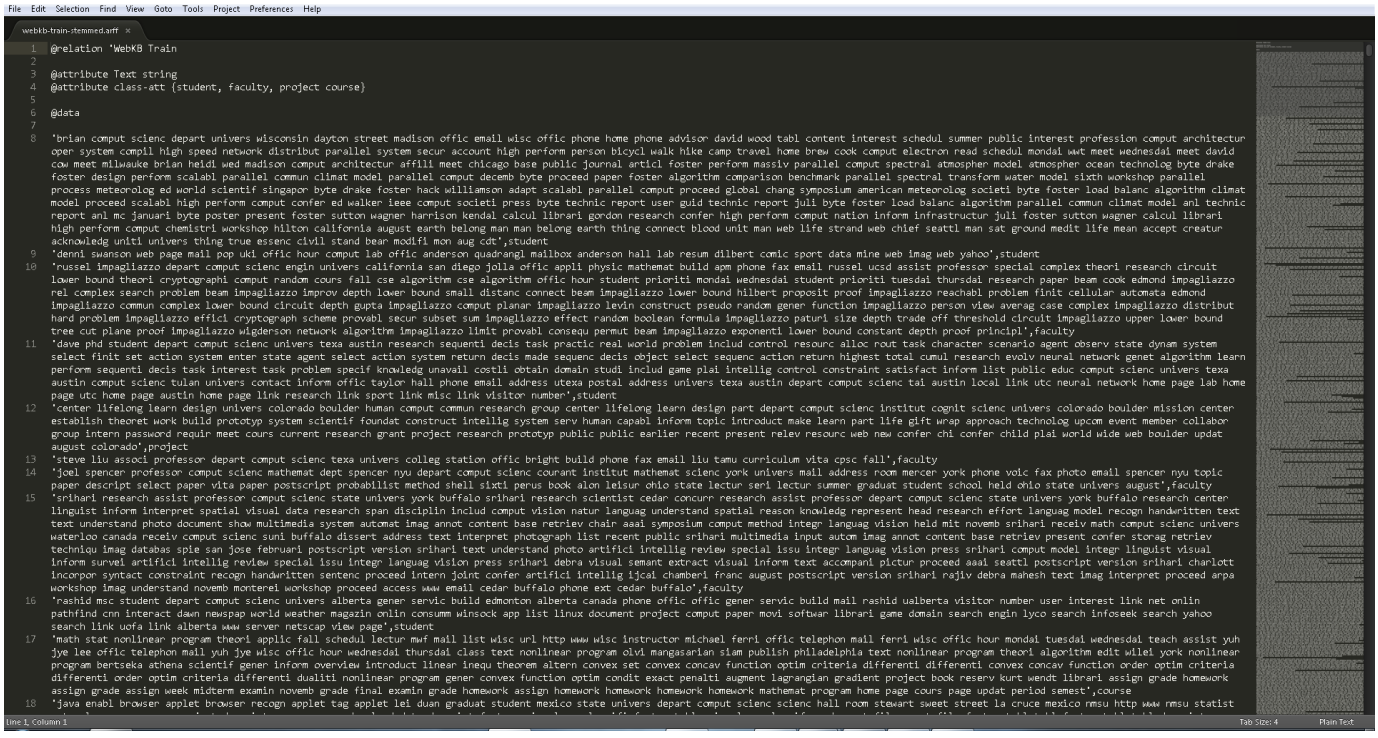
Maciej Medyk – CAP6776 – Information Retrieval – Homework 02

Report – [20.00pt] – Using Weka on “webkb-train-stemmed.txt” and “webkb-test-stemmed.txt”

The files received in the homework were in word format (Image 01) and could not be loaded directly to Weka.
I wrote the program using C# (See Addendum) that would convert the text into string based ARFF files (Image 02).
Both webkb-test-stemmed.txt and webkb-train -stemmed.txt files have been converted this way to new files webkb-test-stemmed.arff and webkb-train-stemmed.arff

Image01 (webkb-train-stemmed.txt)

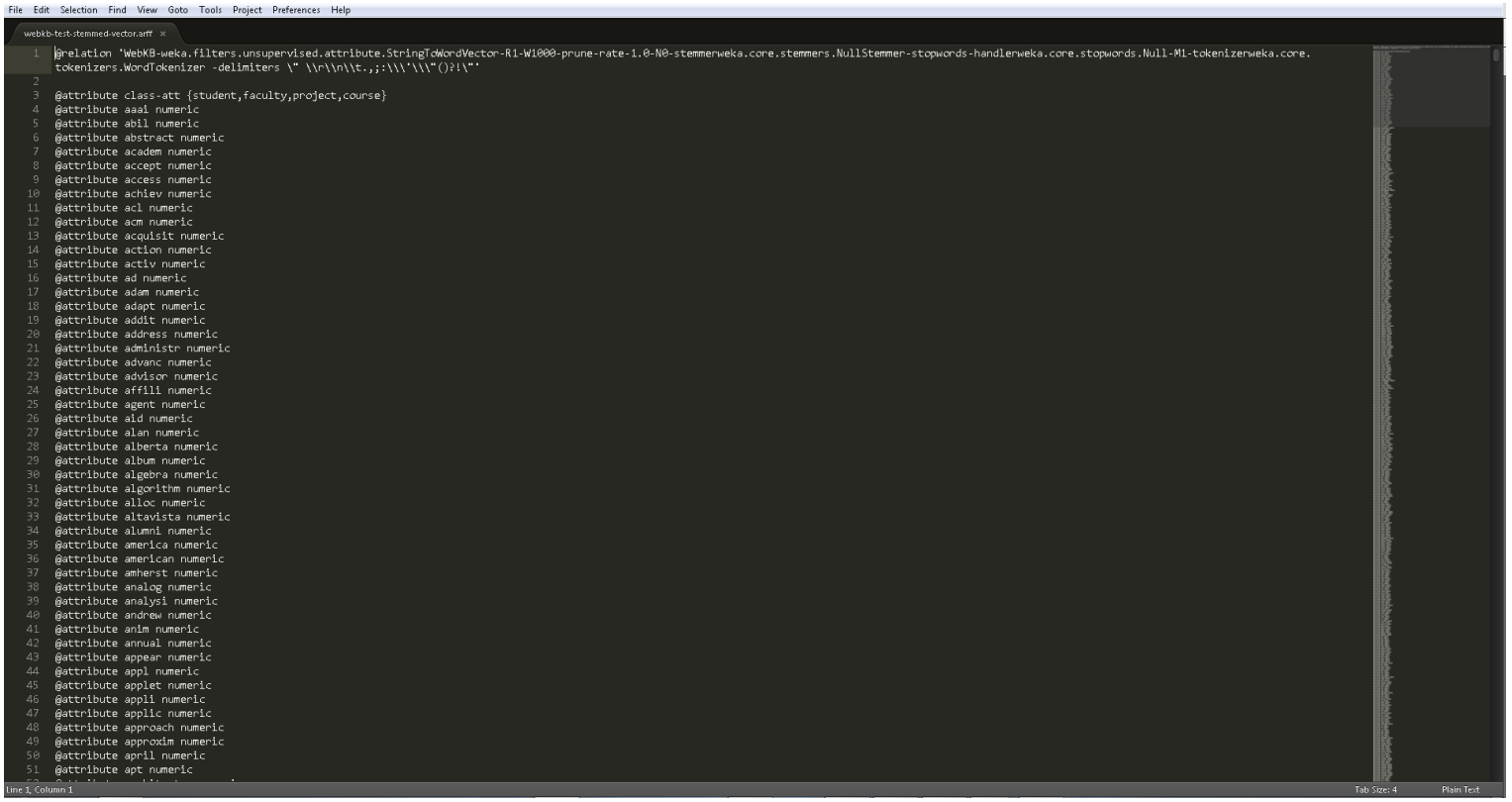
Image 02 (webkb-train-stemmed.arff)



Once ARFF files were obtained it was possible to load them into WEKA where I was able to convert them further using StringToWordVector in order to create ARRF files that are vector based (Image 03 and Image 04). Initially I did booelan word count and later I did integer word count.

Image 03 (webkb-train-stemmed-vector.arff)

Image 04 (webkb-test-stemmed-vector.arff)



Weka has StringToWordVector as filter which shows that training file has been loaded to the program (Image 05). Weka also displayed Boolean word count once the file was loaded (Image 06).

Image 05 (webkb-train-stemmed-vector.arff loaded into Weka)

