Assignment 3 – Relation Extraction

https://github.com/hajoki/EliseMichon_HW3

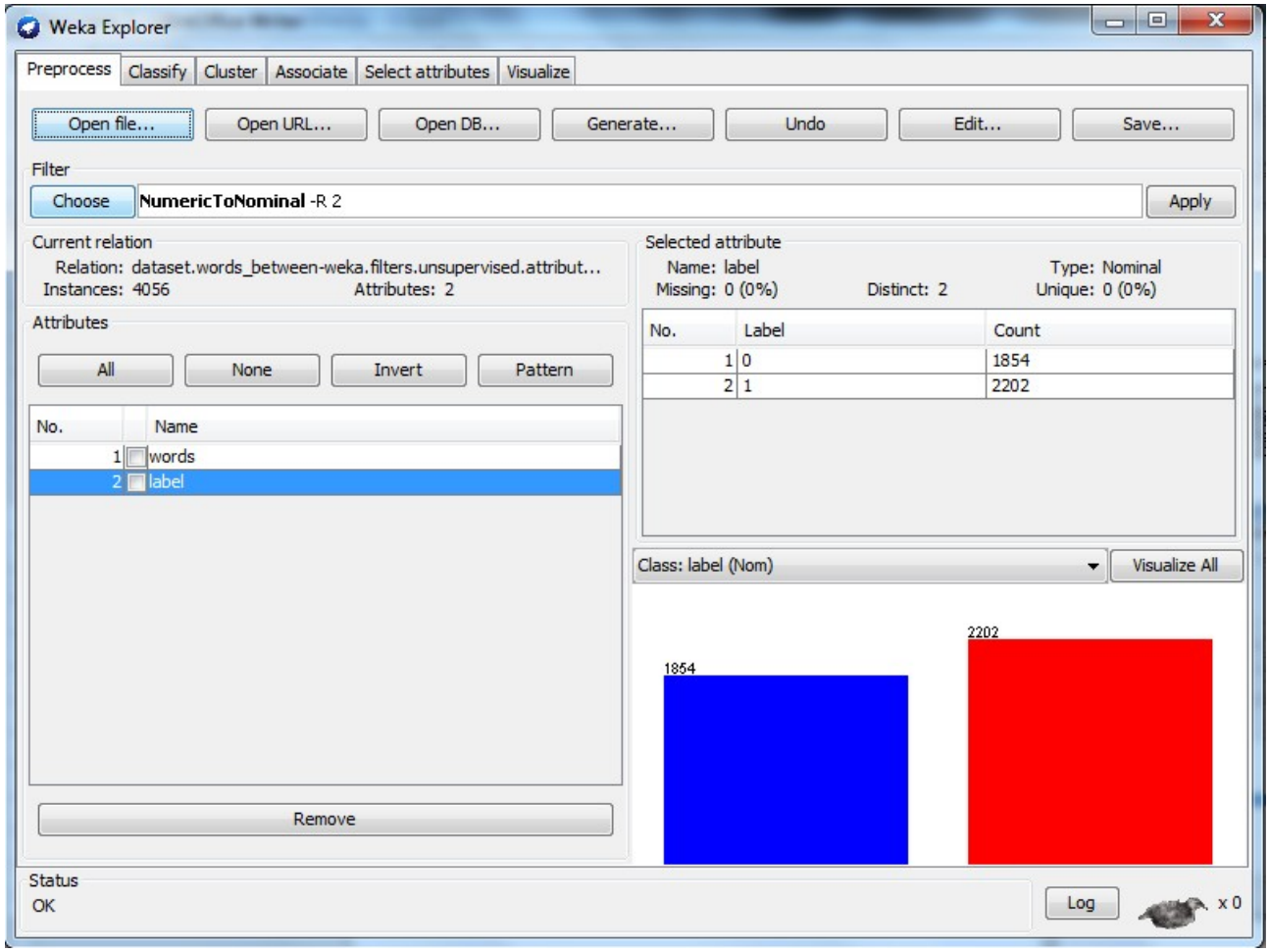
Introduction

In order to extract the relation between the protein pair PROTX1 – PROTX2 from the files *dataset.sentences, dataset.labels* and *int-keywords.txt*,
1. we generated .csv files containing each extracted feature in column, thanks to Python 2.7 scripts run in Eclipse Java Mars
2. we ran classification algorithms on these .csv files through Weka [1].
We chose decision trees as the general type of algorithm, to make it different from Naive Bayes classifiers, and the decision tree algorithm that could apply to our data and gave the best accuracy was reported below. The advantage of decision trees is that they are easily interpretable, which gave us some cues about possible improvements.

