Welcome to CraftML4Txt Manual v01F

July, 2020
Author: Frank Meyer – Orange Labs France

Presentation

CraftML4Txt is an extension to CraftML to use multi-label modelling directly on text data.
CraftML is a forest-based Extreme Multi-Label (XML) learning algorithm, in open source
See Annex if you want to use CraftML for datamining or Extrem Multi Label benchmarks.
See: https://github.com/Orange-OpenSource/OpenCraftML for complete open-source ressources

With this extension, you can learn, predict, and evaluate models directly on text UTF8 (without BOM) format.

In order to run CraftML4Txt, use Eclipse, import the project, and run the class CraftMLText_API_ScriptInterpretor in the textModule package, with a script file given in parameter.
Be careful: the program use Java features available only in **Java version 1.8 or +**.

In order to generate a running jar, use Eclipse, import the project. Then clic-right on the project -> export-> as a running jar and use as launch application: the class CraftMLText_API_ScriptInterpretor

CraftML4Txt executes script-files (not in the same format as for the native, tabular and numeric-oriented, format).
An example with comment of a script file is given in the next section.
CraftML4Txt can create model from text files (UTF8 without BOM), examples are given in the data format section

If you want to use directly the jar program:
You will need **Java version 1.8** or more installed on your computer), simply type:

```
 java  –jar  CraftML4Txt.jar  the_path_of_your_script_file
```

Using the sandbox directory

If you are using Windows, you can simply use the sandbox directory Sandbox4Txt to learn how to use CraftML4Txt.

Just unzip the Sandbox4Txt.zip at the root of your C: disk to obtain the C:\sandbox4Txt\ directory

You can then use directly the CraftML4txt.jar contained in this directory, with several examples.

After reading this documentation, we recommend you first to

1. Take a look at the datasets LittleTrain.txt and LittleTest.txt
2. Look then at the: littleScript.txt and execute it with the command:
   ```
   java –jar CraftML4Txt.jar littleScript.txt
   ```
3. Then take a look at the generated files :

a.	littleTest.txt.predict.txt :	label prediction only on specified labels in test file
b.	littleTest.txt.evalPrecision.txt:	full label prediction and evaluation
c.	littleTest.txt.rapport.txt	performance repport

… Other datasets and scripts are available in the sandbox, and you can modify them!

The script file format

The script file must be a text file in UTF8 format
BE CAREFUL not to use a text editor that adds a “BOM” character at the beginning of an UFT8 file, such as the Notepad editor of Windows ™. Use Notepad++ for instance (a free open source editor with easy file encoding management).

The Script file generally has 2 sections
The first part is the parameters for CraftML4Txt
The second part deals with the learn/predict/eval commands, and with save/load model commands.
Every command as 2 parts: the command itself and the parameter, separated by a least one space
If a line begins by //, it will be considered as a comment

Exemple of a full script, with commentaries

```
// BEGINNING of the CraftML4Txt Script...
// When a line begins by “//” it is considered as a commentary
//
// ================  Paramaters for the craftml4txt model ==============
//
//    global trees’ parameters --
//
// set the number of trees; recommended values to start: 10; to improve performance: 100 or more:
setNumberOfTrees 10
//
// set the number max of sub nodes (sons) for a node:
setBranchFactor 5
//
//    --  parameters for the random projections at each node --
//
// set the dimension of the projection of feature space; it is better to set above 2000 or more:
setDimX 2000
// set the dimension of the projection of the label space;
// except for case with few labels, set above 1000:
setDimY 1000
//
// set the size of the reservoir sampling, that is to say the reservoir of sampled instances used for each node
//   when the clustering and the separators are performed.
// The larger the better, but to avoid memory issues, it’s better to set it between 2000 and 10000:
setSizeReservoir 10000
//
// Set the number of used processor’s threads; if your computer has many cores,
// with this parameter, you can go faster if you use many cores (for instance, 4):
// if you want to use the computer for other tasks, one the other hand, keep at least 1 core free
setNumberOfThreads 4
//
// Set the number max of labels stored in each leaf of each tree; usual values are between 10 and 50:
setTopNMax 50
//
// Set the number min of instances necessary to decide to split the current node; use between 5 and 10:
setMinInstancesInLeaf 5
//
// Set the number of features that will be kept after having computed the separators of each node
// set between 1000 and 2000:
setSparsity 1000
```

```
//
//     --------    text parsing options ---------------
//
//  size max of the word-ngrams: all the word ngram between 1 and the value will be generated:
// minimum value is 1
setCharNgrams 3
//
// size max of the letter-ngrams; all the letter-ngrams between 1 and the value will be generated
// if set to zero, no letter ngrams are generated:
setWordNgrams 3
//
// remove the usual punctuation symbols true/false
remove punctuation false
//
//
// =============== Commands for the craftml4txt model ===============
//
// create, init with the previously seen parameters and the learn a model with the specified text file
learn       C:\\Sandbox4Txt\\littleTrain.txt
//
// save the computed model to the specified directory (be careful, a directory path is needed, not a pure file path!)
// note that the symmetric command for “save” is load, which allows to load a previously saved model
save       C:\\Sandbox4Txt\\MyModels\\ModelForLittleTrain
//
// apply the model to specified new text data to predict the labels;
// this text data do not need to have labels
// BUT if labels are present in the data, then only these labels’ probabilities will be predicted
//    otherwise, the k most likely labels are predicted, with their associated probabilities
// a file using the input file name but with the extension “.predic.txt” will be generated as a result:
predict    C:\\Sandbox4Txt\\littleTest.txt
//
// apply the model to the specified test file (a text data that contains the ground-truth labels for each instance)
// 2 files will be generated:
//    1 file using the name of the test file, with the extension “.eval.txt” for the prediction
//    1 file using the name of the test file, with the extension “.rapport.txt” to give the precision@k performances
eval    C:\\Sandbox4Txt\\littleTest.txt
//---


// END of the CraftML4Txt Script...
```

The data file format

As for the script file format, the data file format but be a text file with an UT8 encoding.

