Chapter 5 – Results and Statistic Analysis

5.1
Enhancements and optimization

5.1.1
Classification and regression trees - selection of the adequate pruning technique

To ensure that the tree will perform as well as in the training sample, a validation procedure was applied. The most preferred type of validation is testing with a sample taken from the original dataset, especially when this dataset is large enough. The sample size can be approximately one-third to one-half of the learning dataset (Brieman et al. 1984). Since the data available is not large enough, the validation will be tested in different techniques. This procedure verifies that the tree will perform equally well with other datasets.

Several pruning techniques, explained on section 4.3.3.3 have been tested in order to achieve a better generalization and therefore better results on independent test (out-of-sample) data.

- Pruning a percentage of the total levels:

Table 9: Example of prediction accuracy per pruning percentage. Prediction accuracy ranges from 0 to 1 and represents “correct predictions” divided by “total number of predictions”.

Percentage of pruned levels	Prediction accuracy
0% (unpruned tree)	0.4721
10%	0.5127
20%	0.5228
30%	0.5482
40%	0.5533
50%	**0.6091**
60%	0.5939
70%	0.5228
80%	0.4721
90%	0.4518
100%	0.4518

- Pruning with a ‘Best level’ option with cross-validation from the training (in-sample) data itself:

Table 10: Example of prediction accuracy with ‘best level’ pruning. Prediction accuracy ranges from 0 to 1 and represents “correct predictions” divided by “total number of predictions”.

Number of pruned levels	Prediction accuracy
„Best level“ (a bit unstable)	0.5330

- Pruning with “validation on independent data”:

Table 11: Example of prediction accuracy with pruning according to validation data. Prediction accuracy ranges from 0 to 1 and represents “correct predictions” divided by “total number of predictions”.

Number of pruned levels	Prediction accuracy
„Validation on independent data“ option used	0.4920

- Pruning in order to leave the tree with one specific number of levels:

Table 12: Example of prediction accuracy with pruning using a fixed number of levels. Prediction accuracy ranges from 0 to 1 and represents “correct predictions” divided by “total number of predictions”.

Number of pruned levels	Prediction accuracy
Total no. of levels-1	0.4518
Total no. of levels -2	0.3858
Total no. of levels -3	0.5076
Total no. of levels -4	0.5482
Total no. of levels -5	**0.6041**
Total no. of levels -6	0.5888
Total no. of levels -7	0.5635
Total no. of levels -8	0.5381
Total no. of levels -9	0.5381
Total no. of levels -10	0.5279

The technique of pruning a percentage of the tree (50-60%) was the technique chosen to be used on all the experiments, since the results were the best and it more sense than the other methods. There is a correlation between the number of input variables used and the percentage of levels to be pruned.

The validation will be done on subsets of the original training set (V-fold cross-validation). This type of cross-validation is adequate for this case because the learning sample is too small to have the test sample taken from it. A specified

V value for V-fold cross-validation determines the number of random subsamples, as equal in size as possible, that are formed from the learning sample. The classification tree of the specified size is computed V times, each time leaving out one of the subsamples from the computations, and using that subsample as a test sample for cross-validation, so that each subsample is used V - 1 times in the learning sample and just once as the test sample. This way one can as well obtain the ideal pruning level.

By using the simulation with the ‘splitmin’ option instead of ‘catidx’ whenever setting up the tree, the regression tree is made only with > and < signs on the tree’s nodes (see figure 3), which makes the tree more comprehensible at sight. Using the ‘catidx’ option, whenever setting up the tree ends up building a tree with one node being decided with not only by > and < signs but by ‘part of {a,b,d,f,k,o}’ for instance which is not that comprehensible. The ‘splitmin’ option showed an average 20% better result than the ‘catidx’ option in several tests, so only the option ‘splitmin’ was used on all the tests.

Example: Input: #turns, CA:#IA, PA:CO, WPST, IC, weighted consecutive CA:IA

Target: Mean Questions B (average of all questions from the questionnaire)

Table 13: Comparison from prediction accuracy with the ‘catidx’ and ‘splitmin’ options. Prediction accuracy ranges from 0 to 1 and represents “correct predictions” divided by “total number of predictions”

Type of pruning method	Prediction accuracy
‘catidx’	0,457
‘splitmin’	0,553

Instead of finding out a number from 0 to 6 on the end of the tree, a class is found: “bad”, “average” or “good”. It doesn’t sound like a big change, but the whole tree structure changes and the number of levels on the tree is reduced to approximately one third of the size of what a regression tree would be.

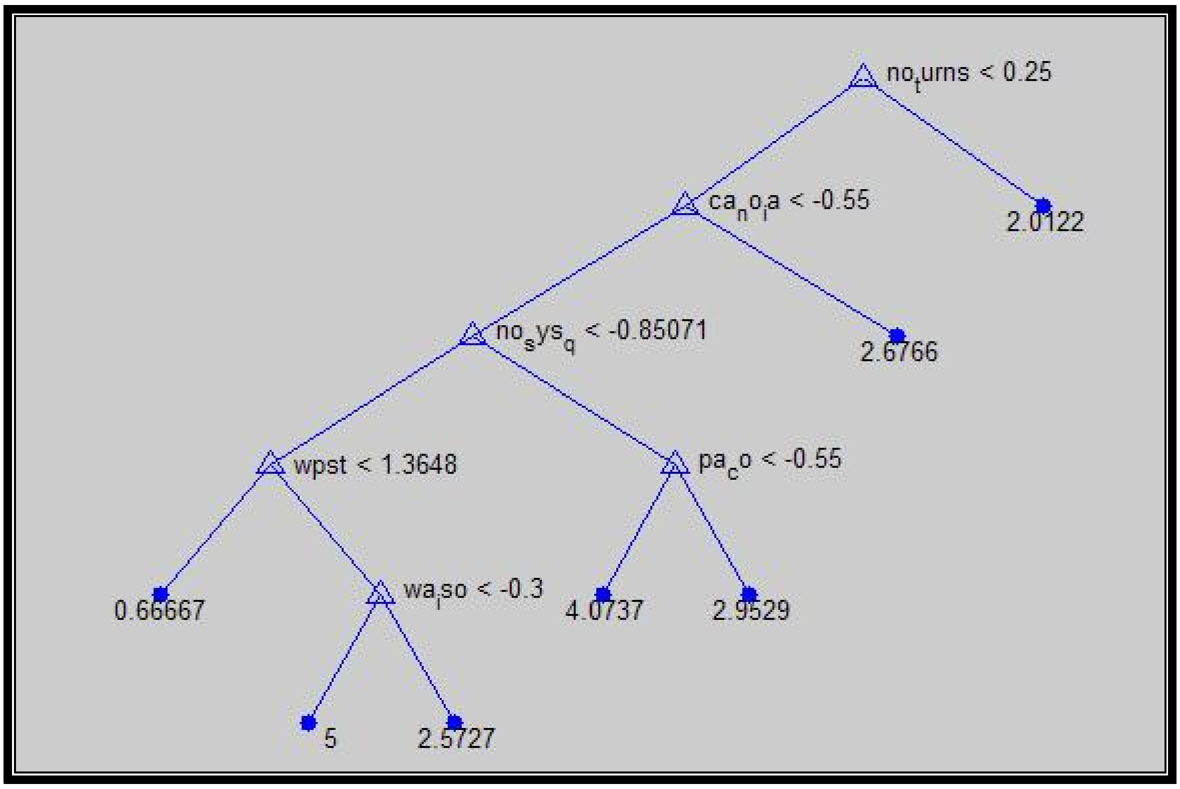

Figure 3: Example of pruned regression tree using 'splitmin'

The problem sometimes with the reduced number of nodes is that all expected values are pretty similar, like 2.01, 2.68, 2.95 which in theory gives a better R^2 but it is like a mean value from all Question B0 values: It has a good R^2 whenever you compare to the Question B0 target but it doesn´t mean it is a good predictor, since it is almost a mean value of all the user evaluations. This must all be taken under consideration when choosing the adequate size of the classification or regression tree.

5.1.2
Neural networks - Experiment setup

The configuration '3-class evaluation', by using 3 'logsig' neurons as output from the neural network, gives the probabilities for each of the classes.

Table 14: Example of how output values look like. These are from 12 cases, using 3 'logsig' neurons on the 2nd layer. The output should be like the target values, meaning that the sum of the three columns should be 1 and each column should show a probability between 0 and 1 that the class represented by the column has of being the target value.

Output values from the experiment				Target Values (1s or 0s only)		
0,9972	1.1487e-020	8.0546e-051		1	0	0
0,00368	0,49795	8.2996e-041		0	1	0
8,7E-06	0,98972	1.4517e-052		0	1	0
1	2.3253e-025	3.3449e-037		1	0	0
0,41445	0,2478	2.8581e-058		0	1	0
2,7E-06	1	2.4741e-026		0	1	0
0,99293	0,01462	6.7628e-040		1	0	0
0,41445	0,2478	2.8581e-058		0	1	0
0,49915	0,49904	8.5404e-056		1	0	0
3,7E-07	0,98888	1.6666e-043		0	1	0
0,21	0,80982	7.2497e-058		0	1	0
4,9E-06	0,00154	2.1516e-042		0	0	1

The direct output therefore needed to be normalized into three probabilities where the sum would be one (table 15).

Table 15: Example from output values from 7 cases using 3 'logsig' neurons on the 2nd layer, but normalizing the probabilities so that the sum of the probabilities would be 1.

Output		
0,009383	0,62313	0,36708
0,24237	0,33142	0,42631
0,3524	0,17033	0,47818
0,28963	0,18949	0,52197
0,46188	0,44509	0,091301
0,37989	0,60989	0,011029
0,11529	0,15258	0,73104

One other issue with the experiment setup when dealing with neural networks was the choice of the training function.

Some training functions left a lot of unstable results. Using the exact same configuration, and just running the experiment again, made different results appear (see table 16 below):

Table 16: Example on test (out-of-sample) data - Column 1 is the target data and columns 2-5 are the linear outputs from the neural network using the same configuration.