I. Baseline system

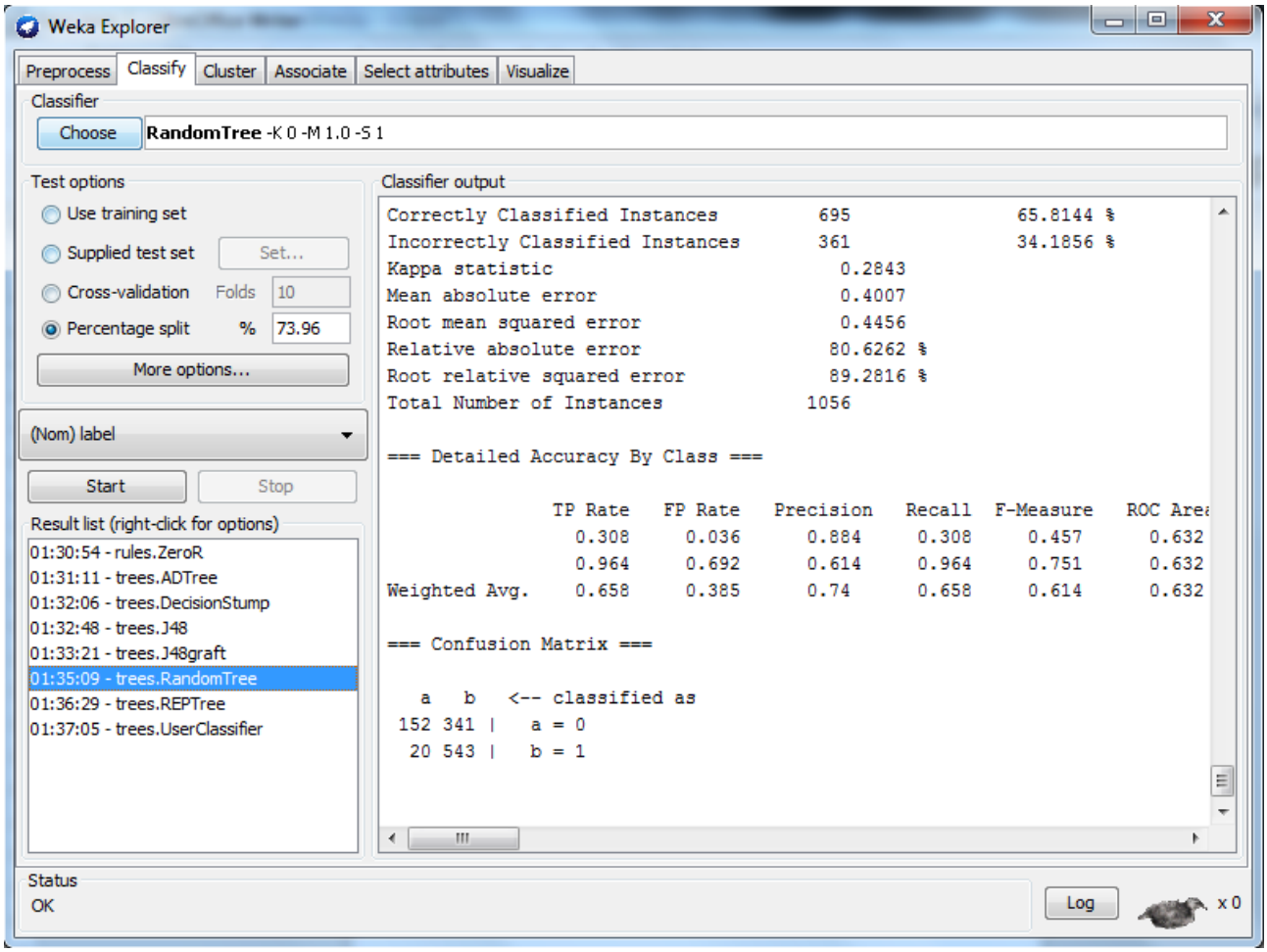
1. Features: words between the strings 'PROTX1' and “PROTX2” or vice versa

Before applying any classification algorithm, labels were converted from numeric to nominal with a preprocessing filter in Weka:



2. Machine Learning Algorithm: Random Tree

We gave to Weka files with all the 4056 instances, and performed a percentage split at 73,96% corresponding to 3000 instances in the training set and 1056 instances in the test set.