Again, BE CAREFUL not to use a text editor that adds a "BOM" character at the beginning of an UFT8 file, such as the Notepad editor of Windows ™. Use Notepad++ for instance (a free open source editor with easy file encoding management).

If you generate or re-encode your text datasets, BE CAREFUL to use an "UTF8 without BOM output" encoding (available in every usual programming language such as Python, Java, C,…).

The data file syntax for train set and test set is the same.
Note that for train and for test set, you will need text file containing full examples, with a text part and a label part

This is an example of multi label example => English ; example ; easy_to_understand

Specific constraints

BE CAREFUL, also, that several symbols have specific meaning in CraftML4Txt's dataset:

• The implication symbol:	"=>"	(using equals character and greater than character)
• The semicolon:	";"	
• The comma	","	
• The pound symbol	"#"	
• The double slash:	"//"	

For each instance, the symbol "**=>**" will be use as a separator between the text input part and the label part
For each instance the semicolons are used as label separators (you can also use blank or tabulation)

For example, the following record:

please Eliza can you turn on the living room light => home_automation ; light_on ; living_room

uses the "=>" to separate the text from the list of labels
and uses the semicolons as label separators.

The valid separators for the labels are: semicolon, comma, tabulation, space.
Note that punctuation signs (and + : -) are separated from label nouns:
for instance,"sound+" will be interpreted as 2 labels, "sound" and "+"

In data files, every line beginning with # will be considered as a comment and will NOT be processed neither in Train sets nor in test or prediction sets.

It is not recommended to use "//" in data file. Indeed, the "//" will be used by CraftML4Txt as a its comment, in conjunction to the tabulation characters, to separate the initial records with its prediction:
tabulation character + "//" + tabulation character after the record and after writing the predicted label
tabulation character + "//" + tabulation character after the predicted label and before the predicted cluster
This syntax will make it easier to post-process the predicted file, or even to copy/paste a predicted file on a spreadsheet like Excel ™.

Example of train dataset, with comments
(remember to save de data file as text file encoded in UTF8 (without BOM)

```
# ----- ----------------------   A little Train set example for CRAFTML4TXT  --------------------------------
#
# This line is a comment.
# In a data file, every line beginning by a pound symbol is skipped by CraftML4Txt
#
# Every example must contain a text part and a label part, separated by the symbol =>
#  The labels must be separated by a semicolon, or a comma, or a tabulation, or a space
#
# start of data area:
La visualisation des vidéos est excellente           =>         FR Content
Video viewing is excellent => ENG  happy
je suis content  =>         Content  FR
I am happy        =>         Happy ENG
ça marche bien la 4g c'est super génial  =>         Content FR
It is a wonderful life       =>         ENG  Content Happy
La 4g reste problématique, ça ne marche pas chez moi !            =>         pasContent FR
4g remains problematic, it does not work with me! => ENG angry
ça marche PAS la 4g !!!!=>         pasContent FR
aucune différence c'est le bazar cette 4g           =>         pasContent
fucking network            =>         ENG angry
my tailor is rich             =>         ENG
je ferai bien encore un essai c'est cool = > FR ; content ; open
I will try again, it's cool => ENG ; cool ; open , happy
# End of the data area
#
#   -----------------  End of the Train set -----------------------
#
# A similar file (with other examples in the same label domain) could be used as Test set
# if you want to predict the labels on new data (after having trained a model),
#  just use a data file with lines of text without the => and the label parts (simple file of text)
```

Annex

Using CraftML for data-mining or Extreme Multi Label Benchmark

You still can use CraftML to benchmark XML repositories or to train/use models for data-mining tables.
You will use the classes in the package Benchmark, on you can use scripts and call the class CraftML API.
You can also generate a specific CraftML.jar program using the Main entry of the CraftML_API.java.

See the repository on Github: https://github.com/Orange-OpenSource/OpenCraftML

An example of config file is given in the main directory of the GitHub repository project: config_API_CRAFTML.txt (in the directory CraftML_ressourrces
Comment / decomment the line of a given action configuration to run it.

To run CraftML just type:

```
java -jar craftML01.jar config_API_CRAFTML.txt
```

(the config file should be in the same directory as the jar file)

To run craftML on XML (Extreme Multi label) data, you should use the -Xmx -Xms option to ask for more RAM
For instance if you can ask 32Gb of RAM, you can use:

```
java -Xms32g -XMX32g -jar craftML01.jar c:\OpenCraftML2019\config_API_CRAFTML.txt
```

(it is assumed in this example that the project has been installed on C:/openCraftML2019/...)

Please note that CraftML use UTF-8 without BOM text file format for both config files and data files.
We recommand to use an editor such as Notepad++ to check, and if necessary to encode, the right text file format.

See the readme.md page for additional information about the CraftML program.

Use the sub-project “CraftML_sources“ for the source if you want to be sure to reproduce the same results as those in the research paper (ICML 2018). Indeed, some adaptations have been done for the text version of CraftML. Otherwise, note that the craftML4Txt sub project contains CraftML and the main entries to build and execute the data-mining and benchmark-oriented program.

CraftML is a powerful random forest open source framework, for mono and multi-label datasets.

Reference

If you use CraftML for benchmark, you can use this reference:

SIBLINI, Wissam, MEYER, Frank, & KUNTZ, Pascale. CRAFTML, an Efficient Clustering-based Random Forest for Extreme Multi-label Learning. In : International Conference on Machine Learning. 2018. p. 4671-4680.

Contact

Franck.meyer@orange.com