COM3004 Assignment 2 Introductory Report

Approach

Overall the approach was broadly similar to the one suggested in the assignment spec. Letter sub-images are extracted from the main image and classified and then a search is performed for each word. The results are then displayed in a window with the unannotated and annotated images side by side.

I decided to use a more object-orientated approach than was typically used in the lab sheets, with the intention of producing cleanly separated code that was Pythonic in nature at first, and then optimising for speed later. (In the words of Knuth, "[A]ttempts at efficiency actually have a strong negative impact when debugging and maintenance are considered. [...] [P]remature optimization is the root of all evil."[1])

Regarding the wordsearch solving algorithm, it would initially generate a letter grid with the target word in a hypothesised position and leave all the other letters blank. Then it would compare this hypothesis grid with the letter labels generated by the classifier to give a score for the hypothesised position. A letter that matched in the hypothesis grid and the classified wordsearch would add a value of 1 to the score, and a mismatch would add 0. Thus, the algorithm had a tolerance for the occasional misclassified letter.

But I decided to try and do one better than this and allow for more graduated scoring than simply 0 or 1 -- surely a mismatch between an "H" and an "N" should be penalised less than between, say, a "J" and an "F"? So when the classifier is trained with the training data (in the case of our *k*-nearest-neighbour classifier, this just means populating an object attribute), a leave-one-out test is performed and a confusion matrix calculated. This is then used to generate a probability distribution of a letter's true class label given the label the classifier guessed for it, in what I have termed a "normalised" confusion matrix. Now when scoring an individual letter, its probability of occurrence given the image data is used rather than a simple "best match" classification.

Additionally, when trying to build a good feature selection algorithm I ran into the problem that in the notes the "divergence" measure wasn't easily generalisable to >2 classes. Therefore after some research I settled on using the ANOVA f-value as a suitable measure of divergence for the "Best K Features" selector. Rather than implementing this myself, I chose to use the implementation from `scipy.stats`.

Code structure and quality

The code is structured in submodules of a package called "`assignment`", which all have explanatory docstrings on their first line. Hopefully the docstrings for the modules, classes,

[1]Structured Programming with Go To Statements, Knuth, D. E., Computing Surveys, Vol 6, No 4, December 1974

methods and functions explain their purpose. Info and debug log output is available using the Python `logging` library, and the code also conforms to PEP8 and PEP257 standards.

Entry point

The main “`wordsearch`” function asked for in the assignment is defined at `assignment.wordsearch`.

Profiling

As outlined above, my approach was to write Pythonic code first and optimise for speed afterwards. To do this I used the `cProfile` and `pstats` modules to incrementally profile and make optimisations, which mostly consisted of converting for-loops to numpy array operations. Included in the code submission are the text outputs from these incremental profiling steps, the final one of which is in the file “`profiling/final_1.txt`”. As can be seen, the heaviest few operations in the code are all now numpy array operations, and as such can’t be optimised further.

