Hw10

Due on 12/11 (F)
Total points: 110
Hw10: feature engineering

• Goal: to determine how the choice of feature templates affects system performance.

• Task: build a classifier to distinguish blogs from left and right media.

• Data: from four sources

<table>
<thead>
<tr>
<th>Site</th>
<th>Training docs</th>
<th>Test docs</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNC Blogs</td>
<td>556</td>
<td>54</td>
</tr>
<tr>
<td>Fox News Blogs</td>
<td>2700</td>
<td>353</td>
</tr>
<tr>
<td>DNC Blogs</td>
<td>448</td>
<td>61</td>
</tr>
<tr>
<td>Mother Johns Blogs</td>
<td>2736</td>
<td>337</td>
</tr>
</tbody>
</table>
Steps:

• Q1 (and Q2): to create the baseline system with unigram features only

• Q3 (and Q4): add new features; you decide what kind of features to add.

• Q5: a wrapper for the whole pipeline

• Q6: run the wrapper in Q5 with different feature combinations and report the results
Q1: create a feature vector for one doc, using unigram features only

• Command: proc_file1.sh input_file targetLabel output_file

• Same as proc_file.sh in hw8 except for the preprocessing step.

• In proc_file1.sh:
  ▪ Tokenization: replace all the chars that are not [0-9\-a-zA-Z] with whitespace, and lowercase all the remaining chars.
  ▪ Features: use word unigrams only; i.e., break the text into token by whitespace, and each word will become a feature.
  ▪ Feature values: the feature values will be the frequency of the words.
Q2: create_vectors1.sh, that creates feature vectors for multiple docs

• Command: create_vectors1.sh output_vector_file dir1 dir2 ...

• Files under the same dir belong to the same class. The class label is the basename of the dir.

• Same as create_vectors.sh in hw8 except that
  • It calls proc_file1.sh, not proc_file.sh
  • It deals with one data set (training or test set) only, so no need to split the data.
Q3: proc_file2.sh, which introduces more features

• Command: proc_file2.sh  param_file  input_file  targetLabel  output_file

• Define new features: what cues do you think would help distinguish left vs. right blogs.

• Divide the features into groups and name groups F1, F2, ...:
  • Ex: All the affix features can be in one group.
  • You need to have at least three groups, excluding word unigram features.
  • In your note file, explain the meaning of each feature group.

• param_file specifies what feature groups are used when creating the feature vector.

  F1=1
  F2=0
  F3=1
  ...

...
Q4: create_vectors2.sh, which calls proc_file2.sh to generate feature vectors

• Command: create_vectors2.sh param_file output_vector_file dir1 dir2 ...

• It should be very similar to create_vectors1.sh except that it calls proc_file2.sh, not proc_file1.sh
Q5: train_classify.sh, a wrapper

- Command: `train_classify.sh param_file train_dir test_dir output_dir`

- It does the following:
  - Call `create_vectors2.sh` to create training and test vectors
  - Call `mallet` to convert vectors to binary format
  - Call `mallet` to build a MaxEnt model using training data
  - Call `mallet` to classify the test data

- `param_file` specifies what features are used when calling `create_vectors2.sh`

- All the output files are written to `output_dir`
Q6: run the wrapper to get experimental results

<table>
<thead>
<tr>
<th>Expt id</th>
<th>Feature groups</th>
<th>Training accuracy</th>
<th>Test accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>expt1 (baseline)</td>
<td>F1 (unigrams only)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>expt2</td>
<td>e.g., F1+F2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>expt3</td>
<td>e.g., F1+F3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>expt4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>e.g., F3+F4+F6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

At least five rows (expt1 – expt5). Output is stored in hw10/expt1, ... You decide what feature combinations you want to try. The first row is the baseline, and the last row is your best result. Other than expt1, a feature combination does not need to include F1. In your note file, what conclusion do you draw from this table?
Total points: 110

• Q1: 10   implement the baseline features (feel free to reuse hw8 code)
• Q2: 10   call code for Q1 (feel free to reuse hw8 code)
• Q3: 40   define new features
• Q4:  5   call code for Q3
• Q5: 10   write a wrapper
• Q6: 35   run experiments and draw conclusion