# HW5

# Algorithm 1

We decided to use the LUCS-KDD implementation of the CBA algorithm given here:

<http://cgi.csc.liv.ac.uk/~frans/KDD/Software/CBA/cba.html>

We assume the tool works correctly, and therefor we did not manually confirm the results on the big dataset.

We did run experiments on 10 records that we engineered for testing, which the tool supplied the right results for.

## Step 0:

The data transformation was from the output of HW2, the transaction list: “ID”, “List”, “Of”,”Canoninized”, “Words”, “Class” to 1 2 3 4 5 (the tool requirement).

For doing so – We built a hashmap between words to an id for both – the words and the topics.

Afterwards, we restructure the file to the correct form: ignoring article ID, reorganizing numbers to be sorted (a requirement of the tool).

Each line has one topic in the end, and the ids of the topic are bigger than those of the words (another requirement of the tool).

## Step 1 + 2:

The tool we’re using automatically determine which are test and which are validation sets, and therefore we did not have to randomly chose those ourselves.

The tool has a default of confidence 80%, support 20%.

These defaults are too high for our dataset (no rules are being generated).

The nice thing is that with 20% as support – the code ran really fast.

We started exploring the values around 0.01% to 1% for support.

It looks like the optimal support for this data-set is around 0.8%.

It worth mentioning that lower support has longer run time.

For this data set at least, we did not notice big difference in run time for different confidence within the same support.

It also looks like the run time difference between 0.1% to 0.08% of confidence is exponential, while it is much lower for 0.1% and 1%, and 1% to 20%.

One can notice that when reducing the confidence more rules are being added, but the rules from the higher confidence remain. It means we can build a mechanism to reuse this data.

Step two is being done as part of this section, in the “startClassification” method.

We supply the code of the apriori opensource we used.

In this code you can clearly see that the library prunes the rules supplied in the same approach required in class.

Example data:

SETTINGS

--------

Training file name = ../output/WordList.csv.fix

Support (default 20%) = 0.1

Confidence (default 80%) = 90.0

Number of classes = 119

Reading input file: ../output/WordList.csv.fix

Number of records = 12116

Number of columns = 6971

Min support = 12.12 (records)

START APRIORI-TFP CBA

------------------------------

Max number of CARS = 80000

Max size antecedent = 2147483647

Support = 0.1, Confidence = 90.0

Minimum support = 6.06 (Records)

Max num frequent sets = 2147483647

Max size of antecedent = 2147483647

Number of records in training set = 6058

NOTE: Data set reordered

Creating P-tree table

Apriori-TFP with X-Checking

Minimum support threshold = 0.1% (6.06 records)

Generation time = 297.32 seconds (4.96 mins)