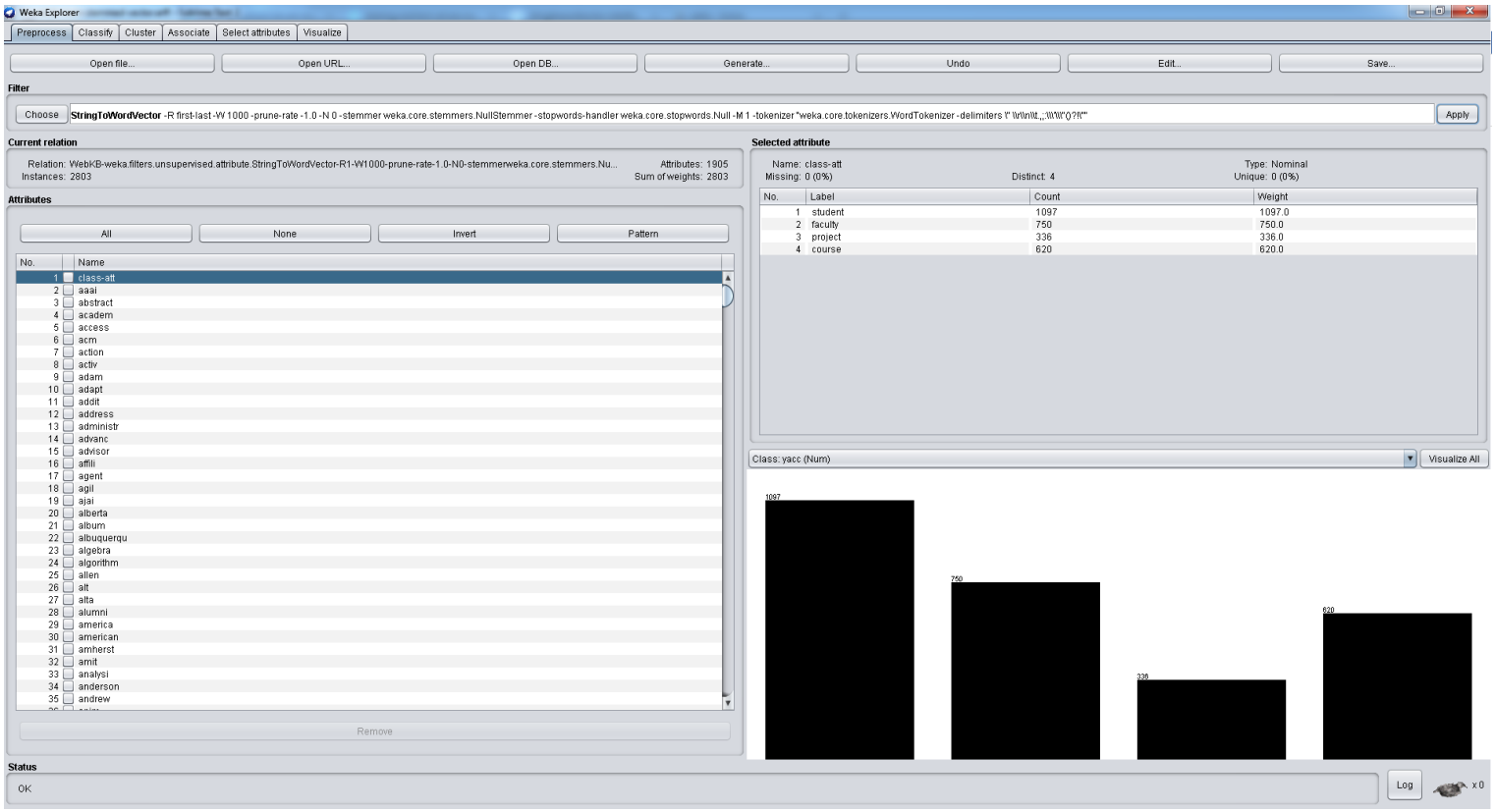
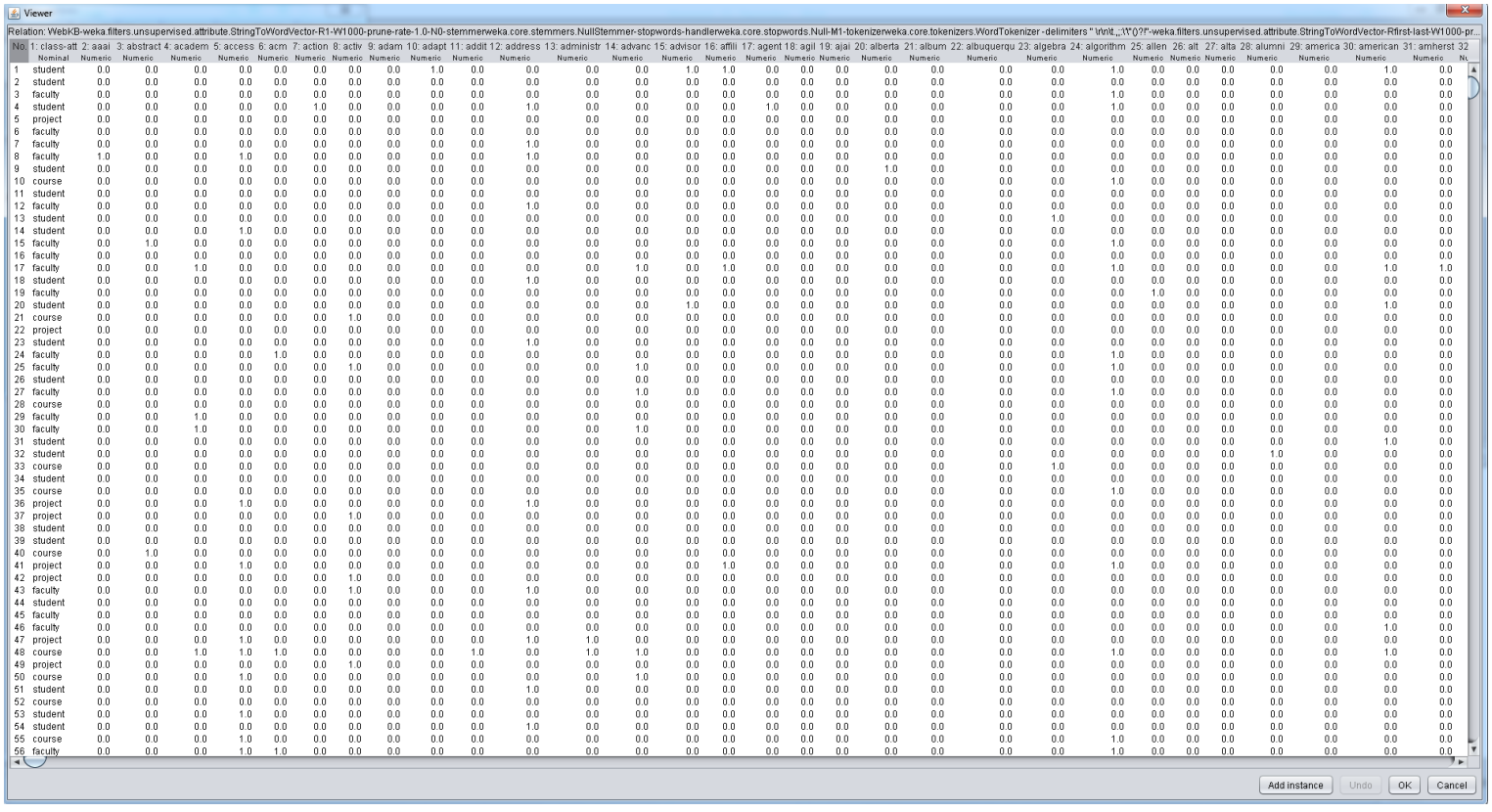


Image 06 (boolean word count in Weka)



Once I had both files converted I was able to run Naïve Bayes classification using 10-fold cross-validation (Image 07). I was able to obtain results in Image 08 and accuracy was calculated to 76.49 % ((885 + 471 + 247 + 541) / 2803).

Image 07 (running NaïveBayes classifier using 10-fold cross validation having Boolean word count)

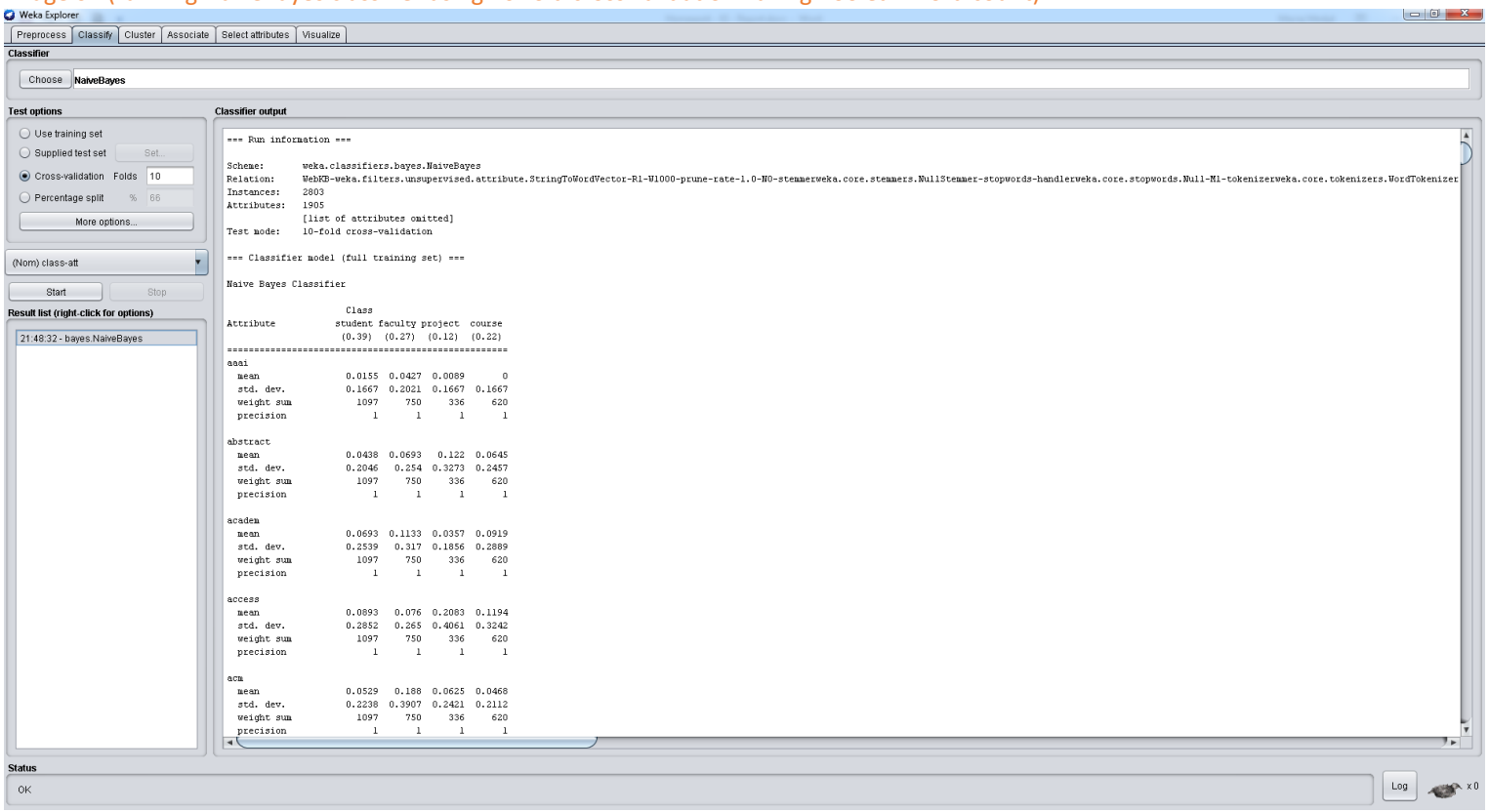
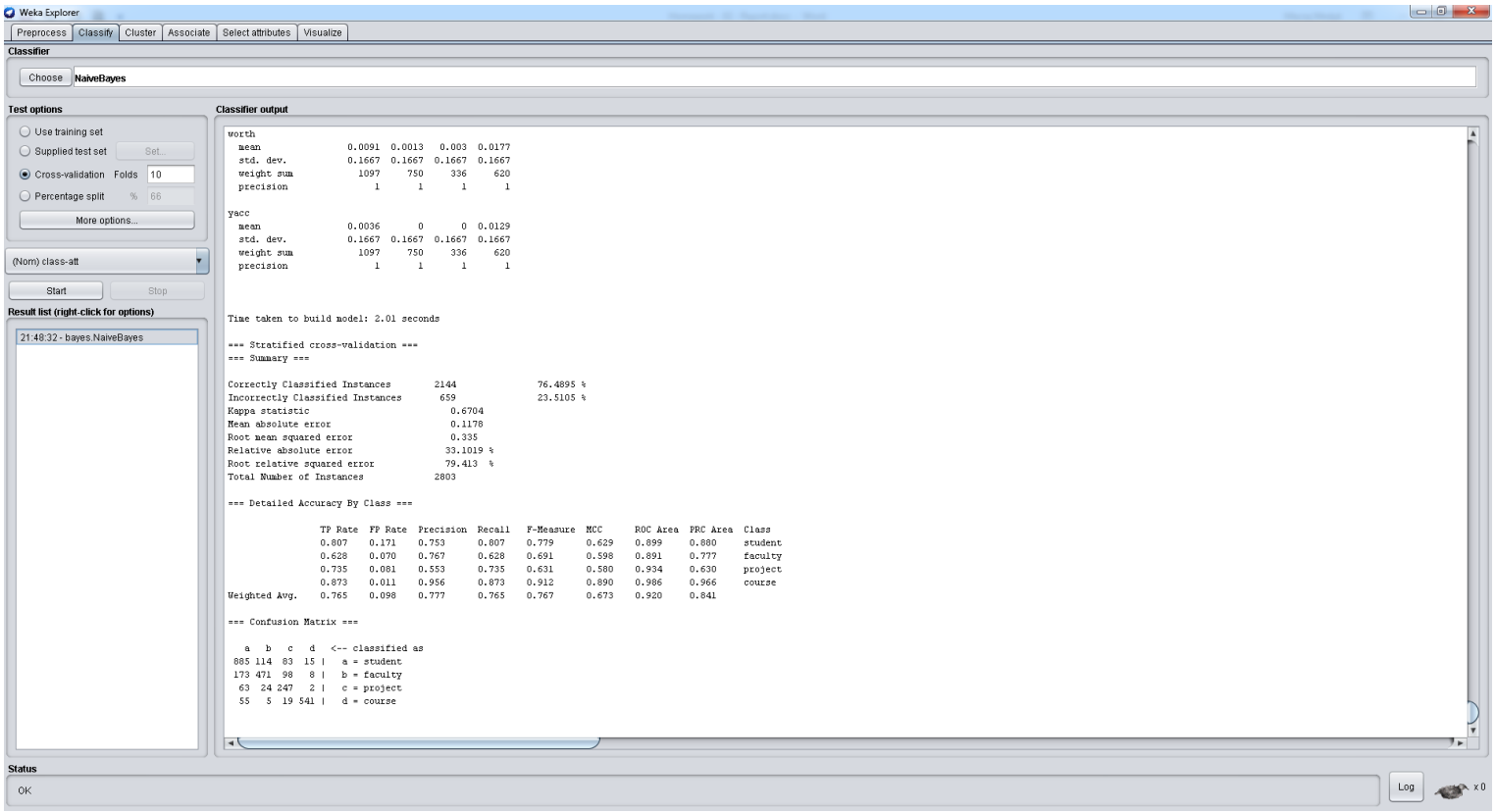