3. Performance measures over test set

Time taken to build model: 4.07 seconds

=== Evaluation on test split ===

=== Summary ===

Correctly Classified Instances	695
65.8144 %	
Incorrectly Classified Instances	361
34.1856 %	
Total Number of Instances	1056

=== Confusion Matrix ===

Classified as ->	0	1
0	152	341
1	20	543

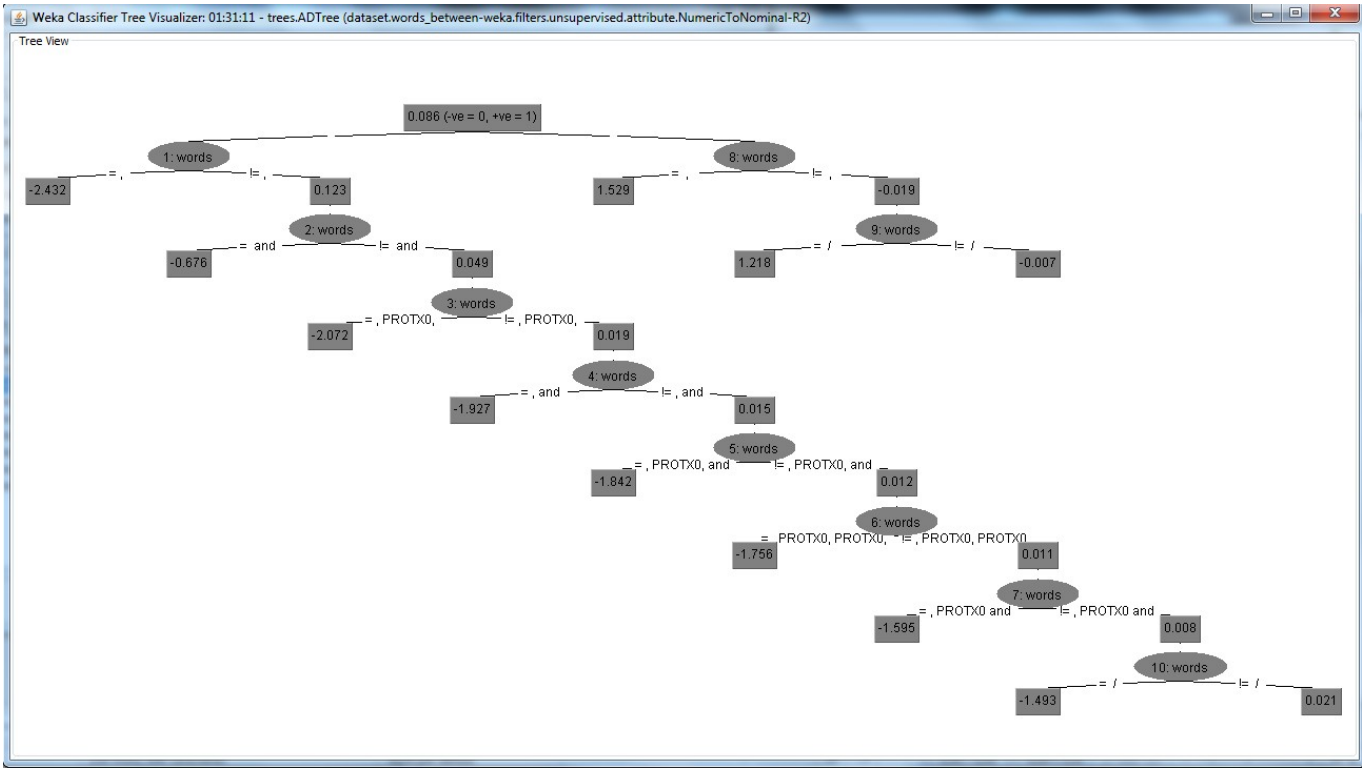
=== Detailed Accuracy By Class ===

Class	TP Rate	FP Rate	Precision	Recall	F-Measure
0	0.308	0.036	0.884	0.308	0.457
1	0.964	0.692	0.614	0.964	0.751
Weighted Avg.	0.658	0.385	0.74	0.658	0.614

4. <u>Possible improvements</u>

The vizualization of Random Tree was very unclear, suggesting a large number of branched nodes in the decision tree. However, ADTree, which gave slightly less accurate results, could build the following simple tree structure, suggesting that classification was mostly made on the basis of the presence of punctuation signs (, . - /). One could imagine they are not really relevant to determine the relationship between the two proteins and better results could be obtained if the algorithm was run on alphanumeric words only.