Target Value	Output from 1 'purelin' neuron			
3	2	2	2	2
4	15,17	13,79	19,94	5
2,5	-2,5	2,15	12,54	8,57
1,5	6,35	5,98	5,17	-6,1
2	-8,22	-11,34	-26,79	5,67
1,2	1,05	2,25	3,53	1,81
4	-0,82	7,91	3,26	6,4
3	2,98	2,98	2,98	2,98
0,5	2,16	25,13	4,3	-0,06
0,7	3,32	11,44	-5,68	40,48
2,1	2,98	2,98	2,98	2,98
0,3	-14,19	-6,8	-9,97	4,93
1,1	0,61	14,1	6,24	-1,3
2,5	2	2	2	2
2	-4,28	1,89	-21,11	-0,26
2	2,63	2,63	2,63	2,63
3	4,21	3,93	2,68	6,83
2,5	4,99	-12,66	19,68	37,7
3,5	2,63	2,63	2,63	2,62
3,5	-6,95	19,49	-1,85	-11,64
1,1	4,29	-1,29	6,5	31,02
4,9	2,98	2,98	2,98	2,98
2,1	-18,48	-3,75	-1,85	-38,25
2,1	2,98	2,98	2,98	2,98
1,9	-6,94	2,34	-22,93	1,59
4	4,19	4,19	4,19	4,18
3,1	2,85	-2	4,16	24,52

For some dialogues, the result on the table 16 above is the same from column 2 to 5, like it should be normally, since no configuration was changed. For other dialogues, the numbers are even outside the normal range (0 to 6) or even negative. The explanation is that the training function used on the neural network was the “traingdm” (Gradient descent with momentum backpropagation, a network training function that updates weight and bias values according to gradient descent with momentum), which in comparison to other functions was not very stable. The “ideal” training function, after several tries was the ‘trainbfg’: a network training function that updates weight and bias values according to the BFGS quasi-Newton method. ‘Trainbr’ is a good function for the linear output as well, it is a network training function that updates the weight and bias values according to Levenberg-Marquardt optimization. It minimizes a combination of squared errors and weights and, then determines the correct combination so as to produce a network which generalizes well. The process is called Bayesian regularization.

Even after the ideal function was found, there were still some issues of instability on the linear outputs from the neural network. This is due to different initial values on the simulation, which change every time the simulation is run. The neural network should not be so dependent on initial values, that the outputs would vary so much (like from 2,34 to 5,4).

The explanation for this is that some kinds of dialogues are well “covered” by the neural network, while others have to be “extrapolated”. The “covered” cases do not depend so much on the initial values, but the “extrapolated” ones are.

Since the training runs are sensitive to the initial conditions for the weights, the neural network will be trained several times using different weight initializations. This leads to a set of different networks, where the best network can be kept or the set can be used as a *committee* (Perrone and Cooper, 1993; Perrone, 1994) of networks and the mean value of the output is used. This method is more reliable than using a single network.

The best solution for this would be to have a bigger amount of training (in-sample) data, so that more cases can be “covered” by the neural network.

A weighted sum of the probabilities, in order to calculate which class was predicted, was created. Results are worst than by choosing the class by the highest probability. (‘Higher p’ is in general 30% better than ‘weighted p sum’). After these results, a different approach to the error calculation was tried. A weighted error, computing the error 'bad'→'good' or 'good' →'bad' to be worse than the 'average'→'good' error was used.

The results, depending on the amount of neurons are unstable as well. The complexity of neural networks can be varied by altering the number of adaptive variables in the network. This procedure is called *structural stabilization.* Its objective is to find an equilibrium between bias and variance, that is the objective of generalization basically as well. It can be done by changing the number of adaptive variables in the network. A way to implement this in practice is to do a comparison study between different models with each having a different amount of hidden units. For instance, a small network can be started and be added units during the learning/training process, with the objective of finding an optimal network structure. The table 17 shows this procedure.

Choice of the ideal number of neurons on the hidden layer:

For any size of data set, there is an optimal balance between bias and variance which produces the smallest average generalization error. To improve the performance of the network the bias should be reduced while reducing at the same time the variance. One way to achieve this is by using a bigger amount of data, but since for now this is not possible, several different kinds of neural networks were used until the appropriate one was found. For feed-forward networks, White (1990) has shown that the complexity of a two-layer network must grow in relation to the size of the data set in order to be consistent. So this means that if the data set is not very big, an "uncomplex" network would have better results than a "complex" one. The "early-stopping" technique described on chapter 4.3.2.1.2 is as well a way of controlling the effective complexity of a network.

An experiment was made to check how many neurons should be used on the hidden layer. The following table shows the results:

Table 17: Statistics from the dependency of the number of neurons for a neural network using #turns, ca:#ia, ts_ord and uct as input variables and Mean B as target value.

Number of neurons	Pearson's Correlation	R^2	$\overline{R^2}$
1	33,3%	0,039	0,018
3	39,2%	0,145	0,126
4	40,2%	0,157	0,138
7	40,7%	0,154	0,135
10	41,4%	0,155	0,137
15	**43,8%**	**0,174**	**0,156**
25	42,6%	0,157	0,138
35	42,9%	0,148	0,129
40	43,8%	0,157	0,139
45	41,9%	0,128	0,108
50	42%	0,128	0,108
75	39,5%	0,059	0,038

In general the best results come with 15-20 neurons, but sometimes a rule of thumb "number of observations divided by number of input variables" has the better results (Bishop, 1995). For the majority of the experiments presented on this work, this rule was used.

5.2
Overall Results

All the experiments below were made using the same principles behind the PARADISE model, but this time calculating the statistics on independent test (out-of-sample) data as well. This chapter compares the four models among themselves in several different aspects.

Experiment 1:

Stepwise inclusion of input variables to predict Question B0 (Overall Quality) using linear regression – BoRIS System - 9 input variables set (#turns, CA:#IA, PA:CO, TS_ord, WA_iso, WPST, UCT, IC, #Sys.Questions), one by one added to the input.

Table 18: Results from experiment 1

Input variables	Training (in-sample) correlation	Training (in-sample) R^2	Training (in-sample) $\overline{R^2}$	Test (out-of-sample) Correlation	Test (out-of-sample) R^2	Test (out-of-sample) $\overline{R^2}$
#turns	34,9%	0,121	0,117	32,3%	0,102	0,097
#turns, CA:#IA	35,8%	0,128	0,119	31,9%	0,099	0,089
#turns, CA:#IA, PA:CO	36,1%	0,130	0,116	31,0%	0,092	0,077
#turns, CA:#IA, PA:CO, ts_ord	38,3%	0,146	0,129	32,5%	0,101	0,082
#turns, CA:#IA, PA:CO, ts_ord, wa_iso	38,3%	0,146	0,129	27,2%	0,056	0,037
#turns, CA:#IA, PA:CO, ts_ord, wa_iso, WPST	38,3%	0,147	0,129	25,6%	0,040	0,020
#turns, CA:#IA, PA:CO, ts_ord, wa_iso, WPST, UCT	39,7%	0,157	0,125	25,9%	0,027	-0,011
#turns, CA:#IA, PA:CO, ts_ord, wa_iso, WPST, UCT, IC	39,8%	0,158	0,121	23,8%	0,003	-0,041
#turns, CA:#IA, PA:CO, ts_ord, wa_iso, WPST, UCT, IC, #sys.questions	40,1%	0,160	0,118	22,6%	-0,027	-0,079

*R^2 is a measure for training (in-sample) data originally, but the same formula was used on test (out-of-sample) data just for comparison

The PARADISE model showed that as more input variables were used, better the accuracy would be, but this happens only for training data. Whenever it comes to test (out-of-sample) accuracy, as the table above shows, this relation is different. The results show that a high training (in-sample) accuracy does not necessarily mean that the test (out-of-sample) accuracy will get better, or stay the same, as it happens with the training (in-sample) data.

The same logic from this experiment applies for all the other three approaches (regression trees, classification trees and neural networks).

Experiment 2:

This experiment has as purpose to show the best results that each approach achieved on experiments using input variables from the input 9 from experiment 1 or the error classification variables (only for the INSPIRE system config.1).

- Best results using linear regression on test (out-of-sample) data for each system:

Table 19: Results from experiment 2 – linear regression

System	Input variables	Target value	Pearson´s correlation	$\overline{R^2}$
BoRIS	#turns, TS_ord	Mean B Questions	39,2%	0,142
INSPIRE config. 1	WPST and UCT	Question B0 (overall quality)	41,3%	0,14
INSPIRE config. 2	#turns	Question B0 (overall quality)	50,1%	0,232

- Best results using regression trees on test (out-of-sample) data for each system:

Table 20: Results from experiment 2 – regression trees

System	Input variables	Target value	Pearson´s correlation	R^2	$\overline{R^2}$
BoRIS	Task success, WPST, UCT	Mean B Questions	46,1%	0,197	0,179
INSPIRE config. 1	PA:CO, IC, #sys. questions	Question B0 (overall quality)	46,6%	0,080	0,04
INSPIRE config. 2	#turns, PA:CO, wa_iso, UCT	Question B0 (overall quality)	60,3%	0,347	0,307

- Best results using classification trees on test (out-of-sample) data for each system:

Table 21: Results from experiment 2 – classification trees

System	Input variables	Target value	Accuracy on predicting the 3-class evaluation
BoRIS	(#turns, CA:#IA, PA:CO,WPST, IC, weighted consecutive CA:IA	Mean B Questions	60,9% (predicting “bad”, “average” or “good”)
INSPIRE config. 1	‘space’, wa_iso’, ‘verb’.	“Use again”	50,7% (predicting ‘no’, ‘undecided’ and ‘yes’)
INSPIRE config. 2	#turns, CA:#IA, WPST, UCT	Question B0 (overall quality)	59,4% (predicting “bad”, “average” or “good”)

- Best results using neural networks on test (out-of-sample) data for each system:

Table 22a: Results (Correlation and $\overline{R^2}$) from experiment 2 – neural networks

System	Input variables	Target value	Pearson´s correlation	$\overline{R^2}$
BoRIS	#turns, CA:#IA, TS_ORD, UCT	Mean B Questions	43,8%	0,156
INSPIRE config. 1	‘space’, ‘repetition’	overall quality	39,1%	0,119
INSPIRE config. 2	‘#turns’, ‘CA:#IA’, ‘wa_iso’	overall quality	59%	0,293

Table 22b: Results (accuracy) from experiment 2 – neural networks

System	Input variables	Target value	Accuracy
INSPIRE config. 1	‘space’, wa_iso’, ‘verb’.	“Use again”	57,3%

These results show that for the BoRIS system, the regression trees achieved the best results on linear output and the classification trees on 3-class evaluation. For the INSPIRE system config. 1 (free w.o.z.), the linear regression had the best adj. r-square and was the best method. For the INSPIRE system config. 2 (formal w.o.z.) the best result was achieved either with the neural networks approach or the regression tree approach. For the majority of the trials, the regression tree method and the neural network approach had better results than linear regression, i.e. better results than the PARADISE model.

<u>Experiment 3:</u>

This experiment has the purpose of comparing the four methods with a large set of input variables (Input9 - #turns, CA:#IA, PA:CO, TS_ord, WA_iso, WPST, UCT, IC, #Sys.Questions) on the BoRIS system. The tables below show the results.