Image 08 (results of NaïveBayes classifier using 10-fold cross validation having Boolean word count)



I also run Naïve Bayes classification while supplying test set (Image 09). I was able to obtain results in Image 10 and accuracy was calculated to 77.87 % ((424 + 258 + 139 + 266) / 1396).

Image 09 (running NaïveBayes classifier using test set file webkb-test-stemmed-vector.arff having Boolean word count in both files)

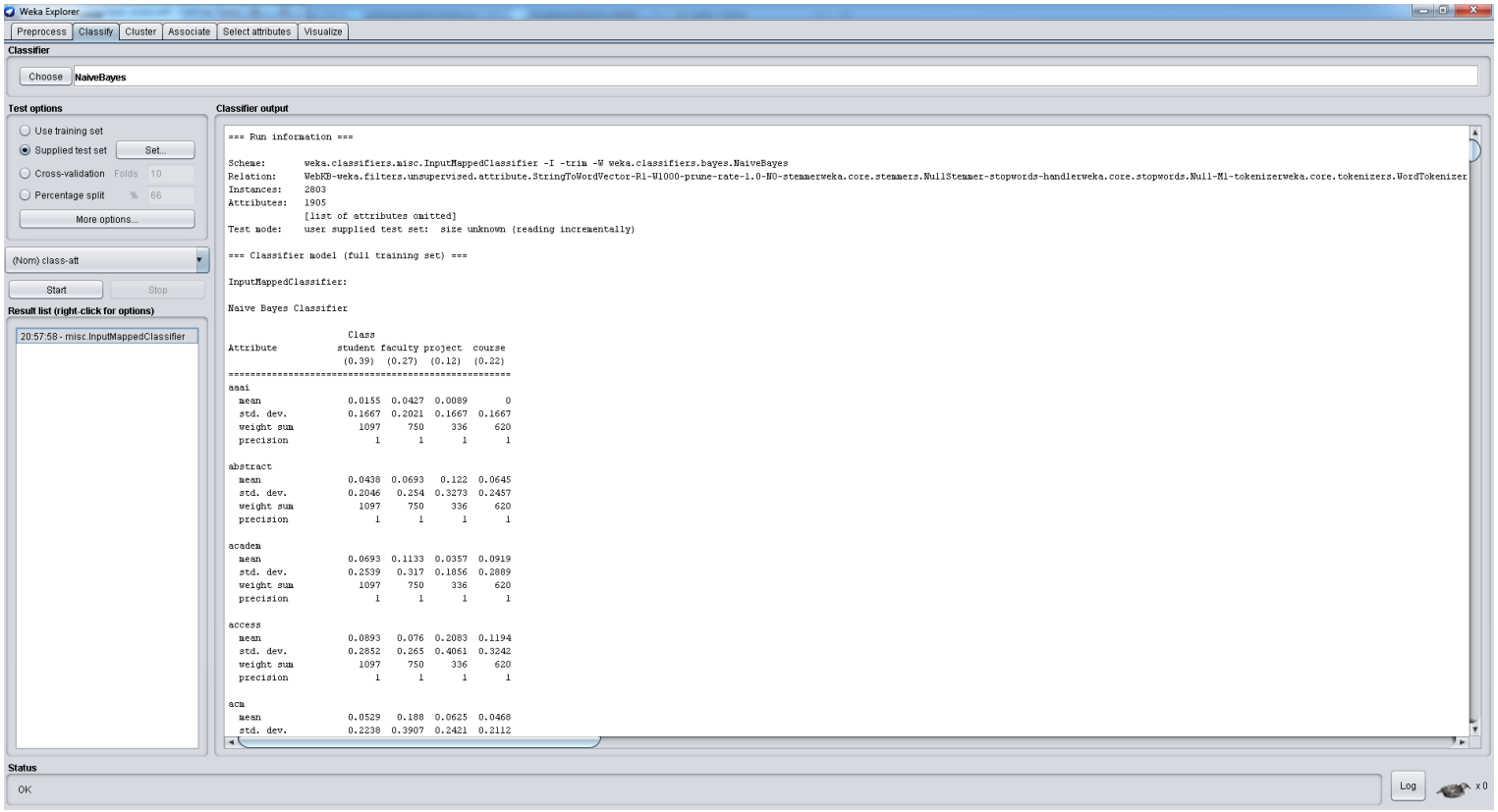
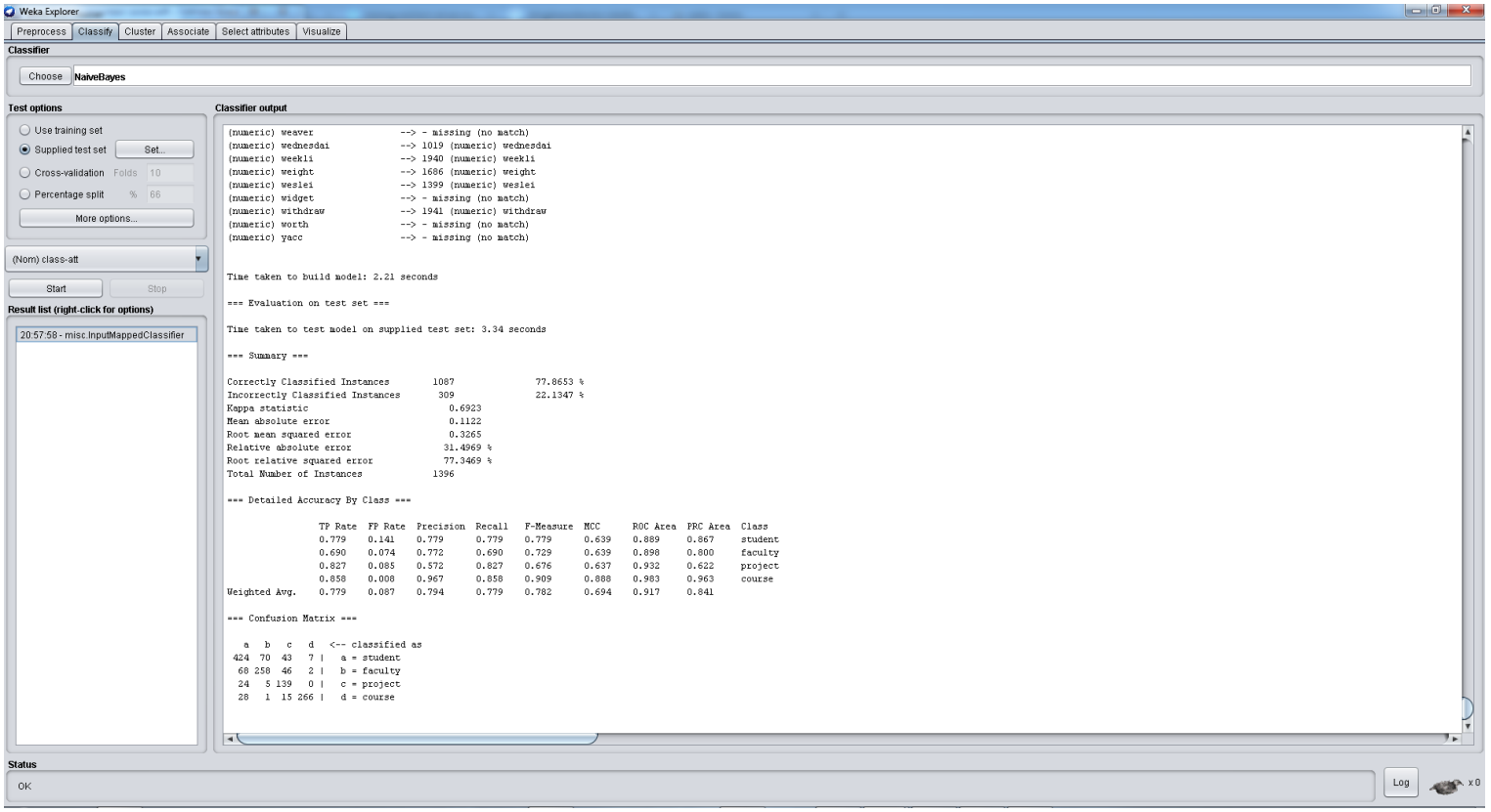


Image 10 (results of NaïveBayes classifier using test set file webkb-test-stemmed-vector.arff having Boolean word count in both files)



Afterwards, I had to install LibSVM as it was an optional package in Weka (Image 11). Then I was able to run LibSVM classification using 10-fold cross-validation (Image 12). I was able to obtain results and accuracy was calculated to 73.56 % ((1078 + 489 + 29 + 466) / 2803).