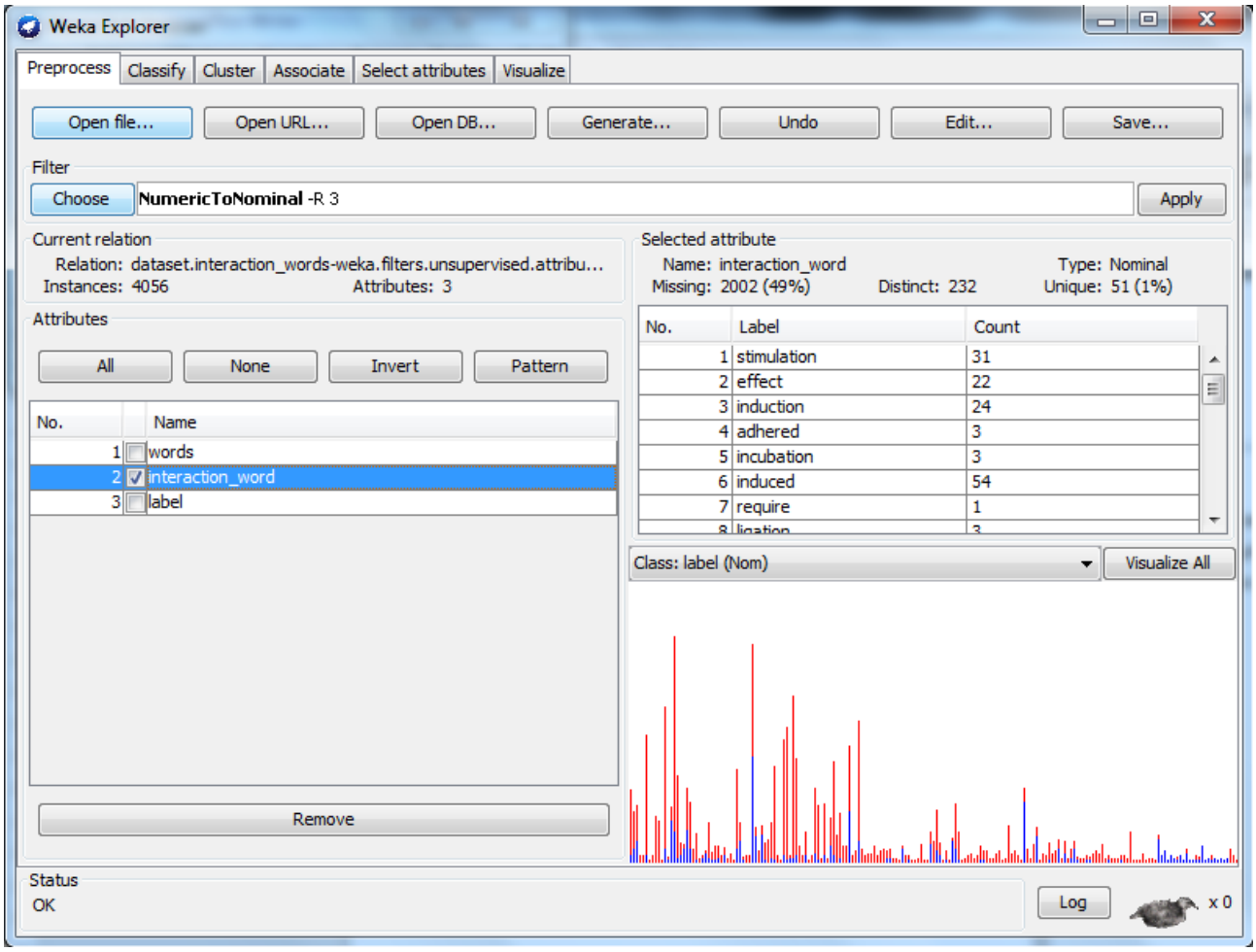
Correctly Classified Instances	663	62.7841 %
Incorrectly Classified Instances	393	37.2159 %
Total Number of Instances	1056	