Number of frequent sets = 34155441

Number of T-tree nodes created = 35542464

Number of T-tree Updates = 67949945

Number of rules = 54

Accuracy = 25.04

AUC value = 0.0121

(1) {390 799} -> {6870} 100.0

(2) {1438} -> {6861} 100.0

(3) {618 799} -> {6870} 100.0

(4) {1 390} -> {6870} 100.0

(5) {390 618} -> {6870} 100.0

(6) {521} -> {6853} 100.0

(7) {1191} -> {6861} 100.0

(8) {1 799} -> {6870} 100.0

(9) {390 619} -> {6870} 100.0

(10) {1 619} -> {6870} 100.0

(11) {1541 1578} -> {6881} 100.0

(12) {44} -> {6853} 100.0

(13) {2502} -> {6853} 100.0

(14) {1 267} -> {6853} 100.0

(15) {284} -> {6853} 100.0

(16) {160} -> {6853} 100.0

(17) {994} -> {6870} 100.0

(18) {996} -> {6870} 100.0

(19) {864 1321} -> {6853} 100.0

(20) {1465 835} -> {6853} 100.0

(21) {278 282} -> {6853} 100.0

(22) {388 1578} -> {6881} 100.0

(23) {2867} -> {6853} 100.0

(24) {1 296} -> {6853} 100.0

(25) {1 1537} -> {6853} 100.0

(26) {1 280} -> {6853} 100.0

(27) {618 619} -> {6870} 100.0

(28) {929 1127} -> {6870} 100.0

(29) {953 524} -> {6880} 100.0

(30) {762 760} -> {6887} 100.0

(31) {1210 2834} -> {6909} 100.0

(32) {1486} -> {6853} 100.0

(33) {2846} -> {6853} 100.0

(34) {1444} -> {6853} 100.0

(35) {1491} -> {6853} 100.0

(36) {1398} -> {6853} 100.0

(37) {1949} -> {6853} 100.0

(38) {1854} -> {6870} 100.0

(39) {2078} -> {6880} 100.0

(40) {2905} -> {6909} 100.0

(41) {483} -> {6853} 95.23

(42) {475} -> {6870} 94.44

(43) {1094} -> {6853} 93.75

(44) {286} -> {6853} 92.85

(45) {1 1465} -> {6853} 91.66

(46) {1252} -> {6853} 90.9

(47) {1809} -> {6853} 90.9

(48) {1 134} -> {6853} 90.0

(49) {24 718} -> {6870} 90.0

(50) {88 967} -> {6915} 90.0

(51) {2030} -> {6853} 90.0

(52) {2312} -> {6853} 90.0

(53) {2551} -> {6853} 90.0

(54) null -> {6861} 0.0

Same support, but lower threshold:

SETTINGS

--------

Training file name = ../output/WordList.csv.fix

Support (default 20%) = 0.1

Confidence (default 80%) = 50.0

Number of classes = 119

Reading input file: ../output/WordList.csv.fix

Number of records = 12116

Number of columns = 6971

Min support = 12.12 (records)

START APRIORI-TFP CBA

------------------------------

Max number of CARS = 80000

Max size antecedent = 2147483647

Support = 0.1, Confidence = 50.0

Minimum support = 6.06 (Records)

Max num frequent sets = 2147483647

Max size of antecedent = 2147483647

Number of records in training set = 6058

NOTE: Data set reordered

Creating P-tree table

Apriori-TFP with X-Checking

Minimum support threshold = 0.1% (6.06 records)

Generation time = 290.65 seconds (4.84 mins)