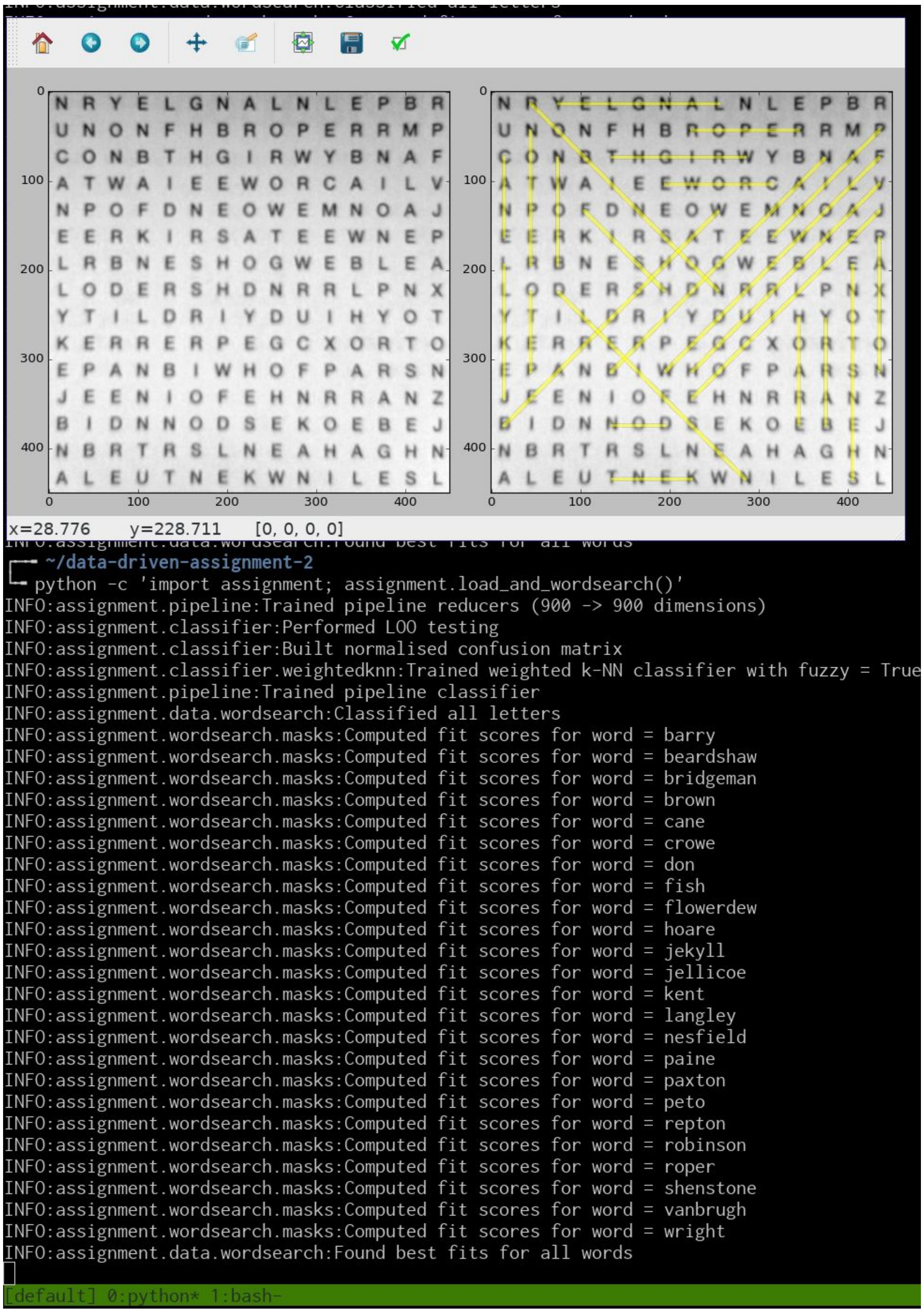
Tuning

The selection of which dimensionality reduction techniques to use (and their parameters) required some tuning steps. I failed to keep a detailed track of exactly what these incremental steps were, but eventually settled on:

- Remove the lower 4 pixels and the rightmost 3 pixels from each letter image
- Reduce the resulting 702-dimension feature vector to 11 dimensions using PCA.
- Drop the first of these features, leaving a 10-dimension feature vector.
- Use a fuzzy 1-nearest-neighbour classifier on these 10-dimension feature vectors to produce a probabilistic class label vector.