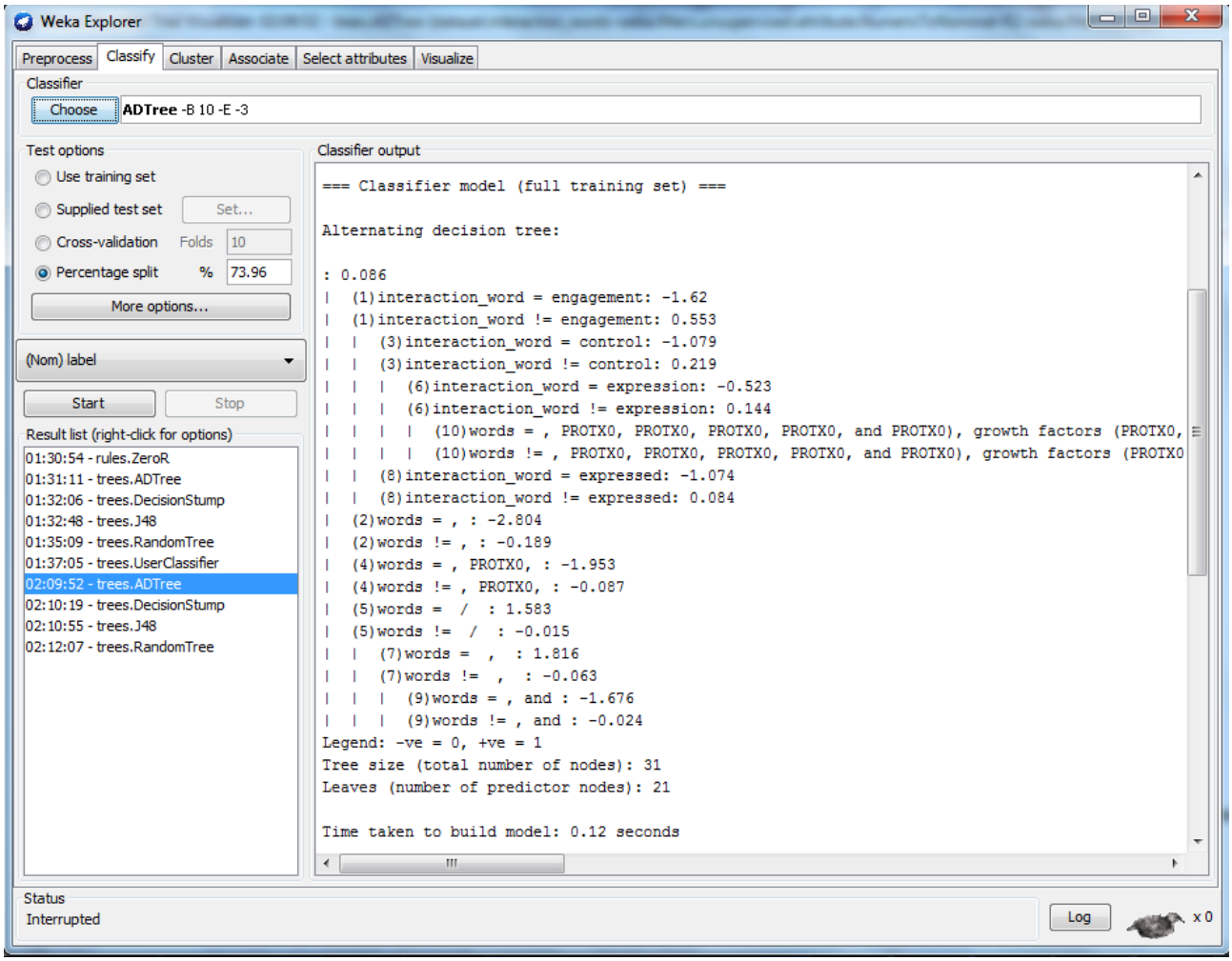
II. Advanced system

1. Features: words between the strings 'PROTX1' and “PROTX2” or vice versa, and the first interaction word of the intersection between the previously mentioned words in between and the list of interaction words

This time again, labels were converted from numeric to nominal with a preprocessing filter in Weka, which enables us to see that some of the interaction terms found in the words in between are shared between the two classes, but others tend to be present only in sentences of one class (1 - describing an interaction - red / 0 – not describing an interaction – blue).