Table 23a: Results (1-class evaluation) from experiment 3 – all methods, input 9 – Question B0

	Linear Regression	Regression Trees	Neural networks
Correlation on training (in-sample) data	40,1%	96,2%	85,5%
R^2 on training (in-sample) data	0,160	0,926	0,729
$\overline{R^2}$ on training (in-sample) data	0,118	0,923	0,715
Correlation on test (out-of-sample) data	22,6%	22,2%	28%
R^2* on test (out-of-sample) data	-0,027	-0,046	0,066
$\overline{R^2}$ * on test (out-of-sample) data	-0,079	-0,097	0,019

*R^2 is a measure for training (in-sample) data originally, but the same formula was used on test (out-of-sample) data just for comparison

Table 23b: Results (3-class evaluation) from experiment 3 – all methods, input 9 – Question B0

	Classification Trees	Neural networks
Training (in-sample) Accuracy on 3-class evaluation	95,9%	94,1%
Test (out-of-sample) Accuracy on 3-class evaluation	52,7%	42,7%

*R^2 is a measure for training (in-sample) data originally, but the same formula was used on test (out-of-sample) data just for comparison

Table 24a: Results (1-class evaluation) from experiment 3 – all methods, input 9 – Mean B Questions

	Linear Regression	Regression Trees	Neural networks
Correlation on training (in-sample) data	45,9%	96,6%	88,6%
R^2 on training (in-sample) data	0,211	0,934	0,784
$\overline{R^2}$ on training (in-sample) data	0,171	0,931	0,773
Correlation on test (out-of-sample) data	31,1%	33,8%	39,3%
R^2* on test (out-of-sample) data	0,034	-0,046	0,129
$\overline{R^2}$ * on test (out-of-sample) data	-0,014	-0,094	0,085

*R^2 is a measure for training (in-sample) data originally, but the same formula was used on test (out-of-sample) data just for comparison

Table 24b: Results (3-class evaluation) from experiment 3 – all methods, input 9 – Mean B Questions

	Classification Trees	Neural networks
Training Accuracy on 3-class evaluation	96,4%	95%
Test Accuracy on 3-class evaluation	48,2% (no pruning – best results)	42,7%

*R^2 is a measure for training (in-sample) data originally, but the same formula was used on test (out-of-sample) data just for comparison

The high correlation on training (in-sample) data from the classification and regression trees is due to the fact that the trees are totally unpruned, so therefore they have a big amount of nodes, big enough to comprehend all the cases from the training (in-sample) data. The regression tree in this case is overfitted for the training (in-sample) data and would have a terrible result whenever analysing and predicting new data. The neural networks approach has excellent results on training (in-sample) data as well for the same reason, the network becomes overfitted for the training (in-sample) data. These training (in-sample) data results are all superior to the ones from the PARADISE model (linear regression).

On the test (out-of-sample) data the results are terrible. The least worst is the neural networks approach. The amount of input variables does not necessarily means that a good prediction will be possible. On test (out-of-sample) data, some of the data can be considered 'noise', since a smaller set of input variables would have a better result than the full one. This is valid for all the four approaches and can be better seen on experiment 6.

Experiment 4:

Using the table 25 of correlations and the table on chapter 7.2, this experiment has the purpose of comparing the four methods with different sets of input variables (from the error classification set) from the INSPIRE system.

Table 25: Correlations between the ‘error classification’ variables and the overall quality from INSPIRE config. 1

		Overall quality	P.uct	WA_iso	ts	repetition	space
overall quality	Pearson Correlation	1	-,356 (** Sig. ,004)	,275 (* Sig. ,028)	,152 (Sig. ,232)	-,271 (* Sig. ,031)	-,375 (** Sig. ,002)
P.uct	Pearson Correlation	-,356 (** Sig. ,004)	1	-,400 (** Sig. ,001)	-,010 (Sig. ,936)	,052 (Sig. ,675)	,508 (** Sig. ,000)
WA_iso	Pearson Correlation	,275 (* Sig. ,028)	-,400 (** Sig.,001)	1	-,015 (Sig. ,904)	-,337 (** Sig. ,005)	-,535 (** Sig. ,000)
ts	Pearson Correlation	,152 (Sig. ,232)	-,010 (Sig. ,936)	-,015 (Sig. ,904)	1	,014 (Sig. ,908)	-,199 (Sig. ,107)
repetition	Pearson Correlation	-,271 (* Sig. ,031)	,052 (* Sig. ,675)	-,337 (** Sig. ,005)	,014 (Sig. ,908)	1	,312 (* Sig. ,010)
space	Pearson Correlation	-,375 (** Sig. ,002)	,508 (** Sig. ,000)	-,535 (** Sig. ,000)	-,199 (Sig. ,107)	,312 (* Sig. ,010)	1

** Correlation is significant at the 0.01 level (2-tailed). * Correlation is significant at the 0.05 level (2-tailed).

The following results, using different sets of input variables, all from table 25, compare the four approaches statistically:

Table 26a: Correlations (1-class evaluation) between the input3b and 'use again' from INSPIRE config. 1 (see INSPIRE System questionnaire, chapter 8.2).

	Linear Regression	Regression Trees	Neural networks
Correlation on training (in-sample) data	47,5%	87,1%	85,1%
R^2 on training (in-sample) data	0,226	0,759	0,725
$\overline{R^2}$ on training (in-sample) data	0,19	0,747	0,717
Correlation on test (out-of-sample) data	20,7%	29,8%	29,7%
R^2* on test (out-of-sample) data	-0,065	-0,054	0,052
$\overline{R^2}$ * on test (out-of-sample) data	-0,115	-0.105	0,007

Table 26b: Correlations (3-class evaluation) between the input3b and 'use again' from INSPIRE config. 1 (see INSPIRE System questionnaire, chapter 8.2).

	Classification Trees	Neural networks
Accuracy on 3-class evaluation (training data)	85,7%	90,6%
Accuracy on 3-class evaluation (test data)	50,7%	57,3%

Table 27a: Correlations (1-class evaluation) between the input3 and 'overall quality' from INSPIRE config. 1

	Linear Regression	Regression Trees	Neural networks
Correlation on training (in-sample) data	42,7%	95%	92%
R^2 on training (in-sample) data	0,182	0,903	0,846
$\overline{R^2}$ on training (in-sample) data	0,144	0,898	0,832
Correlation on test (out-of-sample) data	20%	21,7%	27,9%
R^2* on test (out-of-sample) data	-0,044	-0,052	0,051
$\overline{R^2}$ * on test (out-of-sample) data	-0,093	-0,104	0,001

Table 27b: Correlations (3-class evaluation) between the input3 and 'overall quality' from INSPIRE config. 1

	Classification Trees	Neural networks
Accuracy on 3-class evaluation (training data)	95,5%	92,1%
Accuracy on 3-class evaluation (test data)	46,2%	40,9%

Table 28a: Correlations (1-class evaluation) between the input2 and 'overall quality' from INSPIRE config. 1

	Linear Regression	Regression Trees	Neural networks
Correlation on training (in-sample) data	38,4%	63,3%	83,3%
R^2 on training (in-sample) data	0,147	0,401	0,693
$\overline{R^2}$ on training (in-sample) data	0,121	0,383	0,679
Correlation on test (out-of-sample) data	26,5%	28,1%	39,1%
R^2* on test (out-of-sample) data	0,058	0,024	0,148
$\overline{R^2}$ * on test (out-of-sample) data	0,026	-0,007	0,119

Table 28b: Correlations (3-class evaluation) between the input2 and 'overall quality' from INSPIRE config. 1

	Classification Trees	Neural networks
Accuracy on 3-class evaluation (training data)	62,9%	77,9%
Accuracy on 3-class evaluation (test data)	23,8%	39,4%

Table 29a: Correlations (1-class evaluation) between the input4 (‘Unprogressive state’, ‘verb’, ‘Space’, ‘repetition’) and ‘overall quality’ from INSPIRE config. 1

	Linear Regression	Regression Trees	Neural networks
Correlation on training (in-sample) data	39,8%	83,5%	85,5%
R^2 on training (in-sample) data	0,159	0,698	0,731
$\overline{R^2}$ on training (in-sample) data	0,105	0,684	0,719
Correlation on test (out-of-sample) data	20,2%	17,8%	25,4%
R^2* on test (out-of-sample) data	-0,017	-0,100	0,055
$\overline{R^2}$ * on test (out-of-sample) data	-0,075	-0,154	-0,011

Table 29b: Correlations (3-class evaluation) between the input4 (‘Unprogressive state’, ‘verb’, ‘Space’, ‘repetition’) and ‘overall quality’ from INSPIRE config. 1

	Classification Trees	Neural networks
Accuracy on 3-class evaluation (training data)	98,5%	96,5%
Accuracy on 3-class evaluation (test data)	41,7%	37,7%

Table 30a: Correlations (1-class evaluation) between the input5 (‘%uct, ‘wa_iso’, ‘ts_ord’, ‘space’, ‘repetition’) and ‘overall quality’ from INSPIRE config. 1

	Linear Regression	Regression Trees	Neural networks
Correlation on training (in-sample) data	47,6%	97,2%	92,2%
R^2 on training (in-sample) data	0,227	0,946	0,850
$\overline{R^2}$ on training (in-sample) data	0,165	0,941	0,832
Correlation on test (out-of-sample) data	26,1%	18,4%	41,2%
R^2* on test (out-of-sample) data	0,001	-0,265	0,101
$\overline{R^2}$ * on test (out-of-sample) data	-0,078	-0,372	0,029

Table 30b: Correlations (3-class evaluation) between the input5 (‘%uct, ‘wa_iso’, ‘ts_ord’, ‘space’, ‘repetition’) and ‘overall quality’ from INSPIRE config. 1

	Classification Trees	Neural networks
Accuracy on 3-class evaluation (training data)	85,7%	90,5%
Accuracy on 3-class evaluation (test data)	26,8%	40,9%

Table 31a: Correlations (1-class evaluation) between the input8 (#turns, CA:#IA, PA:CO, WA_iso, WPST, UCT, IC, #Sys.Questions) and 'overall quality' from INSPIRE config. 1

	Linear Regression	Regression Trees	Neural networks
Correlation on training (in-sample) data	59%	100%	98,5%
R^2 on training (in-sample) data	0,349	1	0,97
$\overline{R^2}$ on training (in-sample) data	0,260	1	0,952
Correlation on test (out-of-sample) data	32,5%	10,6%	32,8%
R^2* on test (out-of-sample) data	0,028	-0,541	0,075
$\overline{R^2}$ * on test (out-of-sample) data	-0,103	-0,750	0,005

Table 31b: Correlations (3-class evaluation) between the input8 (#turns, CA:#IA, PA:CO, WA_iso, WPST, UCT, IC, #Sys.Questions) and 'overall quality' from INSPIRE config. 1

	Classification Trees	Neural networks
Accuracy on 3-class evaluation (training data)	100%	83,6%
Accuracy on 3-class evaluation (test data)	32,8%	39,3%

Again, the relatively high correlation on training (in-sample) data from the classification and regression trees is due to the fact that the trees are totally unpruned, so therefore have a big amount of nodes, big enough to comprehend all the cases from the training (in-sample) data. The regression tree in this case is over fitted for the training (in-sample) data and would have a terrible result whenever analysing and predicting new data.