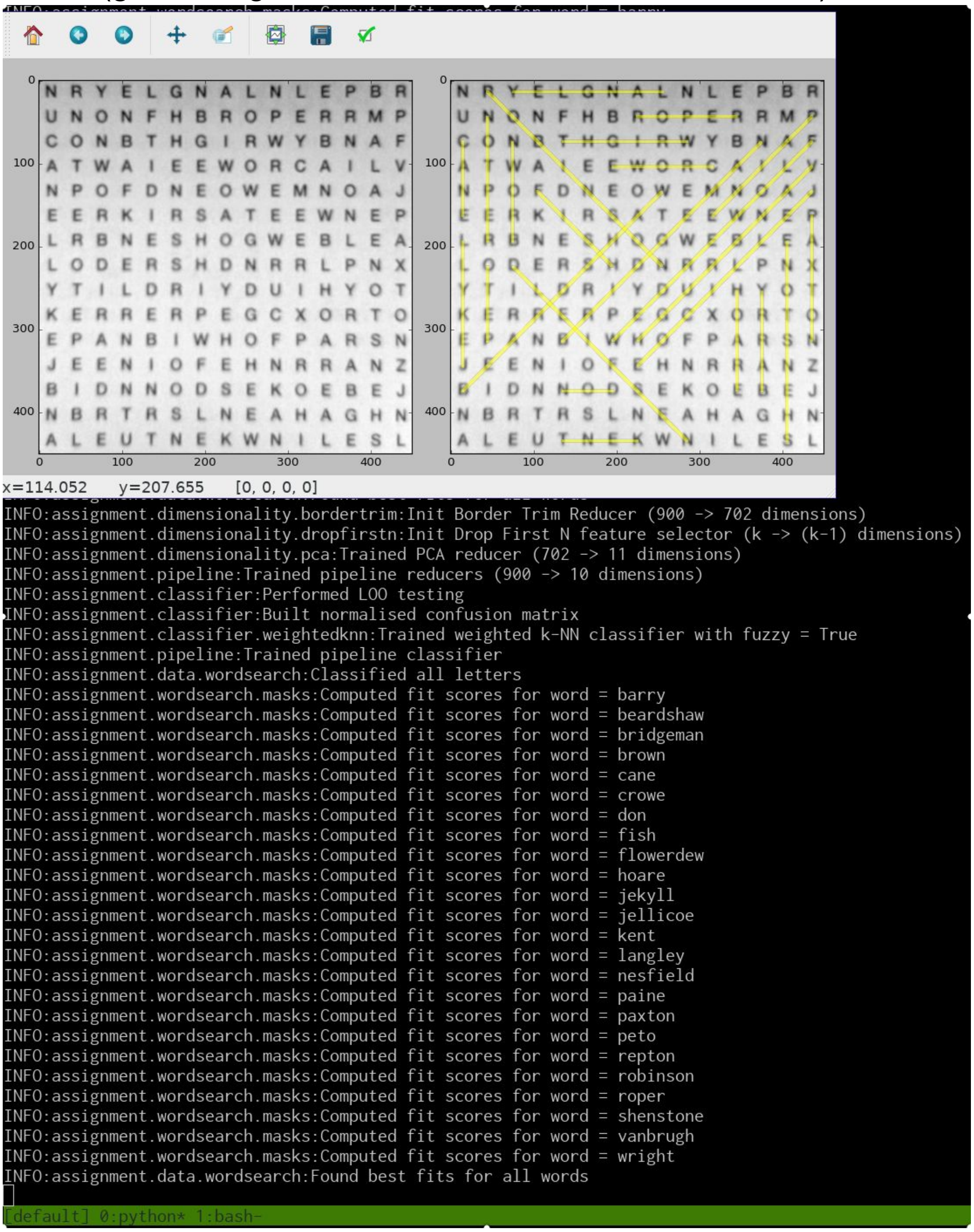
This was able to score a perfect 24 words found on the *test1* image, and 22 out of 24 on the *test2* image. Notably, our simple “drop the first feature” selection algorithm performed better than the more complex “select the best k features according to their ANOVA f-values” selection algorithm. I initially thought this might have been to do with the ANOVA f-value not taking into account the correlation between features, but this doesn’t make much sense as the features it was operating on were in fact the orthogonal “eigenletters”, rather than individual pixels. This is possibly a case of overfitting, and the ANOVA f-value feature selector might perform better in the unseen test data.

