daCSF v0.1

Documentation for NLP course project

Antonello Fodde
817371
antonello.fodde@mail.polimi.it

What is it about?

The name “daCSF” stands for “Dialogue Act Classifier”. By starting from a dataset of file audio, which contains recorded dialogues tagged following DAMLS dialogue act tag set, it extracts from them several characteristics. Using these characteristics, which is called “features”, it will create a model able to classify dialogue acts relying only on those extracted features. Using a cross-validation approach, it will be able to compute the accuracy of classification.

Different Component

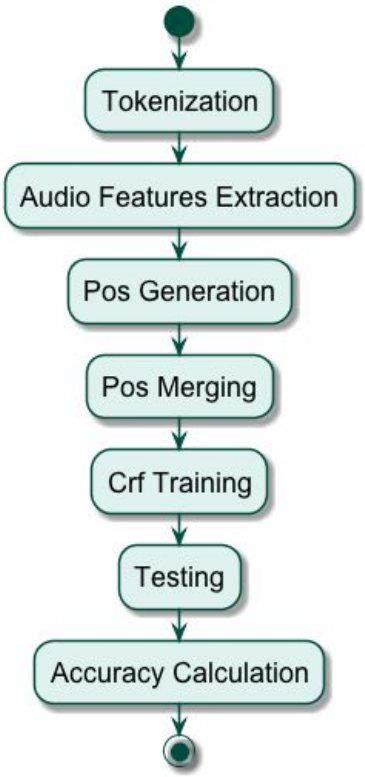
As the classification process involves several steps, each of this step will be handled by a selected tasker: each one could be an external tool or a script/program. The aim of this tool is group all different step handled by different tasker into a single unit, in order to make the process as easy as pressing a button. The list of external tool used by daCSF:

- Praat: features extraction
- Stanford POS tagger: for POS tagging
- CRFsuite: for modelizing, classification and testing
- JavaFX lib: for daCSF’s GUI

The used of each one will be clear with the specification of the process flow.

Process Flow

It is not so easy to describe the entire process, so as first thing a diagram will show the sequence of each involved step.



Tokenization

For the Pos tag generation it is necessary to create first a text file where all the dialogues have to be tokenized by sentences. The problem is that the dialogues, contained into .Textgrid files, don't contain punctuation. So for separating sentences we use silence intervals, that are tagged with '#'. The output format has a sentence for each line, as follows:

```
...
i certainly understand your dilemma
but it really need not be a problem for you at this point
well why is that
i thought i would need to know exactly what i wanted to study
when i applied for college
well some students do and others do not
most students have a general idea before they enroll
but not all students stay on the educational path they started out on
...
```

This task is made by a praat script.

Audio Feature Extraction

The audio features that are extracted are:

- Avarage Pitch
- ΔPitch
- Avarage Intensity
- ΔIntensity
- Avarage F1(first formant)
- Avarage F2(second formant)
- Avarage Harmonicity

All these feature are extracted for each word using a praat script. After that, all values are discretized. The discretization process rely on the standard deviation of each sample. After calculating the mean for each feature of the entire dataset (done by using Wolfram|Alpha Pro), the discretization follows the following formula:

$$x_d = ln(\frac{x_c - \mu}{\alpha} + 1)$$

Where x_d is the discretized value, x_c is the continuous value, μ is the feature mean, α is the scaling factor. The scaling factor is properly chosen for each feature in order to have around 6 discrete class as output.