On the test (out-of-sample) data, the neural networks obtain the best results (with two exceptions - classification trees on input 3 and 4) in comparison to the other methods. The results are still not satisfying in terms of the method really being able to be used to predict the target values, but in comparison to the PARADISE model, the neural networks were superior in all aspects for all inputs from this experiment.

5.3 Results per approach

5.3.1 Linear regression

Experiment 5:

This experiment had the objective of analysing the results of linear regression on a user level. As seen on the past experiment, the input 8 (#turns, CA:#IA, PA:CO, WA_iso, WPST, UCT, IC, #Sys.Questions) from the INSPIRE System will be used as input set and overall quality as target value:

Table 32: Correlations between the input8 and ‘overall quality’ from INSPIRE config. 1, linear regression

Statistics	Linear Regression
Correlation on training (in-sample) data	59%
R^2 on training (in-sample) data	0,349
$\overline{R^2}$ on training (in-sample) data	0,260
Correlation on test (out-of-sample) data	32,5%
R^2* on test (out-of-sample) data	0,028
$\overline{R^2}$ * on test (out-of-sample) data	-0,103

The results on independent test (out-of-sample) data are pretty disappointing, but checking on a user level, the following correlations per user are obtained:

Table 33: Correlation per user as he/she is left out as test (out-of-sample) data, Input 8, Inspire System 1

User no.	Pearson´s correlation
1	0,925865
2	0,981981
3	0,784264
4	-0,50854
5	0,421653
6	-0,73935
7	0,999331
8	-0,95599
9	0,220316
10	0,811826
11	0,883222

12	-0,06824
13	-0,02454
14	-0,04834
15	-0,02039
16	-0,66176
17	-0,62573
18	0,737805
19	0,294147
20	-1
21	0,996473
22	-0,95294
23	0,891364

This means that some users are well predicted by the model, while some others have a terrible correlation. This means that the model has to extrapolate the prediction, since this ‘type’ of user is not covered by the model. If a bigger training set of data was available, less predictions would have to be extrapolated, since several different ‘types’ of users would be covered by the model.

5.3.2 Regression Trees and Classification Trees

Experiment 6:

Classification and regression trees are very dependent on the number of input variables used whenever it comes to independent test (out-of-sample) data prediction.

The example below can illustrate this:

Table 34: Results from experiment 6 - classification trees

System	Input variables	Target value	Accuracy on predicting the 3-class evaluation
BoRIS	Set 1: <u>4 input variables</u> #turns, CA:#IA, PA:CO,WPST	Mean B Questions	60,4%
BoRIS	Set 2: <u>9 input variables</u> #turns, CA:#IA, PA:CO, WPST, task success, WA:iso, UCT, IC, #sys. Questions	Mean B Questions	48,2%

The second set, the bigger set of variables, includes the 4 variables from the first set and has a lower accuracy on predicting the 3-class evaluation. The other five variables are “noise” to the ones from the first set, and make the results worse. This can be explained as well on the next experiment.

<u>Experiment 7</u>:

In this experiment, it was tried to use three factors instead of a set of nine input variables (set 2 from the past experiment). The comparison, in this case, will be done between a classification tree which uses the original 9 input variables and a classification tree which uses the 3 factors (originated from the 9 variables, invariant between each other). This would show if such the method works better with a small quantity of input variables or not.

Table 35: Results from experiment 7 - classification trees

System	Input variables	Target value	Accuracy on predicting the 3-class evaluation
BoRIS	Set 2: 9 input variables #turns, CA:#IA, PA:CO, WPST, task success, WA:iso, UCT, IC, #sys. Questions	Mean Questions B	48,2%
BoRIS	3 factors (that represent the 9 input variables above)	Mean Questions B	55,8%

This means that classification trees work better and more effectively with a small quantity of input variables, so therefore it is more effective to do data reduction on the pre-processing part if there is a big quantity of input variables than using all the variables.

Experiment 8:

This experiment was made to show general characteristics from classification trees as prediction method for overall quality from an user.

Some classification trees with good results can be properly interpreted, like the one on figure 4:

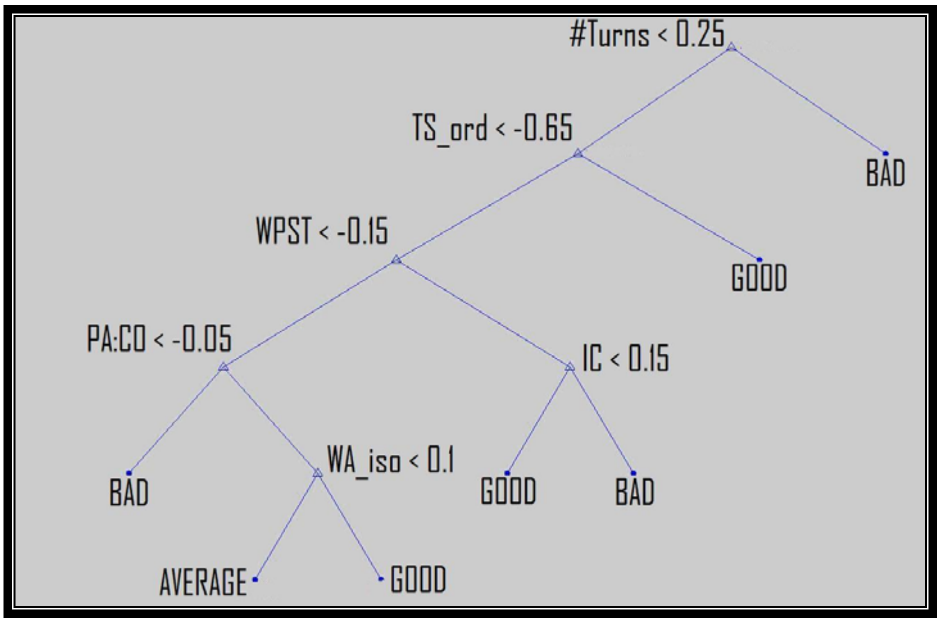

Figure 4: Example of classification tree: Target: Question B23, input variables: #turns, task success, WPST, IC, WA, PA:CO (values are normalized)– accuracy 52,2%

The classification tree above shows on the first node, that if the number of turns is too big, the user would find the dialogue "bad", on the second node that if the dialogue was not long and the task success was a specific value, the user would find the dialogue "good" for instance. This is not a rule that applies to all users, but what the method found from the majority of users.

Some classification trees, despite the good accuracy cannot be well explained whenever an analysis on the 'questions' from the nodes are made, like the one on figure 5:

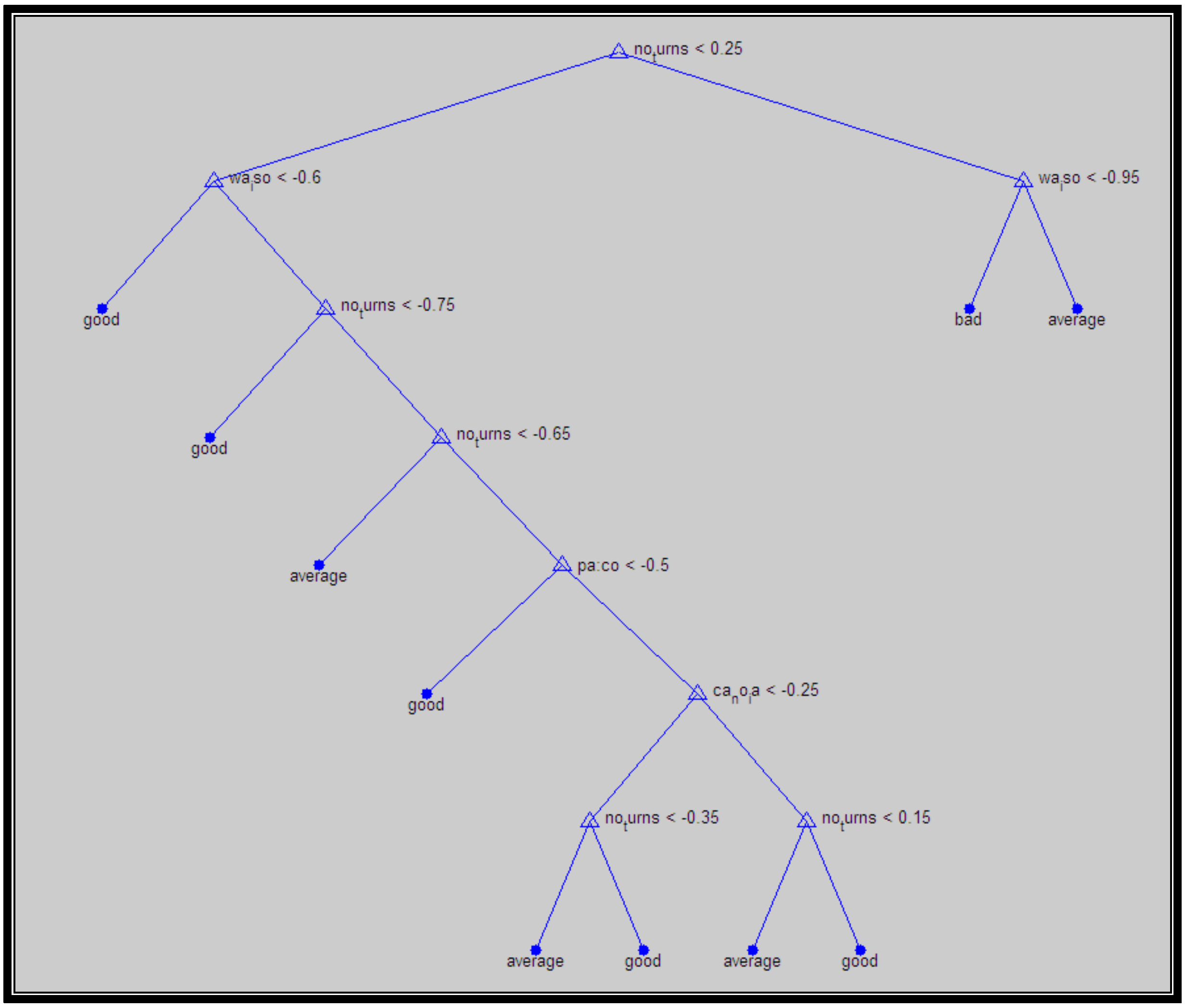

Figure 5: Classification Tree using no_turns, ca_no_ia, pa:co, wa_iso, uct., target Mean Questions B.

The tree from figure 5 has as test (out-of-sample) data result: 59,9 % accuracy to predict in which class 'mean B Questions' is: bad, average, good. The results are good in comparison to other approaches, but the rule in a couple of nodes are not logically explainable (the first node on the left says that if the word accuracy is small, the user would find the dialogue "good"). This can be explained as the tree being fitted only to specific data (overfitted).