Test 1 (good image, no dimensionality reduction, score: 24)



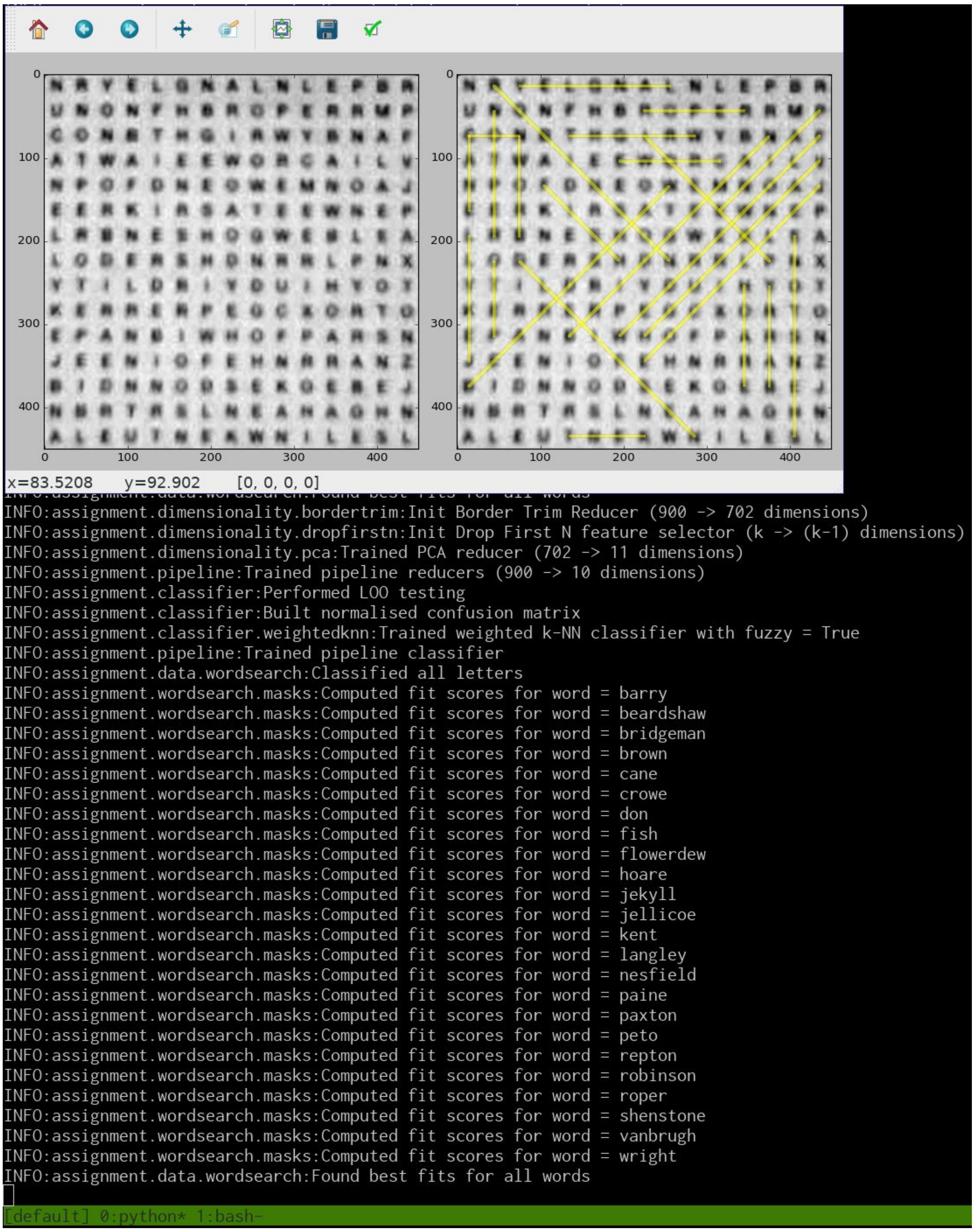
```
x=28.776    y=228.711    [0, 0, 0, 0]
INFO:assignment.data.wordsearch:Found best fits for all words
┌── ~/data-driven-assignment-2
└─ python -c 'import assignment; assignment.load_and_wordsearch()'
INFO:assignment.pipeline:Trained pipeline reducers (900 -> 900 dimensions)
INFO:assignment.classifier:Performed LOO testing
INFO:assignment.classifier:Built normalised confusion matrix
INFO:assignment.classifier.weightedknn:Trained weighted k-NN classifier with fuzzy = True
INFO:assignment.pipeline:Trained pipeline classifier
INFO:assignment.data.wordsearch:Classified all letters
INFO:assignment.wordsearch.masks:Computed fit scores for word = barry
INFO:assignment.wordsearch.masks:Computed fit scores for word = beardshaw
INFO:assignment.wordsearch.masks:Computed fit scores for word = bridgeman
INFO:assignment.wordsearch.masks:Computed fit scores for word = brown
INFO:assignment.wordsearch.masks:Computed fit scores for word = cane
INFO:assignment.wordsearch.masks:Computed fit scores for word = crowe
INFO:assignment.wordsearch.masks:Computed fit scores for word = don
INFO:assignment.wordsearch.masks:Computed fit scores for word = fish
INFO:assignment.wordsearch.masks:Computed fit scores for word = flowerdew
INFO:assignment.wordsearch.masks:Computed fit scores for word = hoare
INFO:assignment.wordsearch.masks:Computed fit scores for word = jekyll
INFO:assignment.wordsearch.masks:Computed fit scores for word = jellicoe
INFO:assignment.wordsearch.masks:Computed fit scores for word = kent
INFO:assignment.wordsearch.masks:Computed fit scores for word = langley
INFO:assignment.wordsearch.masks:Computed fit scores for word = nesfield
INFO:assignment.wordsearch.masks:Computed fit scores for word = paine
INFO:assignment.wordsearch.masks:Computed fit scores for word = paxton
INFO:assignment.wordsearch.masks:Computed fit scores for word = peto
INFO:assignment.wordsearch.masks:Computed fit scores for word = repton
INFO:assignment.wordsearch.masks:Computed fit scores for word = robinson
INFO:assignment.wordsearch.masks:Computed fit scores for word = roper
INFO:assignment.wordsearch.masks:Computed fit scores for word = shenstone
INFO:assignment.wordsearch.masks:Computed fit scores for word = vanbrugh
INFO:assignment.wordsearch.masks:Computed fit scores for word = wright
INFO:assignment.data.wordsearch:Found best fits for all words

[default] 0:python* 1:bash-
```