Image 11 (LibSVM package library being installed)

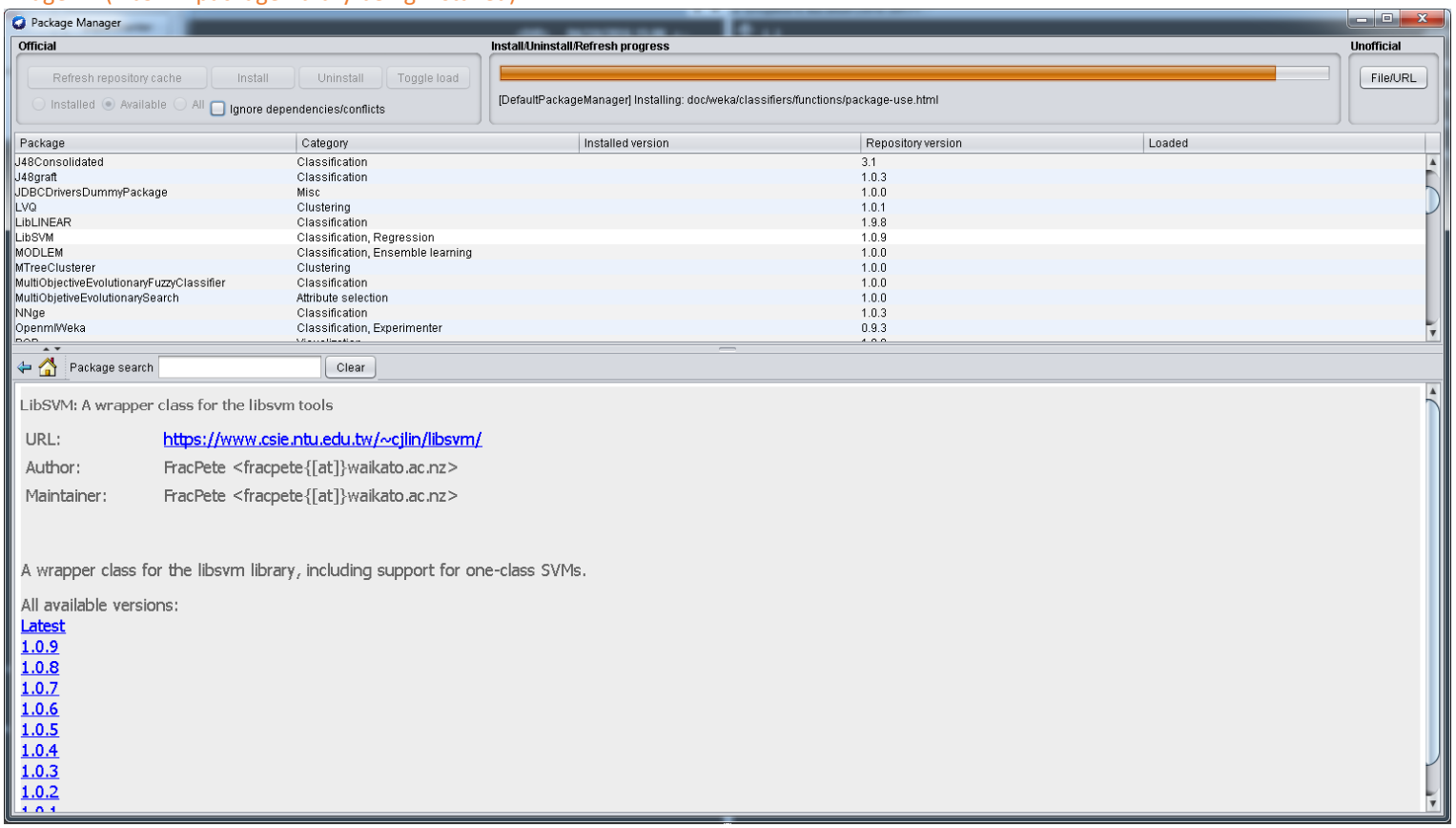
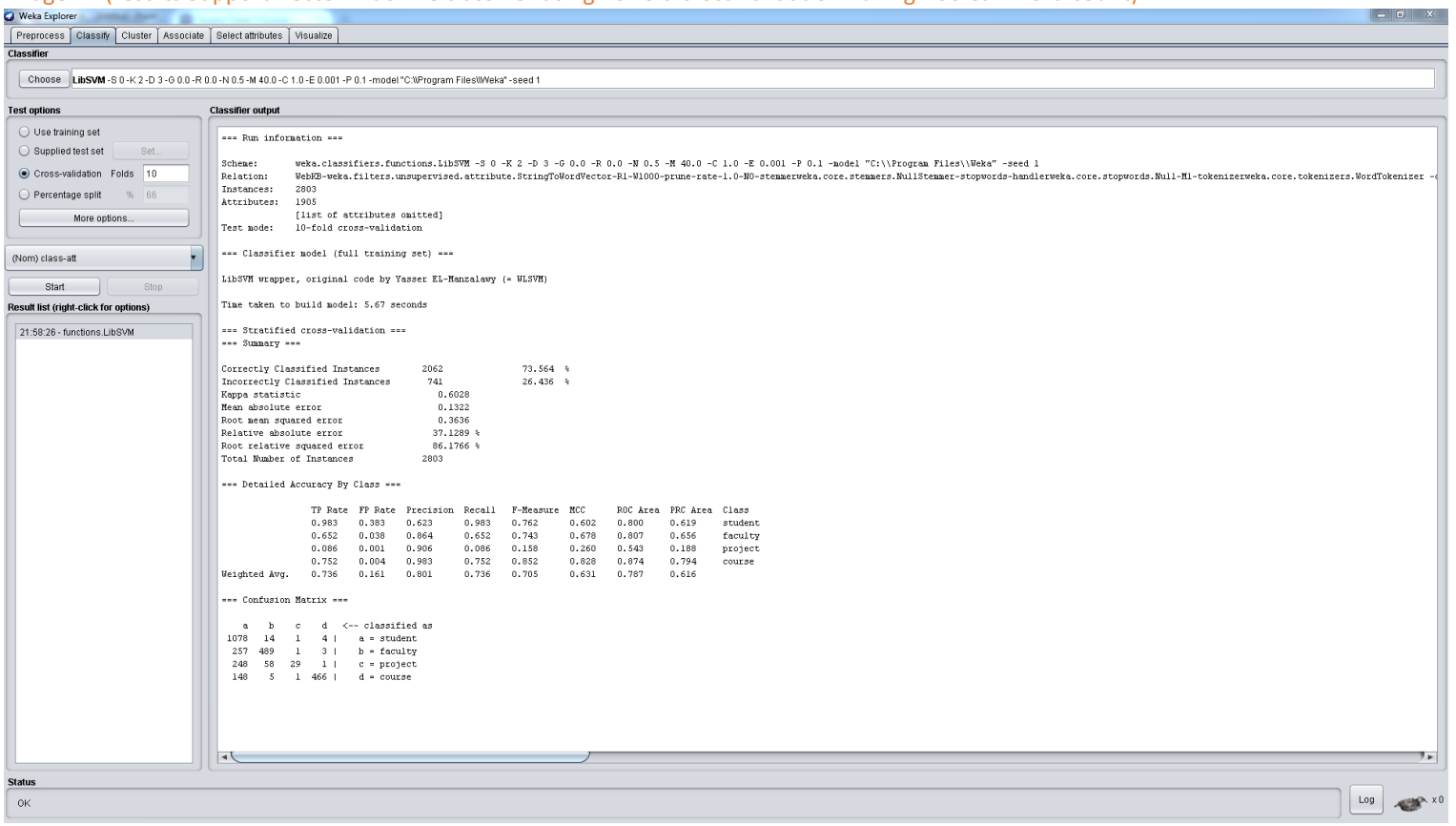


Image 12 (results Support Vector Machine classifier using 10-fold cross validation having Boolean word count)



I also run LibSVM classification while supplying test set (Image 13). I was able to obtain results in Image 14 and accuracy was calculated to 76.07 % ((526 + 279 + 26 + 231) / 1396).

Image 13 (running Support Vector Machine classifier using test set file webkb-test-stemmed-vector.arff having Boolean word count)

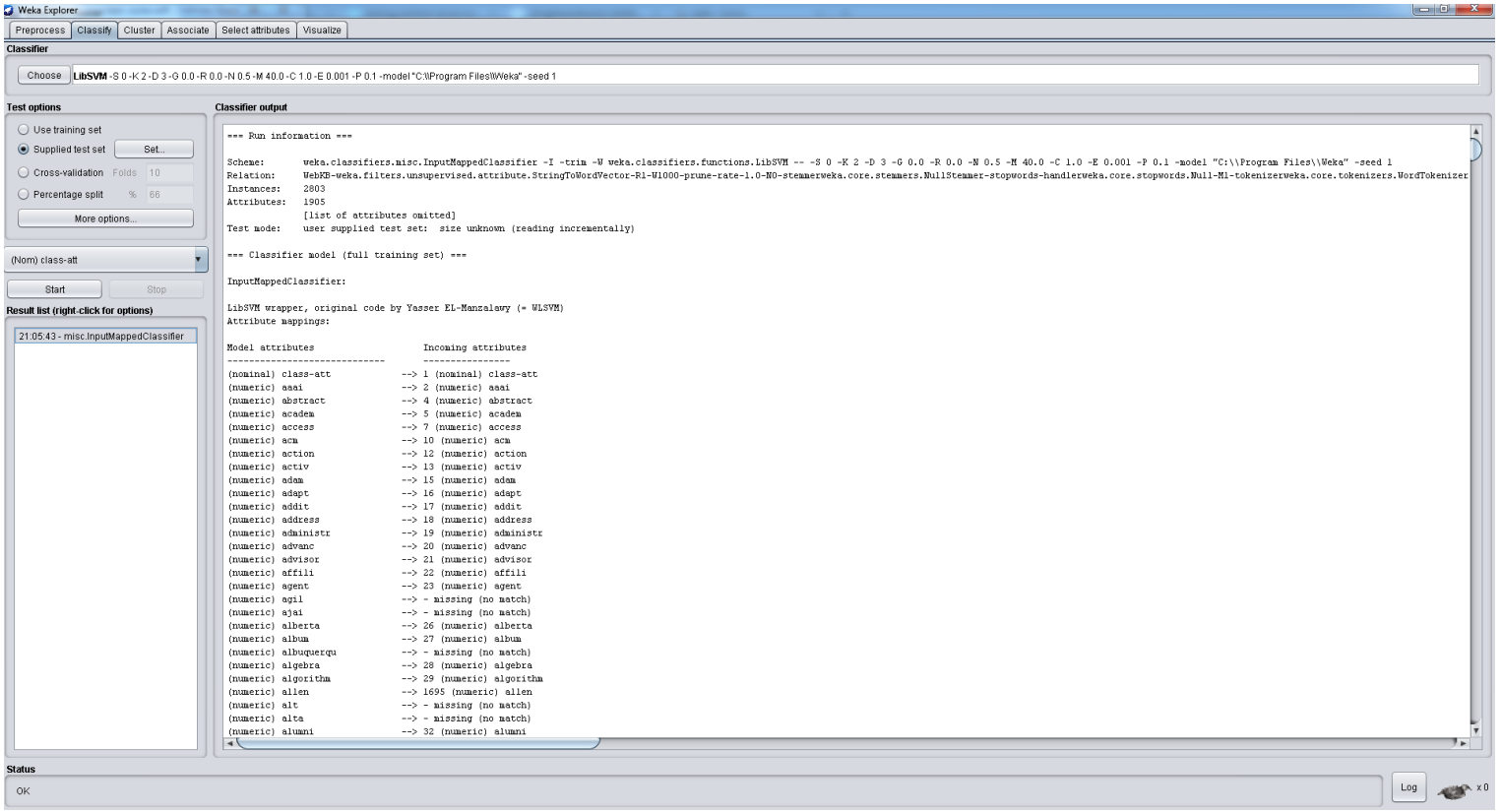
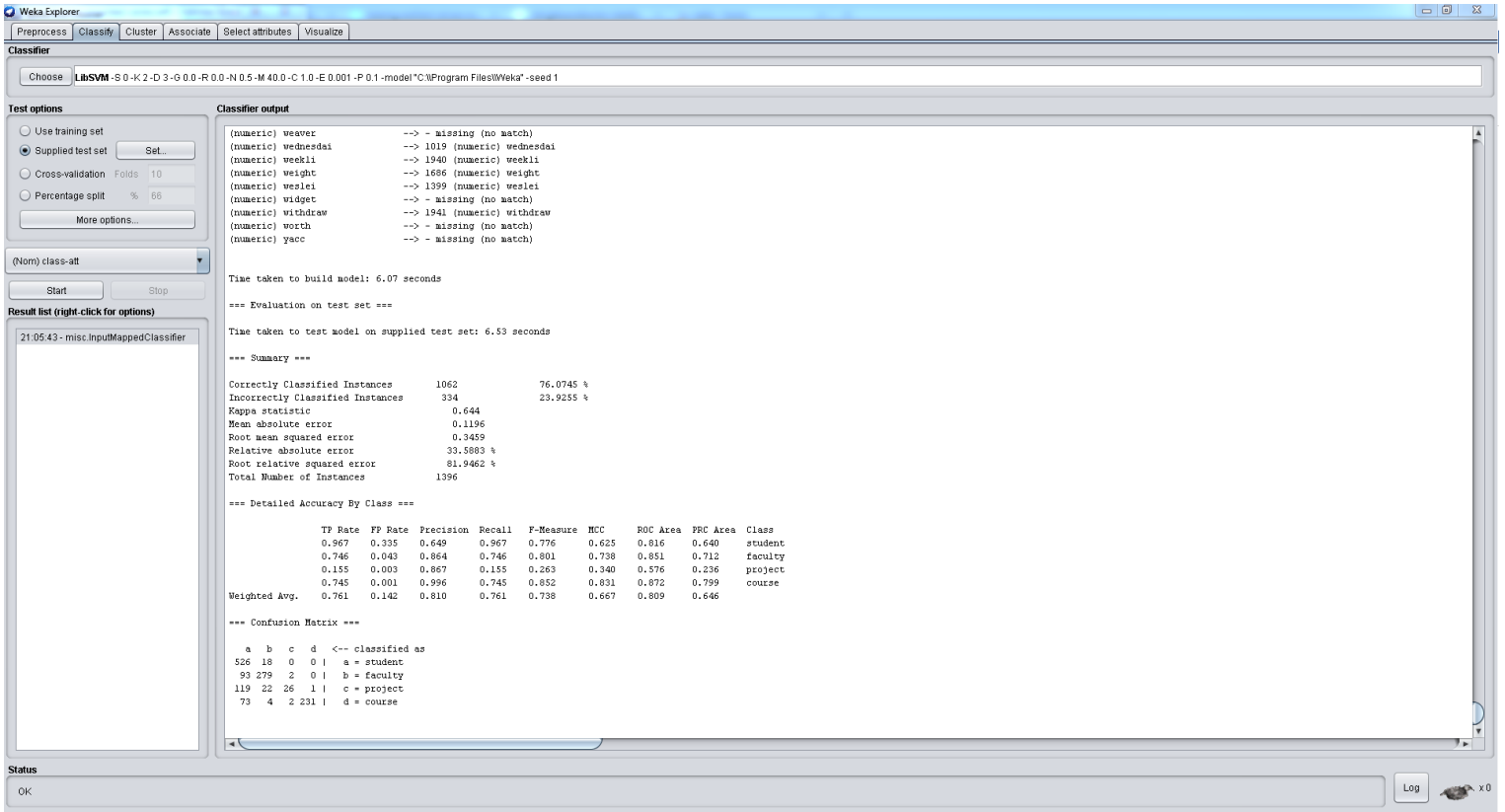


