Department of Computer Science
Technical University of Cluj-Napoca

Intelligent Systems

Radu Razvan Slavescu, Adrian Groza and Anca Marginean

Contents

Assignment 2

Detecting clause end in Romanian

2.1 Assignment objectives

For this assignment, you will have to write a program which, when given a text in Romanian, will label each period sign (.) inside it as end-of-clause marker or not. As an example, given the clause (rom. fraza) :

> Avem S. I. de la 12.00 pe str. G. Barițiu. Prezența obligatorie.↩
> Nu se admit întârzieri mari.

the green periods mark a clause end, while the red periods do not. (↩ means End of Line).

You will have to train different classifiers to achieve this task. To this end, you will have to first identify a set of features relevant for the decision and to build a set of train examples for the classifier. As an example, we might consider this very basic set of 2 features:

1. urmEOLN: Y if the period is followed by an EOLN and N otherwise
2. predPresc: Y if the period is preceded by an abbreviation (e.g., "str") and N otherwise

The eight occurences of period in the text above will lead to the data in Table 2.1, where one line corresponds to one period sign. For example, the line 7 corresponds to the period just after the "Prezenţa obligatorie" text: it is followed by an EOLN sign and is not preceded by any abbreviation.

This data, called the train data, will be fed to the program which generates the classifier. Once this is achieved, you can use it to classify some new examples.

Table 2.1: Train data

Nr.	urmEOLN	predPresc	Class
1	N	N	N
2	N	N	N
3	N	N	N
4	N	Y	N
5	N	N	N
6	N	N	Y
7	Y	N	Y
8	N	N	Y

You will need to write a program (we'll call it "example generator") which generates the content of the table, starting from raw text. This program must be delivered as part of the assignment and should be delivered in such a way that I can start and run it in no more than 1 minute (no fancy libraries or tools to be installed etc.) E.g., a `jar` file would suffice if it can be run with the `java -jar` command.

The program will have as an input a file containing a set of sentences in Romanian and as an output the content of a table like the above, in a format appropriate for the model generator (that would be the `.csv` format). Obviously, the content of the `Class/Target` column (the one we want to predict) should be written by hand for each of the examples you provide. The content of the other columns (`urmEOLN` and `predPresc` in our case), aka "the features" or "the attributes", will be automatically generated by the program written by you.

Once you have the train example set, you'll have to build the classifier(s) you want to employ. Finding the best set of parameters for this classifier and assessing its performance is part of your assignment.

The example generator will serve double duty: generate the train examples set and the test data. For the train example set, you need to deliver, for each example, both the values of the feature columns and the corresponding `Class` value. The train example will be used for generating the model. For the test example set, you also need to deliver the content of both the feature and the class column. The only difference is that, this time the `Class` column is used for assesment only: the new examples (the test example) will be classified by the model built before and the `Class` value wil not interfere in the process; the value of the `Class` will serve only to chech whether the class offered by the classifier to the current example is the correct one. Do not forget that the classifier's training and test set must be kept separate. You may consider the following text for testing your model:

> Avem meci cu C.F.R. Cluj. Va fi egal.

In short, you will have to:

1. write a generator and use it to generate a train example set, starting from a text file
2. use the training set to generate a classifier
3. fine tune the classifier parameters to achieve the best performance
4. generate a test example set, using the same generator
5. assess performance

Note: it is very likely that you will need several iterations until your model reaches a reasonable performance. While struggling for improvement, you will need to extend the example set and/or the feature set, so you generator must be flexible enough to accomodate thit sort of changes on short notice.

2.2 Deliverables

2.2.1 Data generator

This should get a text file with sentences in Romanian and generate de train data for the classifier builder. It must run when tested by your TA.

2.2.2 Train and test examples

The train and test example sets are given as two files containing raw text in Romanian. They must be disjoint.

2.2.3 Model

You either save the generated model or you provide enough information in your report (see below) for you TA to be able to re-generate it. For example, you may want to provide a script which builds the model step-by-step.

2.2.4 Report

Once the assignment is finished, you'll have to write a report which describes in detail, in separate sections, the following:

1. List each feature, together with its meaning and its set of possible values
2. Give a detailed description of the models you have chosen and of its parameters (e..g., for a neural network: the number of layers, on neurons per layer, activation function etc.)
3. Show the results you obtained when assessing its performance
4. Comment on your results, limits and possible improvements