The evaluation from 118 dialogues among 197 could be predicted with this classification tree. From the 79 wrong estimates, only 12 were completely wrong ('good' as 'bad' or 'bad' as 'good'). The histogram analysis shows that from 197 estimates, only 10 were estimated by the model as 'bad', 5 of which correct, 3 of them wrongly classified as the targets were 'average' and 2 of them wrongly classified as the targets were 'good'.

This means that few errors on a higher weight happened ('good' as 'bad' or 'bad' as 'good'), but the kind of approach on a 3-class evaluation is difficult to be made, since a small weighted error could be obtained when all dialogues are classified as "average" for instance.

Experiment 9:

This experiment has the goal of showing how classification trees are dependent on the pruning technique and which precautions should be made when using it. An error analysis is done in this case as well.

Using the correlation table for the inspire system (see chapter 7.2) as basis to choose which input variables would be used, the following results were achieved for the configuration
Input3b 'space', wa_iso', 'verb' / Target: 'Use again', from -2 to 2 (-2 and -1 are "no", 0 is "undecided", and 1 and 2 are "yes" - see INSPIRE System questionnaire, chapter 8.2). The classification tree is displayed on figure 6.

Table 36: Accuracy from classification tree on experiment 8, Leave-one-out technique, done on 23 users.

	Classification Trees
Accuracy on 3-class evaluation (training data)	85,7%
Accuracy on 3-class evaluation (test data)	50,7%

A better result on the test (out-of-sample) data could be achieved with a bigger pruning (58% accuracy), but the tree would be reduced to a single point 'yes', where the user would use the system again. This means that the tree always adapt itself to have the best accuracy possible, even if this means predicting every case from the ‘Use again’ question as ‘yes’

From the 67 dialogues, 50,7% could be properly predicted. This means 34 from the 67 dialogues could be correctly corrected. From the 33 errors, 10 (14,9%) were two-class errors (from ‘no’→’yes’ or ‘yes’→’no’). The other 23 errors were a one-class error. The tree on figure 6 is the unpruned one used for training (in-sample) data (85,7% correct predictions).

As conclusion from the experiment, the ideal size of the classification or regression tree should not be the one that has the best accuracy possible, but the one that obtains good results and comprehend all the input variables chosen.

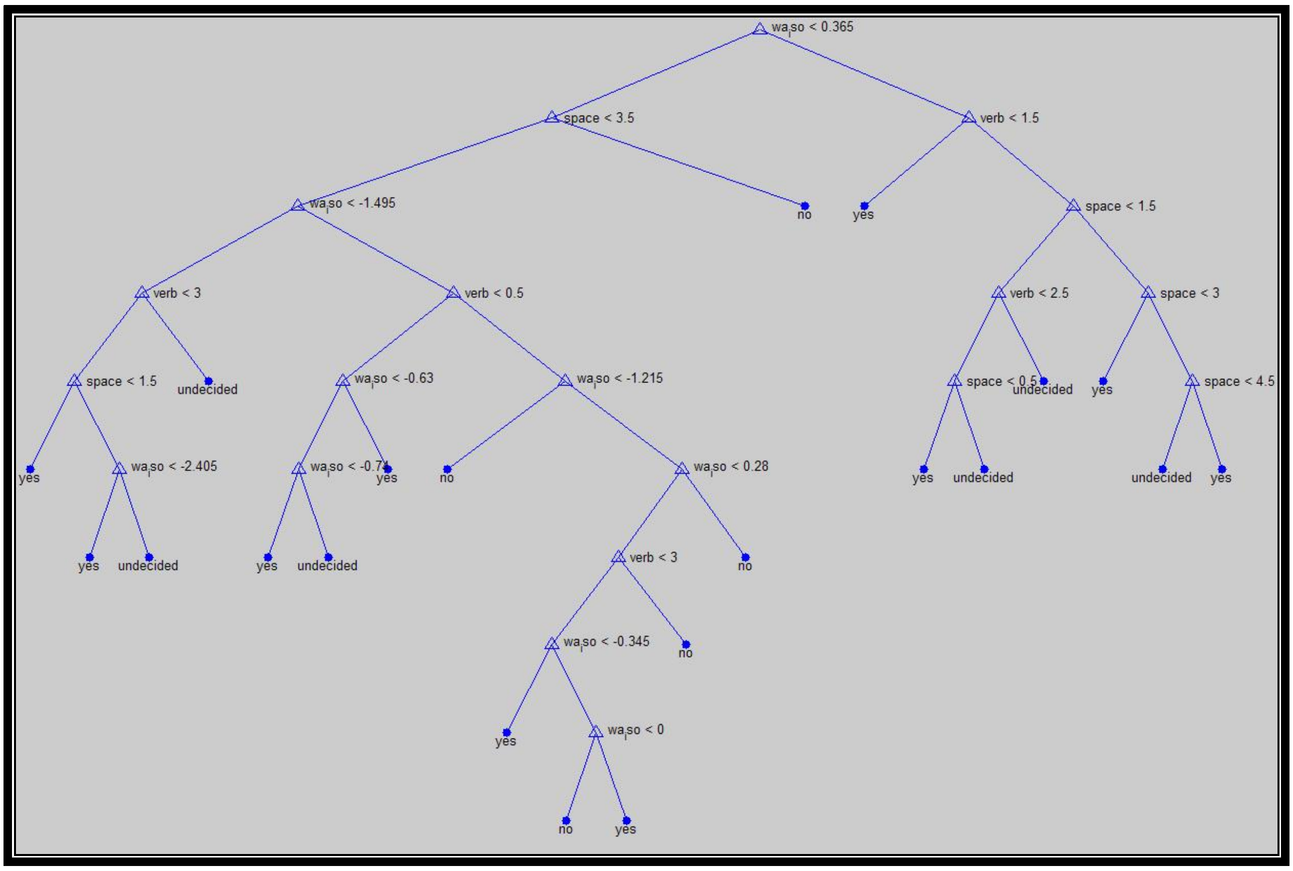

Figure 6: Classification Tree using Input3b ‘space’, wa_iso’, ‘verb’. / Target: ‘Use again’.

5.3.3
Neural Networks

<u>Experiment 10</u>:

This experiment has the goal to do a “user per user” analysis on the neural networks approach.

Configuration 3: BoRIS System, 4 input variables (#turns, weighted CA:IA, ts_ord, UCT), target value is ‘Mean B Questions’. Training function is ‘trainbr’. 45 ‘tansig’ neurons are on the hidden layer and 1 ‘purelin’ neuron is used as output. The performance ratio of the network is set to 0.5, which gives equal weight to the mean square errors and the mean square weights. The Pearson´s correlation for the experiment with configuration 3 is 41,6% , adj. R^2 is 0,14.

Analysing the correlation on a user level we obtain the following table:

Table 37: Pearson´s correlation per user

User number	Pearson´s correlation
1	0,438921
2	0,761254
3	0,169109
4	0,047877
5	0,878463
6	0,87693
7	0,466302
8	0,890653
9	0,956683
10	0,79376
11	0,529455
12	0,902925
13	0,568687
14	-0,29333
15	-0,09746

16	0,686732
17	0,632102
18	0,373147
19	-0,66569
20	0,844149
21	0,223562
22	-0,02458
23	0,000786
24	-0,4127
25	0,24625
26	0,95205
27	0,957146
28	-0,21072
29	0,968122
30	0,499849
31	-0,37068
32	0,075574
33	0,770305
34	0,965653
35	0,927073
36	0,927904
37	-0,39198
38	0,394053

Users 5, 6, 8, 9, 12, 26, 27, 29, 34, 35 and 36 (11 from a total of 38 users – 29% of them) are pretty well covered by the neural network. The correlation is higher than 87,6%.

Users 2, 10, 16, 17, 20 and 33 (6 from 38 users – 16%) are well covered and would be a good approximation. The correlation is between 63 and 84%.+

The other users (21 from 38 users – 55%) are extrapolated and have bad/poor correlations. The correlations are from -41% to 56%.

Analysing these results user by user this difference can be better understood. For instance, user 9 (correlation 95,6% - R^2 0,86) has an excellent prediction considering he is not a part of the training data:

Table 38: Pearson´s correlation for user 9

User number	Target Value	Estimated value
9	3,87	3,799961
9	3,69	3,552077
9	3,78	3,56949
9	2,7	2,896718
9	3,45	3,587818

Users 27 (95,7% correlation R^2 0,48) and 34 (96,5% R^2 0,46) have some good prediction capacity as well.

Table 39: Pearson´s correlation for user 27

User number	Target Value	Estimated value
27	4,63	3,551005
27	3,11	3,202445
27	2,69	2,83792
27	2,45	2,61291
27	4,55	3,551005

Table 40: Pearson´s correlation for user 34

User number	Target Value	Estimated value
34	2,83	2,401377
34	3,74	3,17282
34	4,02	3,369539
34	3,52	3,381826
34	2,17	1,656416

User 20 (correlation 84,4% but R^2 -0,01) still has a good prediction capacity, considering it is not a part of the training data. The target value goes down whenever the target goes down as well.

Table 41: Pearson´s correlation for user 20

User number	Target Value	Estimated value
20	3,3	3,4848
20	3,6	3,014572
20	3,32	3,361447
20	4,18	3,425773
20	2,26	1,246479

On the other hand, user 24 (-41,2% correlation R² -0,68) cannot predict in a good way results over 4 or lower results. It is probably badly extrapolated as well as 55% of the users.

Table 42: Pearson´s correlation for user 34

User number	Target Value	Estimated value
24	2,55	3,753535
24	4,19	3,398728
24	1,48	3,712986
24	3,61	3,35123
24	2,4	3,139745

Using the same configuration above, a scatter plot (see figure 7) from the simulation is done to verify if the model is good or not on the prediction.