Image 14 (results of Support Vector Machine classifier using test set file webkb-test-stemmed-vector.arff having Boolean word count)



At this moment, I loaded the files that were having actual integer count. Weka StringToWordVector filter parameter “outputWordCount” was set to True on both files (Image 15). Weka also displayed actual integer word count for each file that was loaded (Image 16).

Image 15 (webkb-train-stemmed.arff loaded into Weka having actual integer word count)

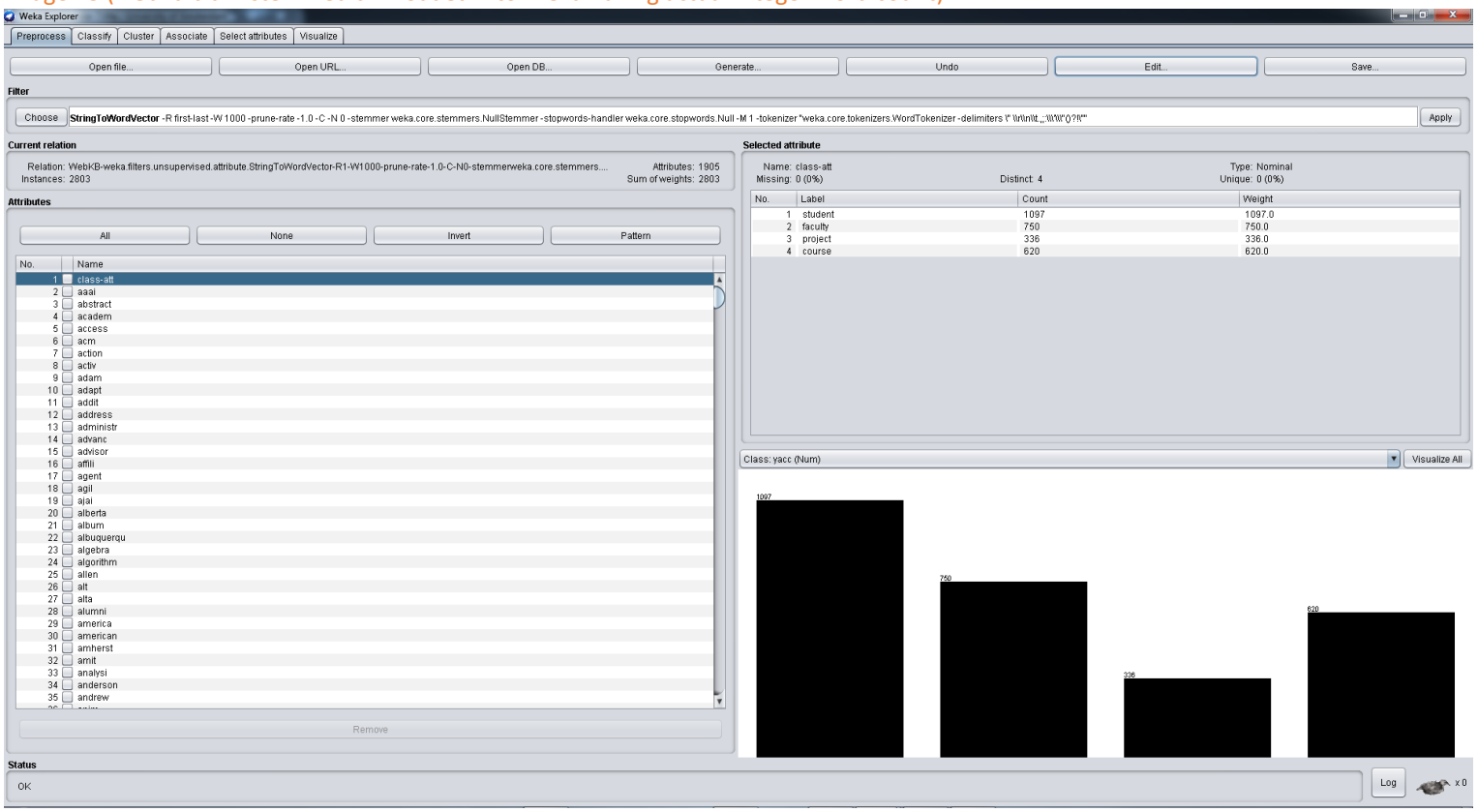
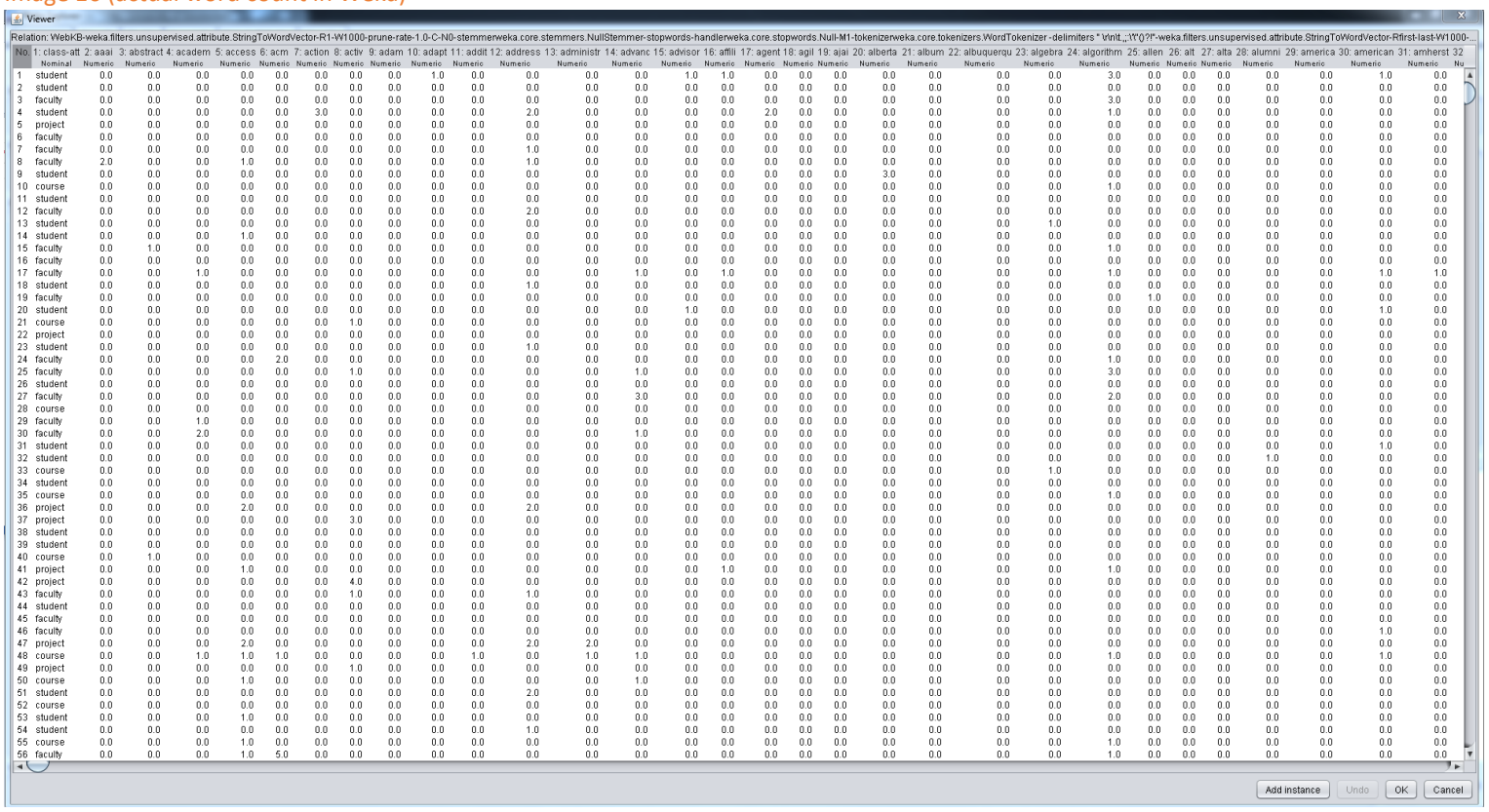


Image 16 (actual word count in Weka)



Once I had both files converted I was able to run Naïve Bayes classification using 10-fold cross-validation (Image 17). I was able to obtain results in Image 18 and accuracy was calculated to 65.39 % ((828 + 365 + 169 + 471) / 2803).

