Deteneg Application

This application detects the negations in the sentence and the scope of each one (negated words) based on machine learning and NLP techniques.

For example, in the sentence: “No cancer neither paralysis”.

1. Negation: No
   Scope: Cancer

2. Negation: Neither
   Scope: paralysis

> **<NEG1>** No **</NEG1>** **<SCOPE1>** cancer **</SCOPE1>** neither paralysis
>
> No cancer **<NEG2>** neither **</NEG2>** **<SCOPE2>** paralysis **</SCOPE2>**

The idea is to train a classifier which, for each negation particle, classifies each word of the sentence into one of the two categories:

1. Is – Inside the scope
2. Os – Outside the scope

Negation particle	Token	Class
No	cancer	Is
No	cancer	Os
No	paralysis	Os
Neither	No	Os
Neither	cancer	Os
Neither	paralysis	Is

To do this, we follow a set of steps:

1. **Training**
   First, we need to build a dataset (.arff, .csv, etc.) file with the attributes we consider necessary to classify the words. This dataset arises from two clinical documents written by medical staff about anamnesis and radiology.

   We select 15 attributes:
   1. TokensP: The negation particles
   2. POSP: The POS (Part of Speech) of the negation particles
   3. Token: The word we want to know the scope of
   4. POST: the POS of the word
   5. Distance: the number of words between the negation particle and the word
   6. Location: {pre, post, ins} To know the location of the word with regard to the negation particle
   7. POSS: Is a chain of the POS of each word between the negation particle and the word we want to know the scope of
   8. Token0: The word before the word we want to know the scope of
   9. Token1: The word after the word we want to know the scope of
   10. POSW0: The POS of the word before the word we want to know the scope of
   11. POSW1: The POS of the word after the word we want to know the scope of
   12. TypeS: A chain of the class (-bn- if it is a negation and -o- if it is not) of each word between the negation particle and the word
   13. PlaceS: Place of the sentence in the document
   14. PlaceT: Place of the token
   15. Class: The attribute we want to know:
       - Is – inside the scope
       - Os – Outside the scope

2. **Model**
   To build the model we use Weka and the IBk algorithm. So, you will need a Weka model file to use the system.

3. **Run application**
   Once we have the model, we develop the application where the user enters the sentence and that build automatically the test set containing the required parameters which will be passed to the model to classify the words.

Classification folder

In the “Config” folder you can configurate the default paths of your app, for example, the library path or the others folder paths.

Classification folder

In the “Classification” folder you will see several files:

1. Scope_clinical_test.arff
2. Scope_clinical_train.arff
3. C_scopes.txt
4. Datos.txt
5. Detect_scopes.bat
6. Modelo_scope_model

1. Dataset containing the data to classify. This file will be built automatically by the system.

2. Dataset headers used for building the model.

3. Temporary file where the result of the classification is stored {is, os}

4. Temporary file used for building the “Scope_clinical_test”

5. The script to classify the words in the Scope_clinical_test file according the model using Weka

6. Weka model, in our case, this model is built using the IBk algorithm and the data tagged training data of Hospital Care

You don´t need to change any file to use the application system.

Changing the model:

If you want to change the algorithm used to build the model you have to:

- Change the “modelo_scope_model” file at Classification folder
- Modify “detect_scopes.bat” to use your own classifier

Also, if you modify the attributes of the model, you must:

- Modify “GenerateInputClassificationARFF_Scopexotherx” class to add or delete attributes from the input sentence