Test 2 (good image, 900 → 10 dim. reduction, score: 24)


x=114.052   y=207.655   [0, 0, 0, 0]
[default] 0:python* 1:bash-

```
INFO:assignment.dimensionality.bordertrim:Init Border Trim Reducer (900 -> 702 dimensions)
INFO:assignment.dimensionality.dropfirstn:Init Drop First N feature selector (k -> (k-1) dimensions)
INFO:assignment.dimensionality.pca:Trained PCA reducer (702 -> 11 dimensions)
INFO:assignment.pipeline:Trained pipeline reducers (900 -> 10 dimensions)
INFO:assignment.classifier:Performed LOO testing
INFO:assignment.classifier:Built normalised confusion matrix
INFO:assignment.classifier.weightedknn:Trained weighted k-NN classifier with fuzzy = True
INFO:assignment.pipeline:Trained pipeline classifier
INFO:assignment.data.wordsearch:Classified all letters
INFO:assignment.wordsearch.masks:Computed fit scores for word = barry
INFO:assignment.wordsearch.masks:Computed fit scores for word = beardshaw
INFO:assignment.wordsearch.masks:Computed fit scores for word = bridgeman
INFO:assignment.wordsearch.masks:Computed fit scores for word = brown
INFO:assignment.wordsearch.masks:Computed fit scores for word = cane
INFO:assignment.wordsearch.masks:Computed fit scores for word = crowe
INFO:assignment.wordsearch.masks:Computed fit scores for word = don
INFO:assignment.wordsearch.masks:Computed fit scores for word = fish
INFO:assignment.wordsearch.masks:Computed fit scores for word = flowerdew
INFO:assignment.wordsearch.masks:Computed fit scores for word = hoare
INFO:assignment.wordsearch.masks:Computed fit scores for word = jekyll
INFO:assignment.wordsearch.masks:Computed fit scores for word = jellicoe
INFO:assignment.wordsearch.masks:Computed fit scores for word = kent
INFO:assignment.wordsearch.masks:Computed fit scores for word = langley
INFO:assignment.wordsearch.masks:Computed fit scores for word = nesfield
INFO:assignment.wordsearch.masks:Computed fit scores for word = paine
INFO:assignment.wordsearch.masks:Computed fit scores for word = paxton
INFO:assignment.wordsearch.masks:Computed fit scores for word = peto
INFO:assignment.wordsearch.masks:Computed fit scores for word = repton
INFO:assignment.wordsearch.masks:Computed fit scores for word = robinson
INFO:assignment.wordsearch.masks:Computed fit scores for word = roper
INFO:assignment.wordsearch.masks:Computed fit scores for word = shenstone
INFO:assignment.wordsearch.masks:Computed fit scores for word = vanbrugh
INFO:assignment.wordsearch.masks:Computed fit scores for word = wright
INFO:assignment.data.wordsearch:Found best fits for all words
```