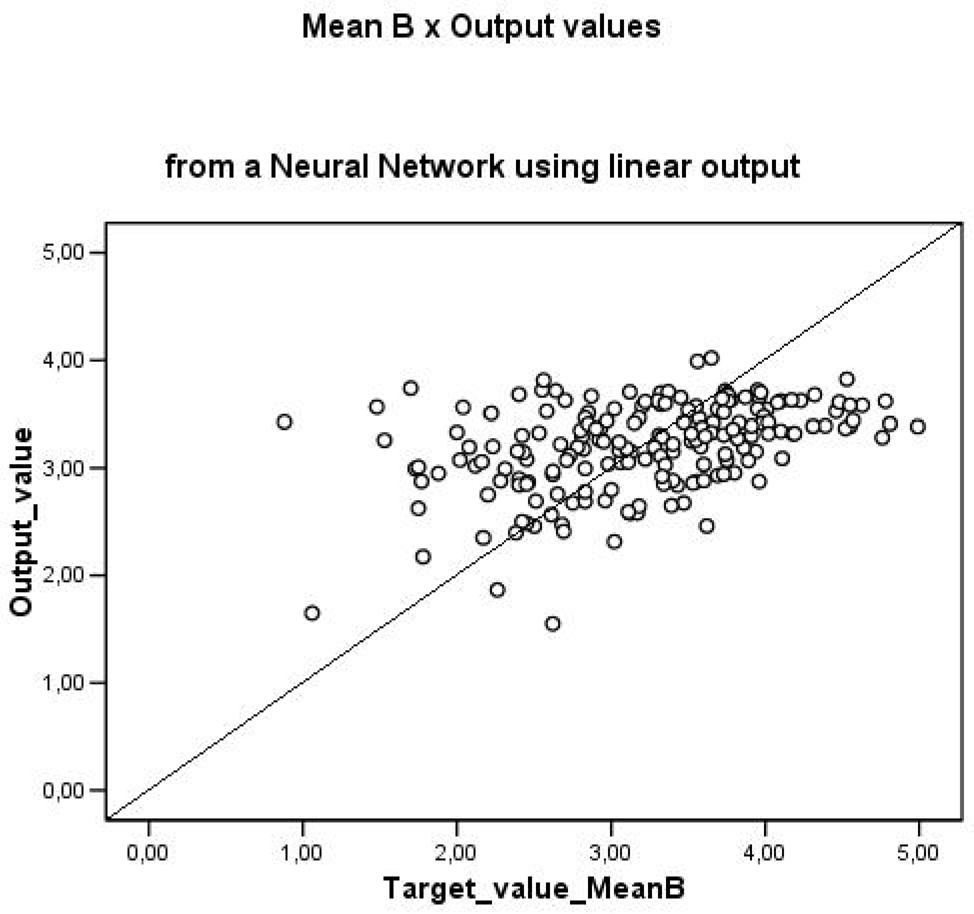

Figure 7: Scatter plot from simulation using 4 input variables (#turns, weighted CA:IA, ts_ord, UCT), target value is 'Mean B Questions'. Training function is 'trainbr'. 45 'tansig' neurons on the hidden layer and 1 'purelin' neuron as output. Validation for the "early-stop" technique is done with two independent users. The line crossing the graphic is the 'ideal' line.

A histogram analysis should be done as well to check if the distributions from the results and from the target values are similar or not.

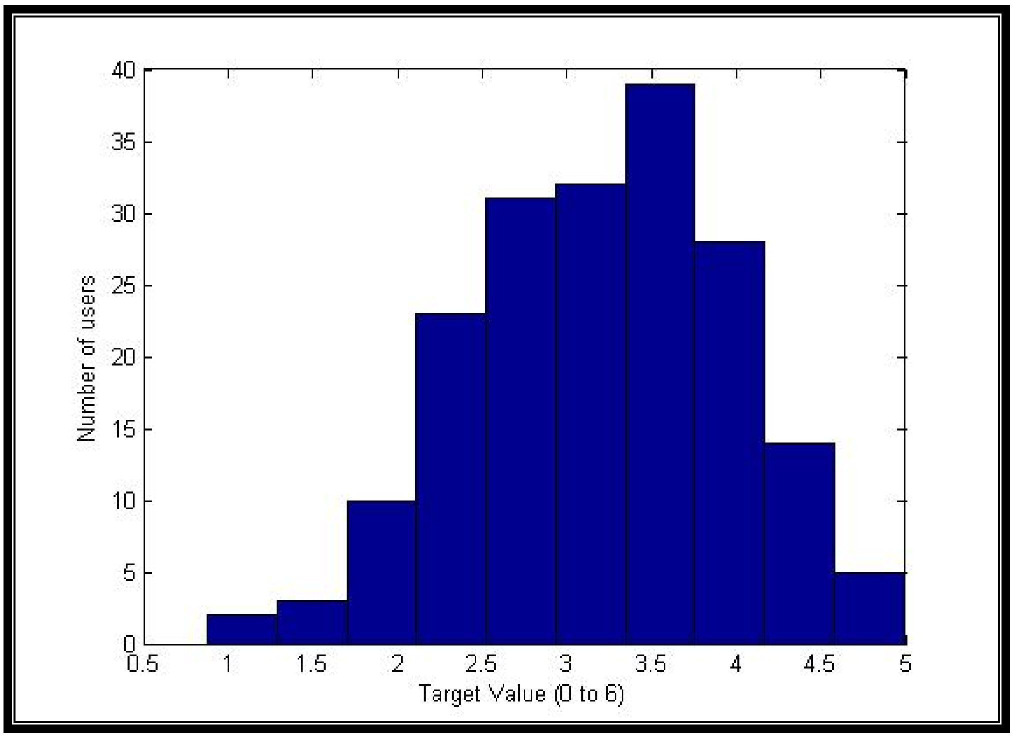

Figure 8: Histogram from the target values (0 to 6 – Mean B)

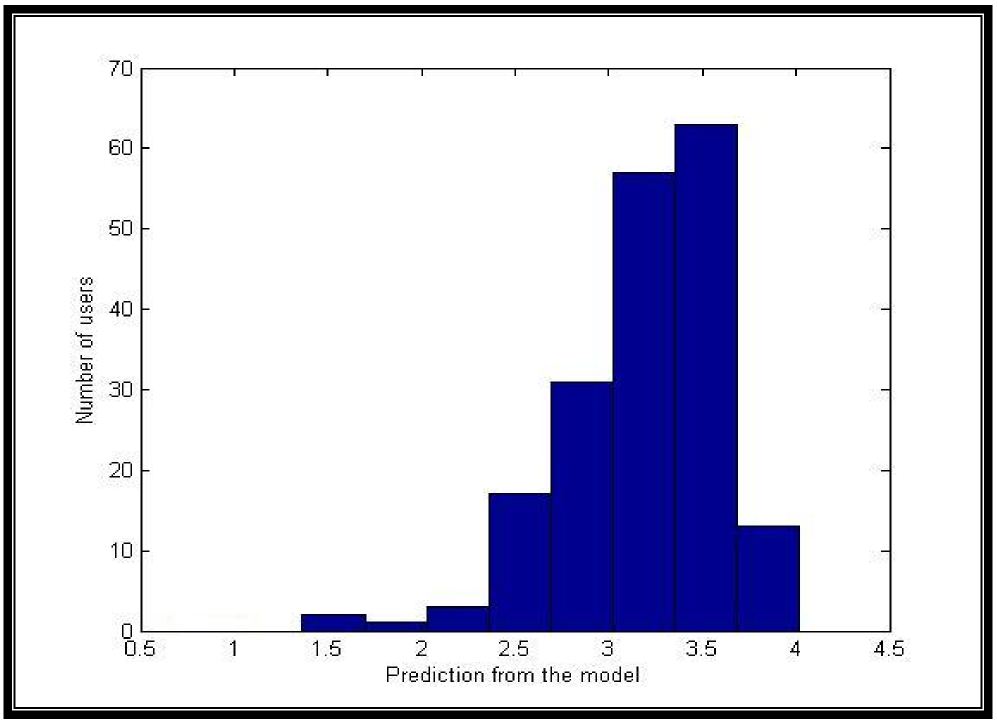

Figure 9: Histogram from the prediction values (0 to 6) made by the neural network done with configuration 3.

The histograms should have a similar distribution, but unfortunately they don´t. The prediction values are very concentrated on the 2,5 – 3,7 area, with very little predictions on the 0 – 2 and 3.7 – 6 area. This needs to observed whenever predictions on the NNs are done.

As seen before, the Pearson´s correlation for the experiment is 41,6% , adj. R-Square is 0,14. The average error is 0,571 on a scale from 0 to 6, which is a good result in principle, but after analysing the scatter-plot and the histogram, it can be seen that these values do not mean very much since the average error would be 0,63 if all the 197 predictions were the average value (3,2) of 'Mean B Questions' for instance. The prediction values have a range from 1.75 to 3.88, while the target values go from 0.88 to 4.99.

Experiment 11:

The experiment below has the purpose of showing how the amount of neurons in the hidden layer correlates to the accuracy on the 3-class evaluation for the INSPIRE system. It uses Input3 'space', wa_iso', 'verb', Target value: "Use again" in 3-class evaluation – 'no', 'undecided', 'yes'; Validation on 2 users, test on 1, 21 times. Training function: Trainbr. 3 'logsig' neurons on the output layer.

Table 43: Accuracy in relation to the amount of neurons in the hidden layer

Number of neurons on the hidden layer.	Accuracy (highest p)	Accuracy (weighted p)
1	54,1%	45,9%
5	55,7%	36,7%
15	55,7%	24,5%
20	**57,3%**	**24,5%**
25	55,7%	29,5%
30	**57,3%**	**29,5%**
40	57,3%	19,8%
50	52,4%	27,8%
100	51,7%	29,5%

The results show basically the same results from the BoRIS system. The rule of thumb “number of observations divided by dimension of input variables” has the best results.

Experiment 12:

As seen before, the NN method using the 4 variables ‘no. turns’, ‘ca:#ia’, ‘ts_ord’, ‘uct’ (input4m) to predict Mean B Questions has the following results:

Table 44: Statistics from #turns, ca:#ia, ts_ord and uct as input variables on a NN to predict Mean Questions B

Pearson’s Correlation	R^2	$\overline{R^2}$
43,8%	**0,174**	**0,156**

Using the same variables from above, adding the age from the user as an input variable to predict ‘Mean B Questions’ has the following results:

Table 45: Statistics from #turns, ca:#ia, ts_ord, uct and age as input variables on a NN to predict Mean B

Pearson’s Correlation	R^2	$\overline{R^2}$
35%	**0,09**	**0,07**

The results are worse than the original results without ‘age’ as an input variable.

The same procedure was made with ‘gender’ as an input variable (0 for man, 1 for woman), and with ‘age’ and ‘gender’ together as input variables. The results are the following:

Table 46: Statistics from #turns, ca:#ia, ts_ord, uct and gender as input variables on a NN to predict ‘Mean B Questions’

Pearson’s Correlation	R^2	$\overline{R^2}$
37,9%	**0,115**	**0,099**

Table 47: Statistics from #turns, ca:#ia, ts_ord, uct, age and gender as input variables on a NN to predict ‘Mean B Questions’

Pearson’s Correlation	R^2	$\overline{R^2}$
29%	**0,021**	**-0,01**

These results show that neither of the two variables (‘age’ and ‘gender’) individually, nor both of them together, help improving the prediction capability from the neural network. The addition of variables most of the times does not help on the enhancement of the prediction, but sometimes if the value is high correlated to the target value and low correlated to the other input variables the results can get better, like on experiment 13.

<u>Experiment 13</u>:

Using the same ‘input4m’ from experiment 12 with the addition of Question B1 (binary) from the BoRIS´ questionnaire to predict ‘Mean B Questions’, the results become significantly better.