2. Machine Learning Algorithm: ADTTree



3. Performance measures over test set

Time taken to build model: 0.12 seconds

=== Evaluation on test split ===

=== Summary ===

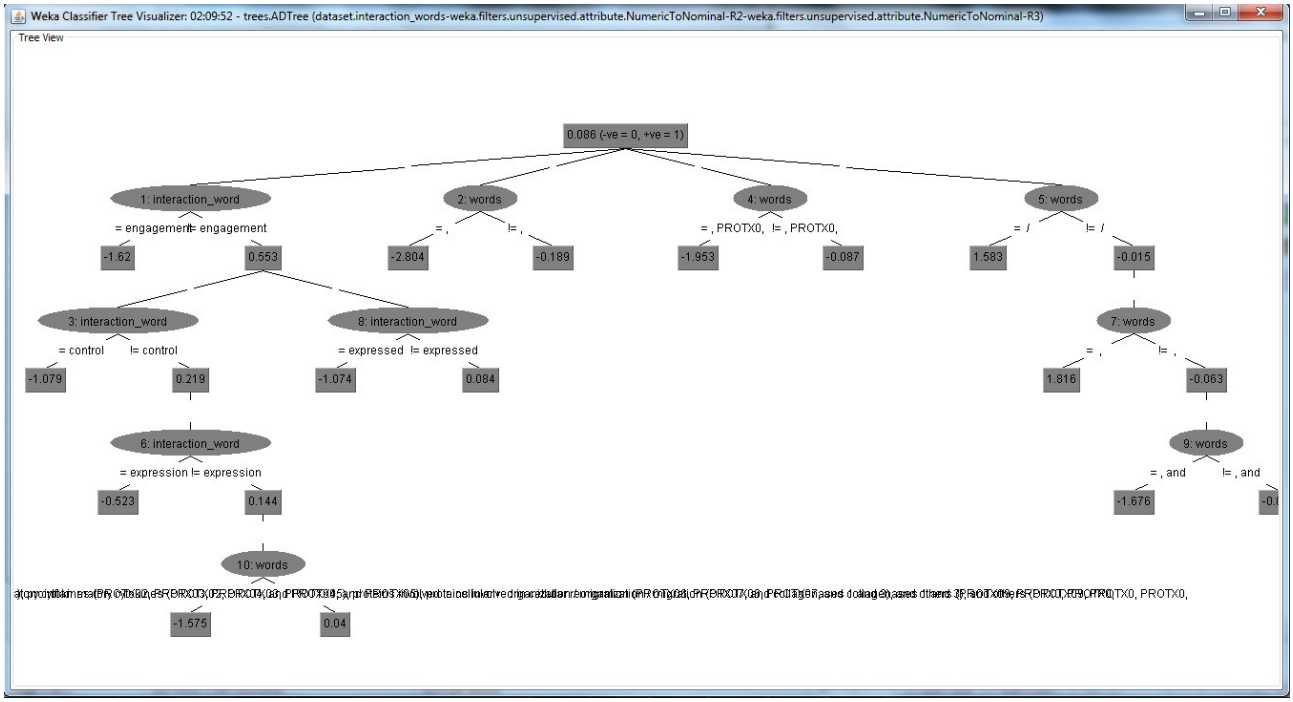
Correctly Classified Instances	809
76.6098 %	
Incorrectly Classified Instances	247
23.3902 %	
Total Number of Instances	1056

=== Confusion Matrix ===

Classified as ->	0	1
0	391	102
1	145	418

=== Detailed Accuracy By Class ===

Class	TP Rate	FP Rate	Precision	Recall	F-Measure
0	0.793	0.258	0.729	0.793	0.76
1	0.742	0.207	0.804	0.742	0.772
Weighted Avg.	0.766	0.231	0.769	0.766	0.766



4. Possible improvements

The results and vizualization of ADTree, the decision tree giving the best accuracy for our advanced system, show us that relying on interaction words for the classification improves performance. We also note that the correct and wrong classification are more balanced across the two classes, while they were not symmetrical at all in the baseline system: a lot of sentences were previously classified as describing an interaction (1) whether they were (true positive) or not (false positive).

We believe further improvement could have been reached with the Stanford Parser [2], taking the dependency or constituency parsing into account, but despite our attempts we did not manage to make it run in Python. We only had access to the GUI provided as a demo in the package, or to the online version of the parser: http://nlp.stanford.edu:8080/parser/index.jsp, but both only accepted very limited amount of text, which did not suit our data.

References

[1] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witten (2009); The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1.

[2] Marie-Catherine de Marneffe, Bill MacCartney and Christopher D. Manning. 2006. Generating Typed Dependency Parses from Phrase Structure Parses. In *LREC 2006*.