Image 17 (running NaïveBayes classifier using 10-fold cross validation having actual integer word count)

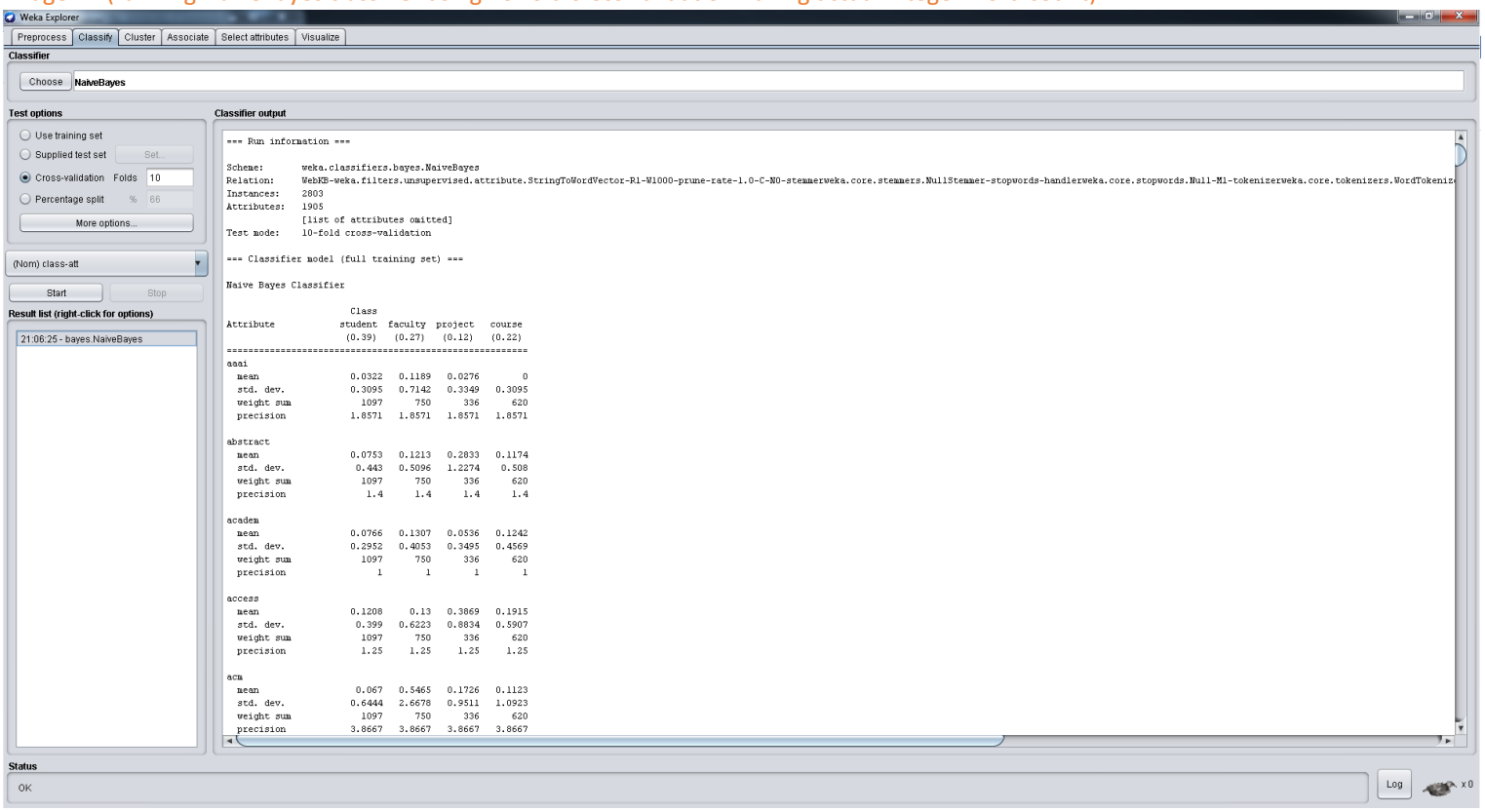
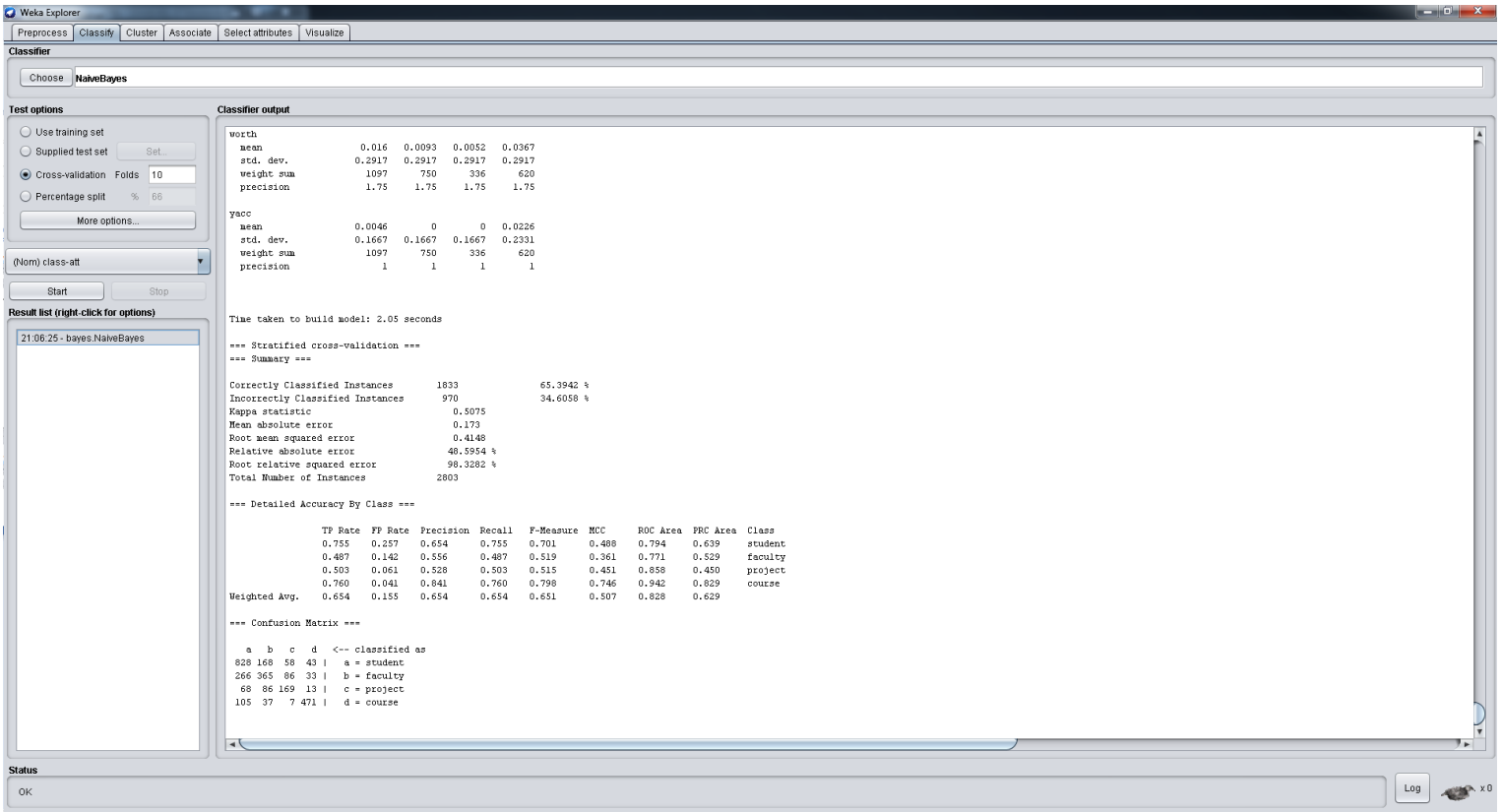


Image 18 (results of NaïveBayes classifier using 10-fold cross validation having actual integer word count)



I also run Naïve Bayes classification while supplying test set (Image 19). I was able to obtain results in Image 20 and accuracy was calculated to 63.40 % ((395 + 204 + 41 + 215) / 1396).

Image 19 (running NaïveBayes classifier using test set file webkb-test-stemmed-vector.arff having actual integer word count)