Table 48: Statistics from #turns, ca:#ia, ts_ord, uct and B1 as input variables on a NN to predict Mean B

Pearson’s Correlation	R^2	$\overline{R^2}$
64,4%	**0,418**	**0,395**

This enhancement happens due to the fact that Question B1 (binary) is highly correlated to ‘Mean B Question’, since it is part of the questionnaire which is filled out by the user himself. A similar variable to Question B1 was used on the PARADISE model as input variable, so the usage of Question B1 as an input variable can be used for a direct comparison to the model. So, the addition of highly correlated variables improves the $\overline{R^2}$.

Experiment 14:

This experiment has the purpose of analysing the different interpretations of 3 'logsig' neurons output. There are two different interpretations from the probabilities that come from the three output neurons and two different error analyses. A weighted interpretation of the probabilities that come from the output as seen on chapter 5.1.2 have usually a 30% worst result than choosing the higher probability, but using a different error analysis by using a weighted system, the results can be different.

Table 49: Statistics from experiment 14, weighted error analysis (PA:CO, WA_iso, IC to predict overall quality on the INSPIRE system config. 2 - 15 neurons on the hidden layer, trainbr as training function. 64 dialogues analysed.)

Interpretation of the 3 output neurons	Accuracy	Normal weighted error (+1 for any classification error)	Using a weighted error that uses 1 for a 1-class error and 2 for a 2-class error	Using a weighted error that uses 1 for a 1-class error and 3 for a 2-class error	Using a weighted error that uses 1 for a 1-class error and 4 for a 2-class error
Using weighted sum of probabilities	42,1%	37	40	43	46
Using higher probability	50%	32	39	49	68

The results show that if the 2-class error is considered to have a relatively big weight, the weighted sum of probabilities would be the best interpretation of the outputs, but since there are only 3 classes on this evaluation, the higher probability is the best option, considering that a smaller weighted error can be obtained just by using the 'middle' class (average). A weighted error evaluation would be more appropriate in the case that a larger number of classes was used, like six classes for instance. An error between 'zero' and 'five' would have a bigger impact than an error between 'zero' and 'two', and the accuracy would be not such a good evaluator in this case.

<u>Experiment 15</u>:

Other experiment tried was to use input variables that are in theory invariant among themselves. The predictor in a regression model are often called independent variables, but this term does not imply that the predictors are themselves statistically from one another. "Multicolinearity" is the term used to describe this case when the intercorrelation of input variables is high. This can be avoided by transforming a group of 9 variables into 3 factors through a factor analysis for instance.

The comparison, in this case, will be done between a neural network which uses the original 9 input variables and a neural network which uses the 3 factors (originated from the 9 variables). This would show if such a "small" neural network works better with a small quantity of input variables and how important it is to have input variables that are invariant among themselves.

The 3 factors originated from the set of 9 variables(#turns, CA:#IA, PA:CO, TS_ord, WA_iso, WPST, UCT, IC, #Sys.Questions) are invariant between themselves:

Table 50: Correlation between the 3 factors originated from input9 (BoRIS System - using a 9 input variables set (#turns, CA:#IA, PA:CO, TS_ord, WA_iso, WPST, UCT, IC, #Sys.Questions) with target value being 'Mean B Questions'.)

		factor_1	factor_2	factor_3
factor_1	Pearson Correlation	1	,000	,000
	Sig. (2-tailed)		1,000	1,000
	N	197	197	197
factor_2	Pearson Correlation	,000	1	,000
	Sig. (2-tailed)	1,000		1,000
	N	197	197	197
factor_3	Pearson Correlation	,000	,000	1
	Sig. (2-tailed)	1,000	1,000	
	N	197	197	197

Table 51: Statistics from input 9 on a NN to predict 'Mean B Questions' (One hidden layer - 21 neurons - Training function: trainbfg)

Correlation on training (in-sample) data	80,8%
R^2 on training (in-sample) data	0,648
$\overline{R^2}$ on training (in-sample) data	0,631
Correlation on test (out-of-sample) data	43,4%
R^2* on test (out-of-sample) data	0,171
$\overline{R^2}$ * on test (out-of-sample) data	0,129

*R^2 is a measure for training (in-sample) data originally, but the same formula was used on test (out-of-sample) data just for comparison

Using the 3 factors (that represent the 9 input variables above):

Table 52: Statistics from #turns, ca:#ia, ts_ord and uct as input variables on a NN to predict ‘Mean B Questions’ (One hidden layer - 62 neurons - Training function: trainbfg.)

Correlation on training (in-sample) data	58,6%
R^2 on training (in-sample) data	0,341
$\overline{R^2}$ on training (in-sample) data	0,331
Correlation on test (out-of-sample) data	39,1%
R^2* on test (out-of-sample) data	0,123
$\overline{R^2}$ * on test (out-of-sample) data	0,109

*R^2 is a measure for training (in-sample) data originally, but the same formula was used on test (out-of-sample) data just for comparison

The results on the training (in-sample) data shows that the amount of input variables improves greatly the training (in-sample) correlation. The 3 factors had a smaller correlation and R^2 than the 9 input variables, due to the fact that not all characteristics are properly summarized on the data reduction to transform 9 input variables into three factors. There is a loss of data in this transformation.

<u>Experiment 16</u>:

Doing a similar experiment with 52 input variables from the INSPIRE system and reducing them to five factors (invariant among themselves) we get even worse results.

Table 53: Statistics from 5 factors representing 52 input variables on a NN to predict overall quality. (INSPIRE system, neural network with one hidden layer - 12 neurons. Training function: Trainbr)

Correlation on test (out-of-sample) data	-9,4%
R^2* on test (out-of-sample) data	-0,17
$\overline{R^2}$ * on test (out-of-sample) data	-0,27

*R^2 is a measure for training (in-sample) data originally, but the same formula was used on test (out-of-sample) data just for comparison

The results, as seen above are terrible, the same happens on the linear regression approach:

Table 54: Statistics from 5 factors representing 52 input variables on a linear regression to predict overall quality

Correlation on test (out-of-sample) data	25%
R²* on test (out-of-sample) data	-0,23
$\overline{R^2}$ * on test (out-of-sample) data	-0,26

This way, we can take as conclusion that the factor analysis on the input variables does not necessarily means that the prediction will be good (even if the amount of input variables is big).

Experiment 17:

Doing another similar experiment with the INSPIRE system but this time with all the input variables that are error classifications (see [1]) and turning them in five factors, invariant among themselves, we try to predict the target variable "easy learning". The correlation between factor 2 and "easy learning" is the biggest one that a factor originated from input variables has with a target value. The following table shows the correlations between the factors and the target value.

Table 55: Correlations from 5 factors acquired from all "error coding" variables from the INSPIRE system (configuration 1, free-woz)

		REGR factor score 1 for analysis 1	REGR factor score 2 for analysis 1	REGR factor score 3 for analysis 1	REGR factor score 4 for analysis 1	REGR factor score 5 for analysis 1
easy learning	Pearson Correlation	-,158	-,661(**)	,018	-,190	-,039
	Sig. (2-tailed)	,219	,000	,888	,138	,764
	N	62	62	62	62	62

Results:

Table 56: Statistics from NN with input variables (5 factors that represent all "error classifications", target value: "easy learning". INSPIRE system, neural network with one hidden layer - 25 neurons)

Correlation on test (out-of-sample) data	53,7%
R²* on test (out-of-sample) data	0,275
$\overline{R^2}$ * on test (out-of-sample) data	0,211

*R² is a measure for training (in-sample) data originally, but the same formula was used on test (out-of-sample) data just for comparison

This is actually a good result considering this is strictly test (out-of-sample) data and the past experiments with factors from input variables, but since this model was only checked with 62 valid dialogues, it should be analysed in a bigger data set (this error "classification" data was only available for the INSPIRE system configuration 1 – free WoZ by the time this thesis was written).

Trying this same experiment with only the three factors that correlate the most (1, 2 and 4), we obtain the following results:

Table 57: Statistics from Neural Networks with input variables: 3 factors from "error classifications", target value: "easy learning" (INSPIRE system, neural network with one hidden layer - 35 neurons. Training function: Trainbr).

Correlation on test (out-of-sample) data	56,2%
R²* on test (out-of-sample) data	0,302
$\overline{R^2}$ * on test (out-of-sample) data	0,281

*R² is a measure for training (in-sample) data originally, but the same formula was used on test (out-of-sample) data just for comparison

Trying this same experiment with only the two factors that correlate the most (2 and 4) we obtain the following results:

Table 58: Statistics from NN with input variables: 2 factors from "error classifications", target value: "easy learning" (INSPIRE system, neural network with one hidden layer - 35 neurons. Training function: Trainbr).

Correlation on test (out-of-sample) data	57,1%
R^2* on test (out-of-sample) data	0,325
$\overline{R^2}$ * on test (out-of-sample) data	0,299

*R^2 is a measure for training (in-sample) data originally, but the same formula was used on test (out-of-sample) data just for comparison

Trying this same experiment with only the one factors that correlate the most (2) we obtain the following results:

Table 59: Statistics from NN with input variables: 1 factor from "error classifications", target value: "easy learning" (INSPIRE system, NN with one hidden layer - 15 neurons. Training function: Trainbr).

Correlation on test (out-of-sample) data	65,5%
R^2* on test (out-of-sample) data	0,402
$\overline{R^2}$ * on test (out-of-sample) data	0,392

*R^2 is a measure for training (in-sample) data originally, but the same formula was used on test (out-of-sample) data just for comparison

This shows that neural networks are pretty sensitive to the amount of variables and to noise from other variables that do not correlate much with the target values. The results are pretty good in comparison with the other ones in this chapter.

5.4 Intersystem/interconfiguration prediction models

In this chapter, experiments were made to test how well a model trained in one system can predict the data from tests on another system and from the same system with different configurations.

Experiment 18:

The experiments consist basically on cross-system and cross-configuration prediction models on all the four approaches seen on the previous chapters. The input variables selected for this experiment were the 3 best correlated ones in common for all the systems and different configurations.

Linear Regression

Table 60: Statistics about intersystem/interconfiguration using linear regression (3 input variables: #turns, #CA:IA, PA:CO)

Training	Test	Target	Pearson´s correlation (on test data)	$\overline{R^2}$
BoRIS	INSPIRE Dataset 1	Question B0 - Overall Quality rating	38,5%	0,435
BoRIS	INSPIRE Dataset 2	Question B0 - Overall Quality rating	51,2%	0,185
INSPIRE Dataset 1	INSPIRE Dataset 2	Question B0 - Overall Quality rating	36,9%	-0,32
INSPIRE Dataset 1	BoRIS	Question B0 - Overall Quality rating	32,2%	0,03
INSPIRE Dataset 2	INSPIRE Dataset 1	Question B0 - Overall Quality rating	39,9%	-0,34
INSPIRE Dataset 2	BoRIS	Question B0 - Overall Quality rating	35,2%	-0,02

The predictions using training (in-sample) data from the BoRIS system and testing on INSPIRE are the best ones. Even with only three input variables 43% of the variance can be covered on the test (out-of-sample) data from Inspire system configuration 1 (free w.o.z.). The other way around INSPIRE → BoRIS has bad results. There is a bigger amount of data on the BoRIS system (197 dialogues), this means the training (in-sample) data is larger, cover different dialogues and therefore predicting better other dialogues, even if they are from other systems. The other way around does not work that good, since the INSPIRE system does not have that much training data as BoRIS.