Test 3 (bad image, 900 → 10 dim. reduction, score: 22)



```
INFO:assignment.dimensionality.bordertrim:Init Border Trim Reducer (900 -> 702 dimensions)
INFO:assignment.dimensionality.dropfirstn:Init Drop First N feature selector (k -> (k-1) dimensions)
INFO:assignment.dimensionality.pca:Trained PCA reducer (702 -> 11 dimensions)
INFO:assignment.pipeline:Trained pipeline reducers (900 -> 10 dimensions)
INFO:assignment.classifier:Performed LOO testing
INFO:assignment.classifier:Built normalised confusion matrix
INFO:assignment.classifier.weightedknn:Trained weighted k-NN classifier with fuzzy = True
INFO:assignment.pipeline:Trained pipeline classifier
INFO:assignment.data.wordsearch:Classified all letters
INFO:assignment.wordsearch.masks:Computed fit scores for word = barry
INFO:assignment.wordsearch.masks:Computed fit scores for word = beardshaw
INFO:assignment.wordsearch.masks:Computed fit scores for word = bridgeman
INFO:assignment.wordsearch.masks:Computed fit scores for word = brown
INFO:assignment.wordsearch.masks:Computed fit scores for word = cane
INFO:assignment.wordsearch.masks:Computed fit scores for word = crowe
INFO:assignment.wordsearch.masks:Computed fit scores for word = don
INFO:assignment.wordsearch.masks:Computed fit scores for word = fish
INFO:assignment.wordsearch.masks:Computed fit scores for word = flowerdew
INFO:assignment.wordsearch.masks:Computed fit scores for word = hoare
INFO:assignment.wordsearch.masks:Computed fit scores for word = jekyll
INFO:assignment.wordsearch.masks:Computed fit scores for word = jellicoe
INFO:assignment.wordsearch.masks:Computed fit scores for word = kent
INFO:assignment.wordsearch.masks:Computed fit scores for word = langley
INFO:assignment.wordsearch.masks:Computed fit scores for word = nesfield
INFO:assignment.wordsearch.masks:Computed fit scores for word = paine
INFO:assignment.wordsearch.masks:Computed fit scores for word = paxton
INFO:assignment.wordsearch.masks:Computed fit scores for word = peto
INFO:assignment.wordsearch.masks:Computed fit scores for word = repton
INFO:assignment.wordsearch.masks:Computed fit scores for word = robinson
INFO:assignment.wordsearch.masks:Computed fit scores for word = roper
INFO:assignment.wordsearch.masks:Computed fit scores for word = shenstone
INFO:assignment.wordsearch.masks:Computed fit scores for word = vanbrugh
INFO:assignment.wordsearch.masks:Computed fit scores for word = wright
INFO:assignment.data.wordsearch:Found best fits for all words

[default] 0:python* 1:bash-
```