Zeno Barcutean
barcute2
CS 440
Homework 6

Problem 1

1.1

```
Newsgroup 0
_ 0.5365853658536586
_ 0.4888888888888889
_ 0.27983539094650206
_ 0.2319277108433735
_ 0.13968957871396895

Newsgroup 1
_ 0.3645833333333333
_ 0.3588709677419355
_ 0.32894736842105265
_ 0.288
_ 0.2792452830188679

Newsgroup 2
_ 0.4928909952606635
_ 0.44015444015444016
_ 0.3623693379790941
_ 0.3090128755364807
_ 0.2508361204013378

Newsgroup 3
_ 0.44680851063829785
_ 0.41005291005291006
_ 0.37399770904925544
_ 0.34375
_ 0.31216931216931215

Newsgroup 4
_ 0.5357142857142857
_ 0.3651877133105802
_ 0.33905579399141633
_ 0.3333333333333333
_ 0.3182751540041068

Newsgroup 5
_ 0.5220338983050847
_ 0.48
_ 0.45454545454545453
_ 0.41964285714285715
_ 0.40236686390532544
```

1.2

```
Newsgroup 0
the 1.0
i 0.9841269841269841
```

```
space 0.96875
and 0.96875
the 0.9506172839506173

Newsgroup 1
the 1.0
the 0.9523809523809523
a 0.9473684210526315
scsi2 0.9473684210526315
the 0.9444444444444444

Newsgroup 2
the 1.0
that 0.9666666666666667
1 0.95
04 0.9444444444444444
_ 0.9444444444444444

Newsgroup 3
the 1.0
keycode 0.9545454545454546
n 0.95
keymap 0.95
to 0.9444444444444444

Newsgroup 4
the 1.0
_ 0.9850746268656716
to 0.96875
to 0.9615384615384616
to 0.9615384615384616

Newsgroup 5
10 1.0
1 0.9655172413793104
it 0.95
1 0.9473684210526315
1 0.9473684210526315
```

1.3 The reason we normalize is that the weights of all the words within the documents are comparable. This way, there are no big discrepancies between long and short documents, or documents that use a lot of vocabulary words and those who use less words.
I think the first method is better at providing information in the case of long documents, because it would produce less “spikes” in the weight of the words. However, in extreme cases the weights would become very small and it would cause problems for the programmer, who would have to use different data types to handle the data.
The second method gives us the frequency relative to the term that occurs the most. Normalizing this way could give bad results if the most popular term is not one that yields significant information (like a conjunction).

1.4 Taking the log of tf might be useful when dealing a large range of weights. This way some terms which occur very frequently will not have a great importance in our classification.

1.5 They are not too helpful because they do not provide any meaningful information. Most of them are articles, conjunctions and empty spaces (especially for the first formula) which occur frequently within a language but do not convey much information.
1.6 the, a, an, and, or, one, to, from, _, this, that, by
I chose them because the articles, adverbs, conjunctions and prepositions are very frequent in every language, yet they are there for syntactic reasons, rather than have an important role in conveying information.

Problem 2
2.1
```
_sunview 6.2314652154886145
loadup 5.82600010738045
gruber 5.538318034928669
champions 5.315174483614459
relies 5.1328529268205045
analyze 4.9787022469932465
compsys 4.845170854368724
assistance 4.727387818712341
relief 4.622027303054514
fewer 4.526717123250189
```

2.2 They are not the same, as it was expected. This formula counts the inverse term frequency, so the words with a higher value were the least frequent. To get the most frequent words, we should have looked for the lowest values.

2.3
```
Newsgroup 0
the 116.0
_ 104.63414634146342
the 96.1696113074205
_ 95.33333333333333
the 94.15384615384616

Newsgroup 1
the 115.0939226519337
the 87.9463087248322
the 87.23076923076923
the 86.74698795180723
the 84.9787525702536

Newsgroup 2
the 115.72205438066466
the 105.94594594594595
the 99.50214592274678
the 99.29552238805971
_ 96.11374407582937

Newsgroup 3
the 97.54838709677419
the 93.62229102167183
_ 87.12765957446808
_ 79.96031746031746
the 78.44357976653697
```

```
Newsgroup 4
_ 104.46428571428571
the 103.6595744680851
the 94.5
the 89.46745562130177
the 88.81967213114754

Newsgroup 5
_ 101.79661016949153
_ 93.6
_ 88.63636363636364
_ 81.83035714285714
_ 78.46153846153847
```

2.4 The parameter p can work as a scaling factor and for input validation purposes: for example, assigning p as positive value can make sure the evaluation expression will not make a division by 0 (it shouldn't do that anyway for our purposes, but it could if/when re-using the code). I think that assigning p negative values is not recommended, because it could nullify the number of documents the term has showed up in and then we would get an error because of division by 0. As for other positive values, it could be just for scale, as long as we keep its value consistent for all terms.

2.5 From problem 1, it looks like the limitations of the term frequency include the fact that many words which have little significance for us (some of the stop-words I listed are the best examples) have a higher weight. When trying to classify a document, these words would interfere with the classification. Looking at the results of tf-idf, it looks like it didn't perform that much better. If anything, at least there aren't that many blank spaces showing up in the results, but the need for stop words is obvious.

2.6 Even though tf-idf seems to focus more on the significant terms, rather than the most frequent terms, sometimes a high-frequency can “over-power” the results. We can see that in newsgroup 5, where all the top results are made up of blank spaces, so it can behave similarly to tf if the values of the term frequency are high enough.

Problem 3

3.1 I have used a J48 tree with pruning enabled. See Problem1.java and the readme for the list of stop words and the formulas used. The accuracy on the testing set 78% and it is better than what I obtained using a multi-layer perceptron, even with cross-validation (not to mention it runs faster).

3.2 See README and barcute2_application_result.txt