Finally all word with their dialogue act tag and discrete feature are written into a file in the following format:

```
...
assert word=I avgPitch=2 deltaPitch=1 avgIntensity=1 deltaIntensity=0 avgF1=-3 avgF2=-4
avgHarmonicity=-1
...
```

Pos generation

Starting from the file created in the tokenization step, it use Stanford POS tagger for generate POS tags for each word, following the Penn Treebank tag set. The tagger is set with a new line as sentence delimiter. The output file format will be:

```
...
well_RB some_DT students_NNS do_VBP and_CC others_NNS do_VBP not_RB
most_RBS students_NNS have_VBP a_DT general_JJ idea_NN before_IN they_PRP enroll_VBP
but_CC not_RB all_DT students_NNS stay_VBP on_IN the_DT educational_JJ path_NN they_PRP
started_VBD out_RP on_IN
...
```

Pos merging

The goal in this step is add the generated POS tag for each word into the features file. This is done by the Java class `PosMerger` that scan line by line the features file and the POS tag file and add the tag for each word at the end of the line. The result will be like this:

```
...
assert word=I avgPitch=2 deltaPitch=1 avgIntensity=1 deltaIntensity=0 avgF1=-3 avgF2=-4
avgHarmonicity=-1 pos=PRP
...
```

Training & Testing

The user will be able to specify a number *k* for performing a *k*-fold cross validation.
So given *k*, the dataset is split into k subsets and, with CRFsuite tool, *k-1* subsets are used to train the model. After that, the *k-th* subset is used for validation using again CRFsuite. This process is repeated *k* times. The tool will write into a file several information (macro-average precision, recall, F1, item accuracy...).

Accuracy Calculation

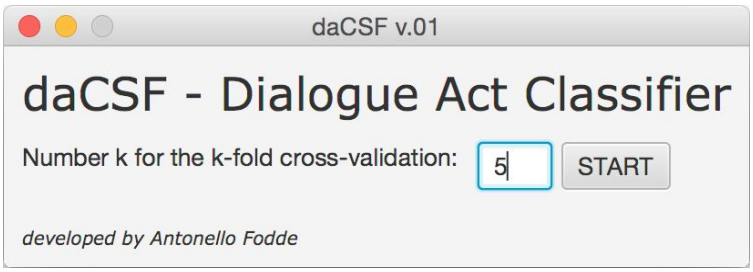
Among all the output information of CRFsuite, what we are interested in is the item accuracy. So with the Java class `AccuracyCalculator` we scan the output file and for each cross-validation we retrieve its item accuracy. By doing this for all *k* cross-validation, we also compute the total avarage accuracy. daCSF will display the result in a dialog.

Results

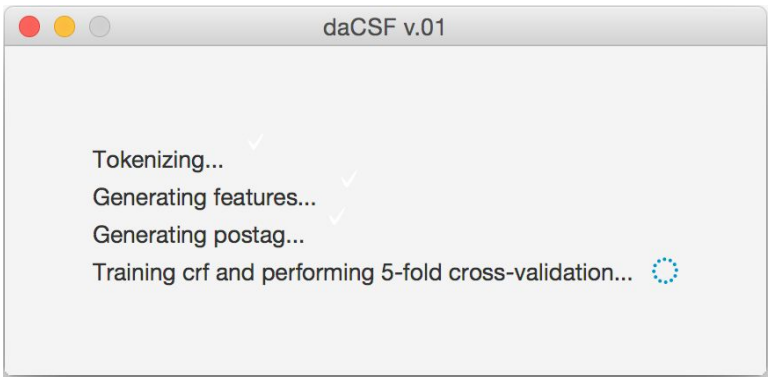
After many trials, including different features, the best result in the total average accuracy for the classifier performing a 5-fold cross-validation is 32%. The results are not so good, but one of the reasons is the size of the dataset, which is too small for sure. But the main problem probably relies on the fact that the dialogue act classification cannot rely deeply on audio feature: the increment of accuracy by adding one by one the audio features is really small. By the way the accuracy results are computed per word: by doing so, the accuracy will be of course smaller than the accuracy computed per sentence.

GUI

For an easy interaction with the user, daCSF has a GUI implemented using JavaFX libs. With the main screen the user will be able to insert a number *k* for performing a *k*-fold cross validation:



By pressing start button the process begins and the GUI will display the progress screen:



At the end of the process the accuracy results will be displayed in a dialog.