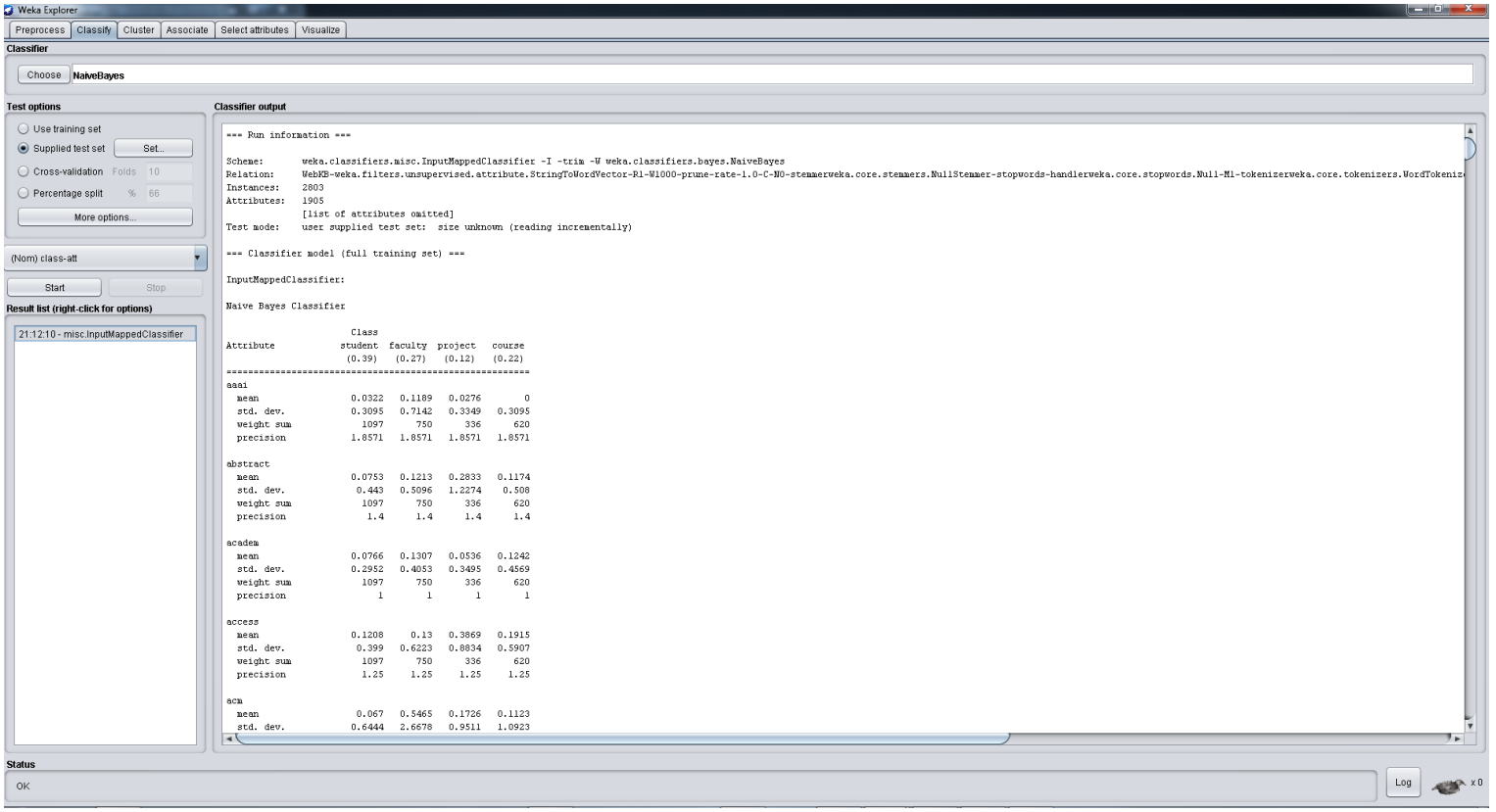
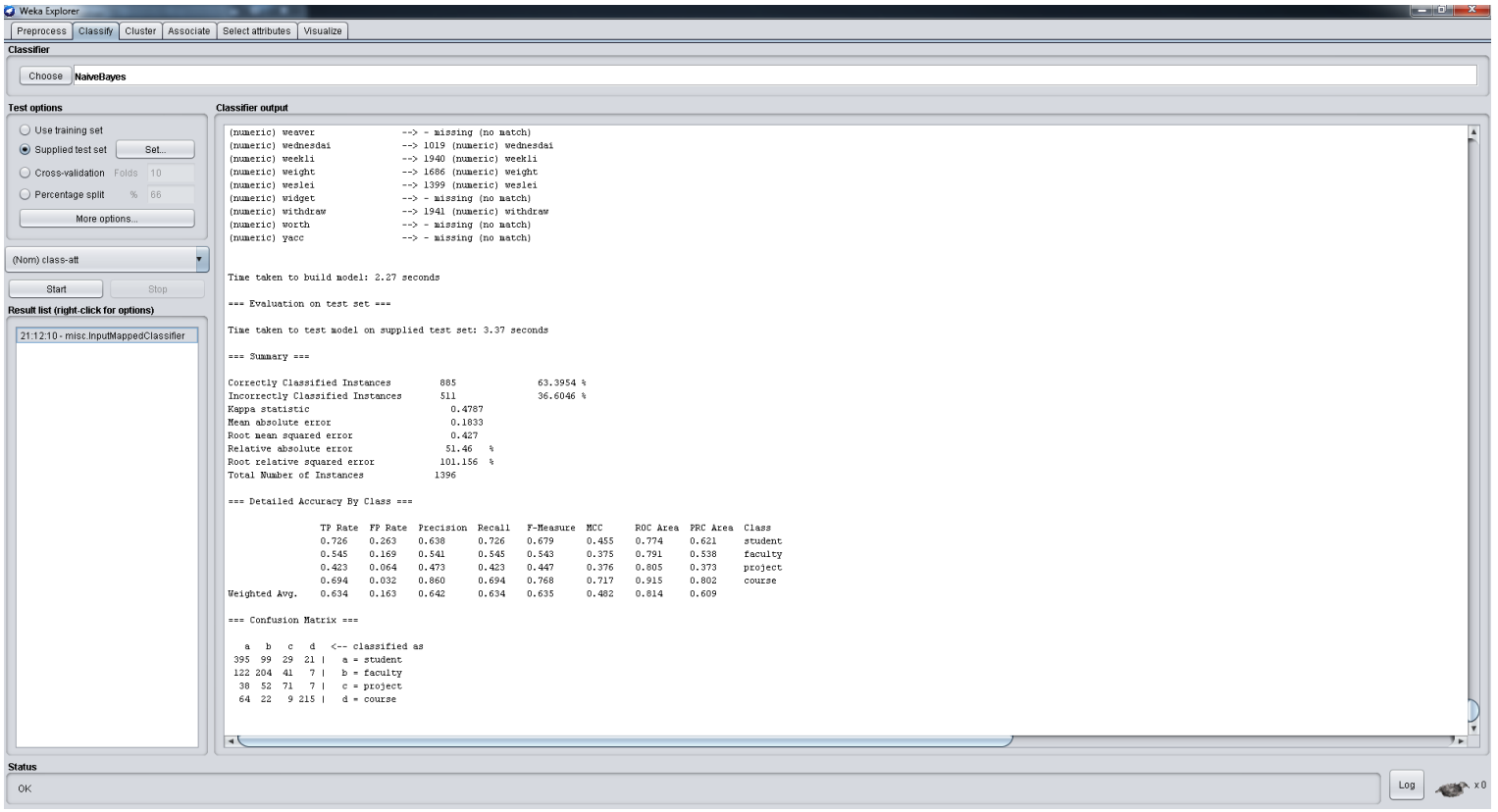
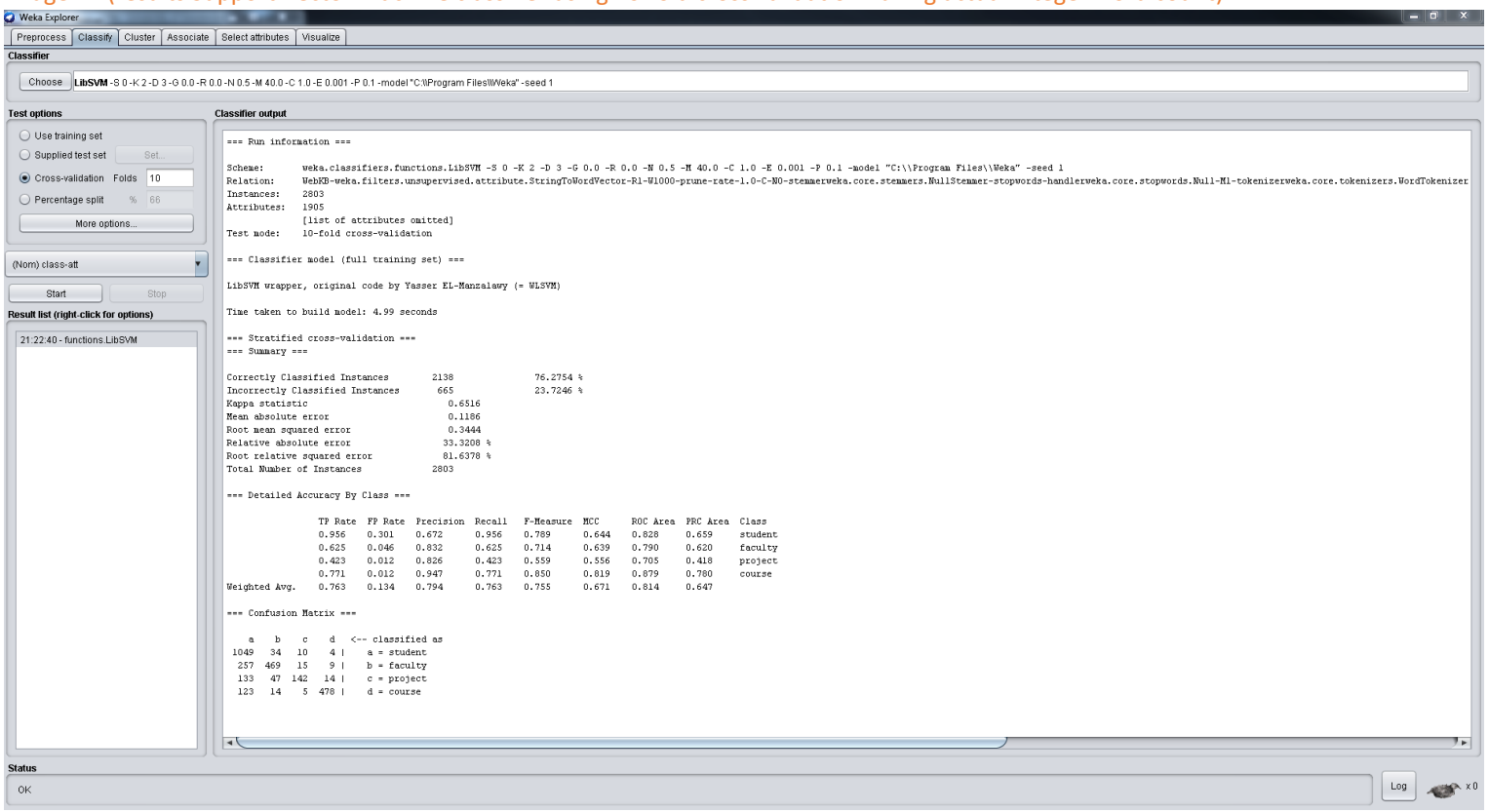


Image 20 (results of NaïveBayes classifier using test set file webkb-test-stemmed-vector.arff having actual integer word count)



Afterwards, I run LibSVM classification using 10-fold cross-validation (Image 21). I was able to obtain results and accuracy was calculated to 76.28 % ((1049 + 469 + 142 + 478) / 2803).

Image 21 (results Support Vector Machine classifier using 10-fold cross validation having actual integer word count)



I also run LibSVM classification while supplying test set (Image 22). I was able to obtain results in Image 23 and accuracy was calculated to 78.72 % ((515 + 267 + 84 + 233) / 1396).