Number of frequent sets = 34155441

Number of T-tree nodes created = 35542464

Number of T-tree Updates = 67949945

Number of rules = 236

Accuracy = 32.11

AUC value = 0.0256

(1) {390 799} -> {6870} 100.0

(2) {1438} -> {6861} 100.0

(3) {618 799} -> {6870} 100.0

(4) {1 390} -> {6870} 100.0

(5) {390 618} -> {6870} 100.0

(6) {521} -> {6853} 100.0

(7) {1191} -> {6861} 100.0

(8) {1 799} -> {6870} 100.0

(9) {390 619} -> {6870} 100.0

(10) {1 619} -> {6870} 100.0

(11) {1541 1578} -> {6881} 100.0

(12) {44} -> {6853} 100.0

(13) {2502} -> {6853} 100.0

(14) {1 267} -> {6853} 100.0

(15) {284} -> {6853} 100.0

(16) {160} -> {6853} 100.0

(17) {994} -> {6870} 100.0

(18) {996} -> {6870} 100.0

(19) {864 1321} -> {6853} 100.0

(20) {1465 835} -> {6853} 100.0

(21) {278 282} -> {6853} 100.0

(22) {388 1578} -> {6881} 100.0

(23) {2867} -> {6853} 100.0

(24) {1 296} -> {6853} 100.0

(25) {1 1537} -> {6853} 100.0

(26) {1 280} -> {6853} 100.0

(27) {618 619} -> {6870} 100.0

(28) {929 1127} -> {6870} 100.0

(29) {953 524} -> {6880} 100.0

(30) {762 760} -> {6887} 100.0

(31) {1210 2834} -> {6909} 100.0

(32) {1486} -> {6853} 100.0

(33) {2846} -> {6853} 100.0

(34) {1444} -> {6853} 100.0

(35) {1491} -> {6853} 100.0

(36) {1398} -> {6853} 100.0

(37) {1949} -> {6853} 100.0

(38) {1854} -> {6870} 100.0

(39) {2078} -> {6880} 100.0

(40) {2905} -> {6909} 100.0

(41) {483} -> {6853} 95.23

(42) {475} -> {6870} 94.44

(43) {1094} -> {6853} 93.75

(44) {286} -> {6853} 92.85

(45) {1 1465} -> {6853} 91.66

(46) {1252} -> {6853} 90.9

(47) {1809} -> {6853} 90.9

(48) {1 134} -> {6853} 90.0

(49) {24 718} -> {6870} 90.0

(50) {88 967} -> {6915} 90.0

(51) {2030} -> {6853} 90.0

(52) {2312} -> {6853} 90.0

(53) {2551} -> {6853} 90.0

(54) {1 485} -> {6853} 88.88

(55) {1 1132} -> {6853} 88.88

(56) {1374} -> {6853} 88.88

(57) {1927} -> {6853} 88.88

(58) {2083} -> {6853} 88.88

(59) {1279} -> {6853} 88.88

(60) {1031} -> {6853} 88.88

(61) {299} -> {6853} 88.88

(62) {485} -> {6853} 88.23

(63) {1323} -> {6853} 87.5

(64) {1681} -> {6853} 87.5

(65) {2193} -> {6870} 87.5

(66) {1959} -> {6879} 87.5

(67) {976} -> {6861} 86.84

(68) {1463} -> {6861} 86.66

(69) {1686} -> {6853} 85.71

(70) {296} -> {6853} 85.0

(71) {290} -> {6853} 84.61

(72) {618} -> {6870} 84.37

(73) {619} -> {6870} 83.33

(74) {497} -> {6853} 83.33

(75) {2107} -> {6853} 83.33

(76) {1 558} -> {6853} 83.33

(77) {589} -> {6853} 82.85

(78) {721 524} -> {6880} 81.81

(79) {791 919} -> {6870} 81.81

(80) {135} -> {6853} 81.81

(81) {1909} -> {6853} 80.95

(82) {277 746} -> {6881} 80.0

(83) {2264} -> {6853} 80.0

(84) {1440} -> {6870} 80.0

(85) {173} -> {6861} 79.31

(86) {1465} -> {6853} 79.16

(87) {1051} -> {6870} 78.57

(88) {839} -> {6861} 78.57

(89) {721 1602} -> {6880} 77.77

(90) {1201 1203} -> {6880} 77.77

(91) {270 1365} -> {6853} 77.77

(92) {1 1500} -> {6853} 77.77

(93) {2482} -> {6870} 77.77

(94) {426} -> {6897} 76.92

(95) {1944} -> {6861} 76.92

(96) {1621} -> {6853} 76.92

(97) {519} -> {6853} 76.92

(98) {291} -> {6853} 76.19

(99) {1369} -> {6853} 76.19

(100) {2284 1967} -> {6870} 75.0

(101) {259} -> {6861} 75.0

(102) {1578} -> {6881} 75.0

(103) {782 719} -> {6871} 75.0

(104) {1251} -> {6853} 74.07

(105) {394} -> {6853} 73.91

(106) {267} -> {6853} 73.68

(107) {1280} -> {6853} 73.68

(108) {753} -> {6870} 73.68

(109) {1253} -> {6853} 73.33

(110) {53} -> {6853} 72.72

(111) {1562} -> {6853} 72.72

(112) {2442} -> {6875} 72.72

(113) {390} -> {6870} 72.54

(114) {755} -> {6853} 72.22

(115) {224} -> {6853} 72.22

