , carcinogenesis:d5_2 , carcinogenesis:d277_30 , carcinogenesis:bond4069 , carcinogenesis:bond349 , carcinogenesis:bond774

lpres:result_25neg  lpprop:belongsToLP  false ;
        lpprop:pertainsTo   lpres:lp_25 ;
        lpprop:resource     carcinogenesis:bond8267 , carcinogenesis:d252_5 , carcinogenesis:bond1726 , carcinogenesis:bond7354 , carcinogenesis:bond6677 , carcinogenesis:bond

lpres:result_19pos  lpprop:belongsToLP  true ;
        lpprop:pertainsTo   lpres:lp_19 ;
        lpprop:resource     carcinogenesis:d327 , carcinogenesis:d216 , carcinogenesis:d290 , carcinogenesis:d256 , carcinogenesis:d329 , carcinogenesis:d257 , carcinogenesis:

lpres:result_14neg  lpprop:belongsToLP  false ;
        lpprop:pertainsTo   lpres:lp_14 ;
        lpprop:resource     carcinogenesis:d186a_82 , carcinogenesis:bond6523 , carcinogenesis:d152_42 , carcinogenesis:bond6176 , carcinogenesis:bond8352 , carcinogenesis:d7_

lpres:result_9pos  lpprop:belongsToLP  true ;
        lpprop:pertainsTo   lpres:lp_9 ;
        lpprop:resource     carcinogenesis:d289 , carcinogenesis:d132 , carcinogenesis:d116 , carcinogenesis:d31 , carcinogenesis:d98 , carcinogenesis:d87 , carcinogenesis:d12
```

Figure 6: Result file for training data

5 Conclusion

In conclusion, to classify individuals, there are other algorithms in DL-Learner like OCEL, PCELOE, etc., along with different learning problems like PosNegLP-Standard, etc. They gave poor classification results. However, CELOE gave better results than any of the other combinations of learning problems and algorithms. For CELOE, we have an execution period of around 24 minutes for each grading and training data.

References

[1] https://github.com/SmartDataAnalytics/DL-Learner

[2] https://jens-lehmann.org/files/2011/celoe.pdf

[3] https://dl-learner.org/Resources/Documents/manual.pdf