Image 22 (running Support Vector Machine classifier using test set file webkb-test-stemmed-vector.arff having actual integer word count)

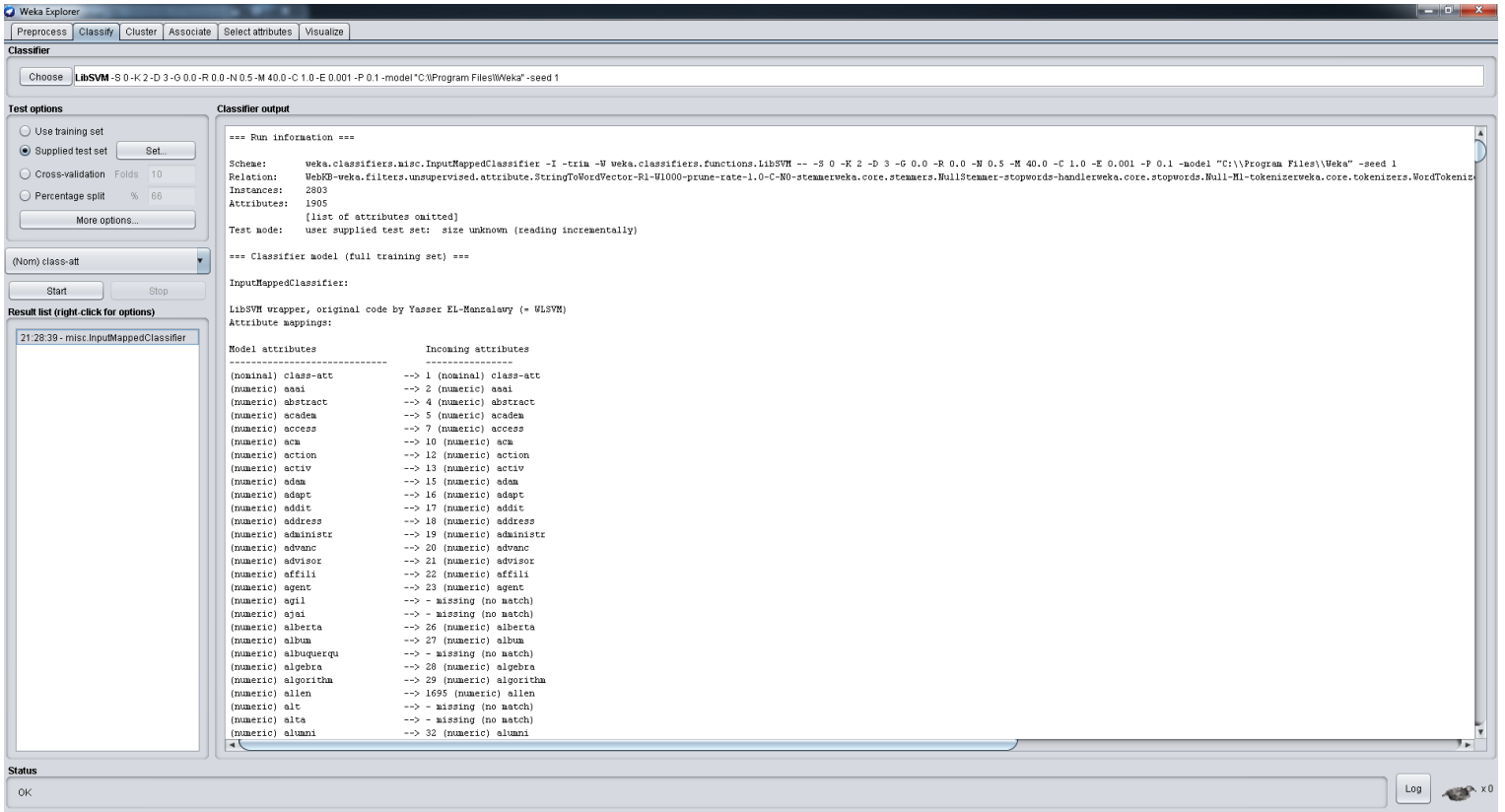
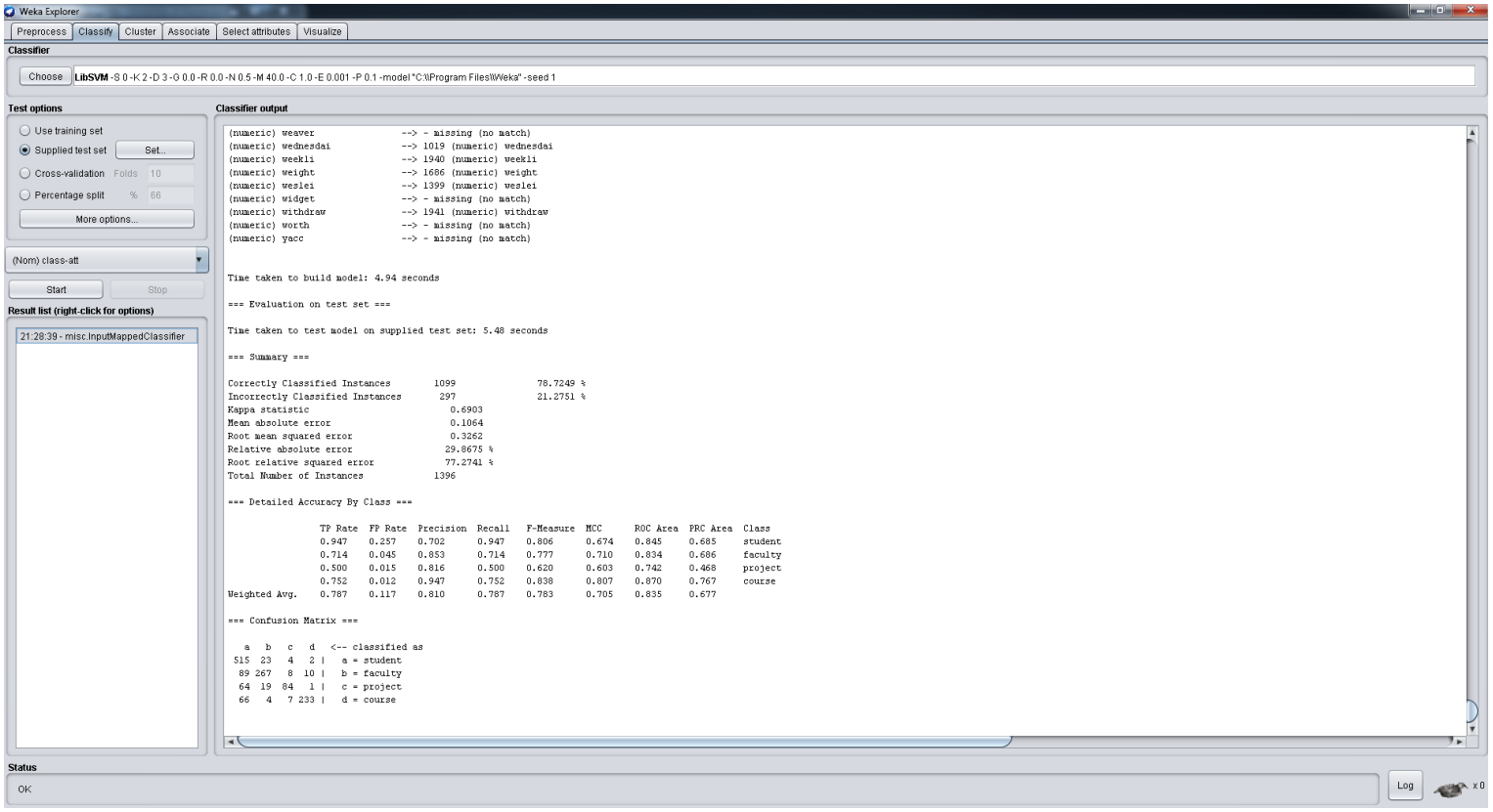


Image 23 (results of Support Vector Machine classifier using test set file webkb-test-stemmed-vector.arff having actual integer word count)



After the analysis of both Naïve Bayes and Support Vector Machine classifiers we can see that SVM while supplying test set file having actual word count was the most accurate method with accuracy of 78.28 %.

Naïve Bayes Using Test Set Having Boolean Word Count				
	A	B	C	D
A	424	70	43	7
B	68	258	46	2
C	24	5	139	0
D	28	1	15	266
Accuracy	(424 + 258 + 139 + 266) / 1396 = 77.87 %			

SVM Using Test Set Having Boolean Word Count				
	A	B	C	D
A	526	18	0	0
B	93	279	2	0
C	119	22	26	1
D	73	4	2	231
Accuracy	(526 + 279 + 26 + 231) / 1396 = 76.07 %			

Naïve Bayes Using 10-Fold Cross-Validation Having Boolean Word Count				
	A	B	C	D
A	885	114	83	15
B	173	471	98	8
C	63	24	247	2
D	55	5	19	541
Accuracy	(885 + 471 + 247 + 541) / 2803 = 76.49 %			

SVM Using 10-Fold Cross-Validation Having Boolean Word Count				
	A	B	C	D
A	1078	14	1	4
B	257	489	1	3
C	248	58	29	1
D	148	5	1	466
Accuracy	(1078 + 489 + 29 + 466) / 2803 = 73.56 %			

Naïve Bayes Using Test Set Having Actual Word Count				
	A	B	C	D
A	395	99	29	21
B	122	204	41	7
C	38	52	71	7
D	64	22	9	215
Accuracy	(395 + 204 + 71 + 215) / 1396 = 63.40 %			

SVM Using Test Set Having Boolean Word Count				
	A	B	C	D
A	515	23	4	2
B	89	267	8	10
C	64	19	84	1
D	66	4	7	233
Accuracy	(526 + 279 + 26 + 231) / 1396 = 78.72 %			

Naïve Bayes Using 10-Fold Cross-Validation Having Boolean Word Count				
	A	B	C	D
A	828	168	58	43
B	266	365	86	33
C	68	86	169	13
D	105	37	7	471
Accuracy	(828 + 365 + 169 + 471) / 2803 = 65.39 %			

SVM Using 10-Fold Cross-Validation Having Boolean Word Count				
	A	B	C	D
A	1049	34	10	4
B	257	469	15	9
C	133	47	142	14
D	123	14	5	478
Accuracy	(1049 + 469 + 142 + 478) / 2803 = 76.28 %			

In my experiments, I wanted to compare how word count would affect both classifiers. I elected to convert first batch of files to vector based files with Boolean word count and we saw that on average accuracy was around 76.00%; however, when I converted files to vector based files with actual integer word count the average dropped to 70.95%. Nevertheless, the most accurate classifier was SVM using actual integer word count tested against test file and the accuracy was calculated to 78.72%.

Files used for classification are available under this link:
https://www.dropbox.com/s/yw6guvubqrxcgec/Homework%20-%2002%20-%20Datasets.zip?dl=0

Program code solution is available under this link:
https://www.dropbox.com/s/pnv0ngppdyrb1u6/Homework%20-%2002%20-%20Code.zip?dl=0

Addendum – Program used for conversion from txt format to arff format using C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.IO;
using System.Text;
using System.Threading.Tasks;

namespace CAP6776_HW2
{
    class Program
    {
        private static List<List<string>> tokens = new List<List<string>>();
        private static List<string> sentences = new List<string>();

        static void Main(string[] args)
        {
            CreateFilesARFF("webkb-train-stemmed.txt", "Train");
            CreateFilesARFF("webkb-test-stemmed.txt", "Test");
        }

        public static void CreateFilesARFF(string filename, string filetype)
        {
            List<string> sentences = ReadFile(filename);
            sentences = ConvertFile(sentences, filetype);
            WriteFile(filename.Substring(0, filename.Length - 4) + ".arff", sentences);
        }

        public static List<string> ReadFile(string filename)
        {
            List<string> result = new List<string>();
            var lines = File.ReadLines(filename);
            foreach (var line in lines)
            {

                result.Add(line);
            }
            return result;
        }

        public static void WriteFile(string filename, List<string> sentences)
        {
            File.WriteAllLines(filename,sentences);
        }

        public static List<string> ConvertFile(List<string> sentences, string type)
        {
            List<string> result = new List<string>();
            result.Add("@relation 'WebKB " + type);
            result.Add("");
            result.Add("@attribute Text string");
            result.Add("@attribute class-att {student, faculty, project course}");
            result.Add("");
```

```
            result.Add("@data");
            result.Add("");
            foreach (var line in sentences)
            {
                int index = Search(line);
                string bodyClass = line.Substring(0, index);
                string bodyText = "'" + line.Substring(index + 1, line.Length - index - 1) + "'";
                result.Add(bodyText + "," + bodyClass);
            }
            return result;
        }

        public static int Search(string line)
        {
            for (int i = 0; i < line.Length; i++)
            {
                if (line[i] == '\t') return i;
                {

                }
            }
            return -1;
        }
    }
}
```