Neural Networks

Definition of the ideal number of neurons:

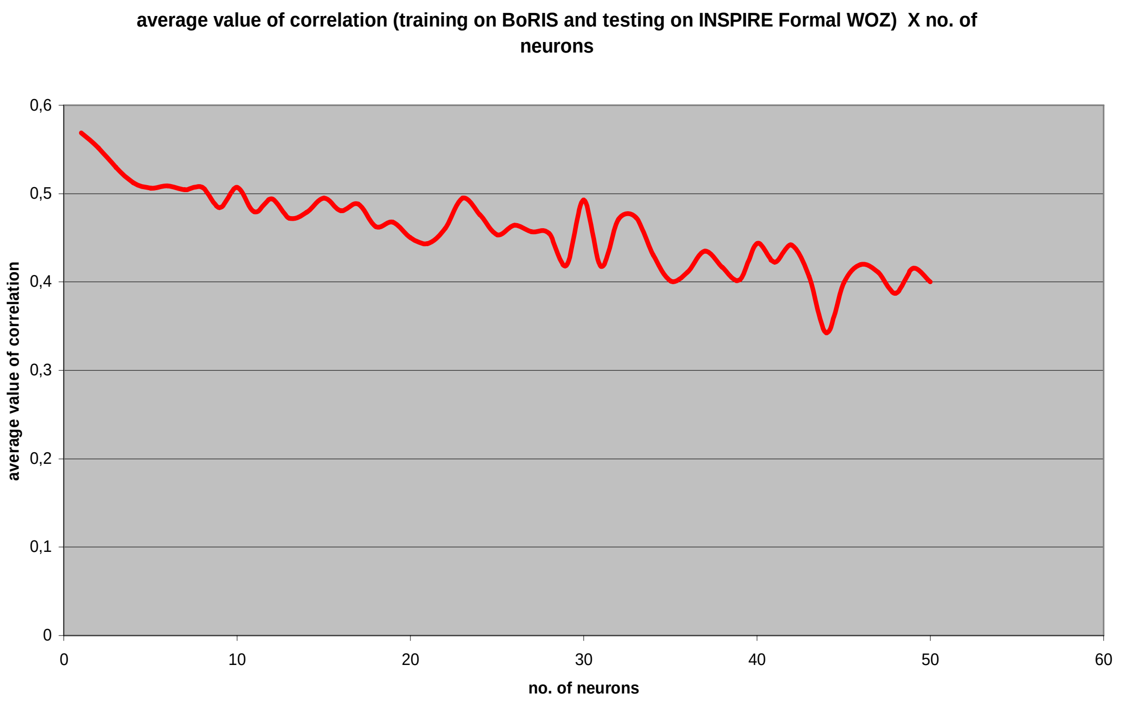

Figure 10: No. of neurons X average value of correlation

Conclusion from figure 10: the less neurons on the hidden layer, the better the correlations are, when it comes to intersystem prediction modelling. This means that the linear regression (the most simple correlation on a neural network) is the best way to generalize in intersystem prediction models.

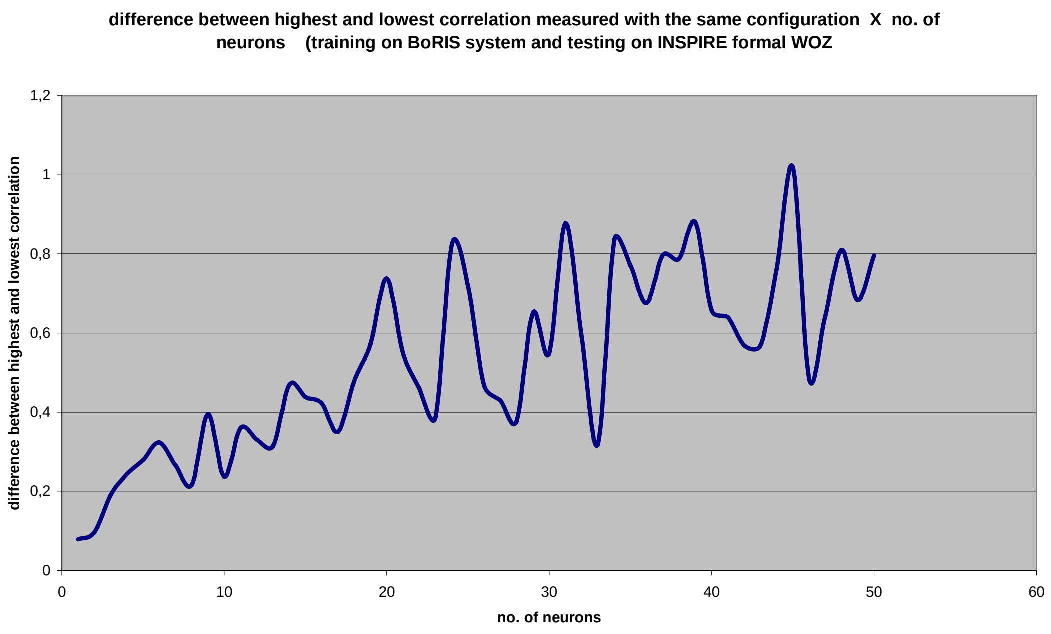

Figure 11: No. of neurons X difference between highest and lowest correlation

Conclusion from figure 11: the less neurons on the hidden layer, the more stable the values for the correlations are (less dependent on initial values.

Table 61: Statistics about intersystem/interconfiguration using neural networks (Neural network: 5 ‘tansig’ neurons on the hidden layer + 1 ‘purelin’ neuron as output, Best training function: ‘trainbfg’, 3 input variables: #turns, #CA:IA, PA:CO)

Training	Test	Target	Pearson´s correlation (on test data)	$\overline{R^2}$
BoRIS	INSPIRE Dataset 1	Question B0 - Overall Quality rating	43%	-0,01
BoRIS	INSPIRE Dataset 2	Question B0 - Overall Quality rating	61%	0,09
INSPIRE Dataset 1	INSPIRE Dataset 2	Question B0 - Overall Quality rating	49%	0,06

INSPIRE Dataset 1	BoRIS	Question B0 - Overall Quality rating	34,8%	0,01
INSPIRE Dataset 2	INSPIRE Dataset 1	Question B0 - Overall Quality rating	45,9%	-0,05
INSPIRE Dataset 2	BoRIS	Question B0 - Overall Quality rating	37,1%	0,02

Regression trees

Table 62: Statistics about intersystem/interconfiguration using regression trees (3 input variables: #turns, #CA:IA, PA:CO)

Training	Test	Target	Pearson´s correlation (on test data)	$\overline{R^2}$
BoRIS	INSPIRE Dataset 1	Question B0 - Overall Quality rating	38,1%	-0.20
BoRIS	INSPIRE Dataset 2	Question B0 - Overall Quality rating	53,6%	0,25
INSPIRE Dataset 1	INSPIRE Dataset 2	Question B0 - Overall Quality rating	55,1%	0,17
INSPIRE Dataset 1	BoRIS	Question B0 - Overall Quality rating	29,5%	-0,09
INSPIRE Dataset 2	INSPIRE Dataset 1	Question B0 - Overall Quality rating	48,2%	-0,40
INSPIRE Dataset 2	BoRIS	Question B0 - Overall Quality rating	32,7%	-0,03

Classification Trees

Table 63: Statistics about intersystem/interconfiguration using classification trees (3 input variables: #turns, #CA:IA, PA:CO)

Training	Test	Target	Accuracy on 3-class evaluation
BoRIS	INSPIRE Dataset 1	Question B0 - Overall Quality rating	51,4% (80% pruned tree has the best results)
BoRIS	INSPIRE Dataset 2	Question B0 - Overall Quality rating	56,5% (80% pruned tree has the best results)
INSPIRE Dataset 1	INSPIRE Dataset 2	Question B0 - Overall Quality rating	46,4%(tree unpruned has the best results)
INSPIRE Dataset 1	BoRIS	Question B0 - Overall Quality rating	46,1%
INSPIRE Dataset 2	INSPIRE Dataset 1	Question B0 - Overall Quality rating	48,5% (tree unpruned has the best results)
INSPIRE Dataset 2	BoRIS	Question B0 - Overall Quality rating	46,7%

With these results we can see that the bigger the pruning on the tree is, the better it is to predict from Boris to Inspire configuration 1. Below is how the classification tree that has the best accuracy:

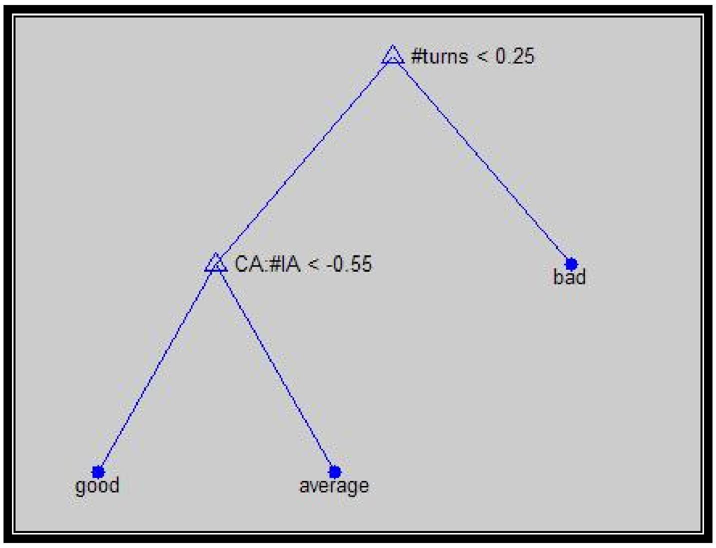

Figure 12: Classification tree(3 input variables #turns, #CA:IA, PA:CO to predict B0) that has the best accuracy(56,5%) on intersystem models. Whenever the original tree is pruned 80% the results get better.

Conclusion

The linear regression is in average the best method on the “Train Boris → Test Inspire” predictions when it comes to R^2, but does a terrible job on the interconfiguration prediction for the INSPIRE system. The neural networks approach has better Pearson´s correlations than the linear regression for all the intersystem and interconfiguration experiments.

The Regression trees had a very good result considering the input set has only three variables: 53,6% correlation ($\overline{R^2}$ = 0,25) training on the Boris system and testing on the INSPIRE system 2nd configuration data and 55,1% correlation ($\overline{R^2}$ = 0,25) on the “INSPIRE 1st configuration → INSPIRE 2nd configuration” experiment, but all the other results are pretty bad.

In order to test the reliability and generalization of intersystem or interconfiguration prediction models, a larger sample from a bigger amount of experiments is needed.