(116) {352} -> {6853} 72.22

(117) {1539} -> {6853} 71.42

(118) {283} -> {6853} 71.42

(119) {1411} -> {6853} 70.83

(120) {1132} -> {6853} 70.0

(121) {545 877} -> {6853} 70.0

(122) {1291} -> {6861} 70.0

(123) {1564} -> {6870} 70.0

(124) {2489} -> {6870} 70.0

(125) {2834} -> {6909} 69.23

(126) {1 957} -> {6853} 69.23

(127) {545 277} -> {6881} 69.23

(128) {899} -> {6853} 68.75

(129) {946} -> {6853} 68.75

(130) {78} -> {6870} 68.75

(131) {2284} -> {6870} 68.42

(132) {558} -> {6853} 66.66

(133) {328} -> {6853} 66.66

(134) {555} -> {6853} 66.66

(135) {472} -> {6870} 66.66

(136) {1672} -> {6879} 66.66

(137) {1214} -> {6853} 66.66

(138) {1352} -> {6853} 66.66

(139) {303} -> {6870} 66.66

(140) {279} -> {6853} 65.21

(141) {1321} -> {6853} 65.0

(142) {288} -> {6853} 65.0

(143) {1091} -> {6853} 64.86

(144) {967 2063} -> {6915} 64.7

(145) {455 524} -> {6880} 64.28

(146) {1155 1162} -> {6871} 64.28

(147) {469} -> {6870} 64.0

(148) {929} -> {6870} 64.0

(149) {1382} -> {6853} 63.63

(150) {579} -> {6870} 63.63

(151) {2934} -> {6870} 63.63

(152) {372} -> {6879} 63.63

(153) {1801} -> {6881} 63.63

(154) {1162} -> {6871} 62.96

(155) {568} -> {6861} 62.5

(156) {1 47} -> {6853} 62.5

(157) {1007} -> {6853} 62.5

(158) {2109} -> {6870} 62.5

(159) {1152 1155} -> {6871} 61.53

(160) {941} -> {6853} 61.53

(161) {66} -> {6853} 61.53

(162) {285} -> {6853} 61.53

(163) {354} -> {6853} 60.86

(164) {278} -> {6853} 60.86

(165) {570} -> {6853} 60.71

(166) {1676 951} -> {6881} 60.0

(167) {1142} -> {6853} 60.0

(168) {814} -> {6870} 60.0

(169) {676} -> {6853} 59.09

(170) {584} -> {6870} 59.09

(171) {1676 1733} -> {6881} 58.82

(172) {1170} -> {6881} 58.82

(173) {802} -> {6870} 58.82

(174) {83} -> {6853} 58.82

(175) {51} -> {6853} 58.82

(176) {730} -> {6870} 58.33

(177) {707} -> {6881} 58.33

(178) {1514} -> {6881} 58.33

(179) {1733} -> {6881} 57.69

(180) {1676 1868} -> {6881} 57.14

(181) {1 1066} -> {6853} 57.14

(182) {126} -> {6853} 57.14

(183) {995} -> {6870} 57.14

(184) {385} -> {6853} 57.14

(185) {699} -> {6881} 56.52

(186) {1066} -> {6853} 56.41

(187) {54} -> {6853} 56.25

(188) {2474} -> {6880} 56.25

(189) {702} -> {6881} 56.25

(190) {1155 1150} -> {6871} 56.0

(191) {757} -> {6870} 55.0

(192) {1889} -> {6880} 55.0

(193) {357} -> {6853} 54.83

(194) {275} -> {6853} 53.84

(195) {397} -> {6853} 53.84

(196) {862} -> {6853} 53.84

(197) {1384} -> {6853} 53.84

(198) {1320} -> {6870} 53.84

(199) {1646} -> {6872} 53.84

(200) {1698} -> {6881} 53.84

(201) {1 60} -> {6853} 53.84

(202) {1150} -> {6871} 53.57

(203) {1365} -> {6853} 53.57

(204) {1306} -> {6853} 53.33

(205) {1731} -> {6881} 53.33

(206) {1676 1933} -> {6870} 53.33

(207) {58} -> {6853} 53.12

(208) {543} -> {6853} 52.94

(209) {1038} -> {6853} 52.63

(210) {471} -> {6870} 52.63

(211) {1937} -> {6881} 52.63

(212) {833} -> {6870} 52.17

(213) {240} -> {6861} 52.17

(214) {280} -> {6853} 52.17

(215) {1441} -> {6853} 52.0

(216) {834} -> {6870} 51.85

(217) {245} -> {6853} 51.61

(218) {347} -> {6853} 51.28

(219) {247} -> {6853} 50.0

(220) {418} -> {6853} 50.0

(221) {1537} -> {6853} 50.0

(222) {1192} -> {6870} 50.0

(223) {1407} -> {6871} 50.0

(224) {816} -> {6853} 50.0

(225) {522} -> {6853} 50.0

(226) {892} -> {6853} 50.0

(227) {1 122} -> {6853} 50.0

(228) {877} -> {6853} 50.0

(229) {560 561} -> {6881} 50.0

(230) {1545} -> {6853} 50.0

(231) {2187} -> {6861} 50.0

(232) {2400} -> {6871} 50.0

(233) {1532} -> {6871} 50.0

(234) {1206} -> {6879} 50.0

(235) {2678} -> {6893} 50.0

(236) null -> {6861} 0.0

# Algorithm 2