Extreme Computing Assignment Two

Miles Osborne miles@inf.ed.ac.uk

October 31, 2011

This assignment deals with distributed machine learning. You should use the teaching Hadoop Cluster and any programming language you want.

1 Task

The task is to create a system which takes text that is all in lower case and produces a version where the case is restored. Here is an example input sentence:

> john loves a mary

You should produce the following version:

> John loves a Mary

You can assume that there are only two possibilities –a word is either all in lower case or is capital-initial. You should not consider other possibilities (eg words that are all capitalised, or contain mixed capitalisation).

You can assume the data consists of space delimited sentences and there is no need to tokenise the data (split punctuation, etc). **You do not need to preserve the original word ordering**.

1.1 Representation

You will need to represent the training data in terms of a vector of features and values, along with the target label. 'Features' and 'values' describe how you represent words and 'labels' encode the correct case for each word.

For example, possible features might be: the word itself, the last three letters and the last two letters. Each feature might have a binary value, taking the value 1 if that feature was present, and 0 otherwise. Note that you need only represent the features that are active on an example (and so only need list those that have a value of one). Here is an example. Suppose the set of features included:

Feature	Identifier
W=John	1
W=loves	2
W=a	3
W=Mary	4
W=ohn	5
W=ary	6
...	...

(there are other features not listed here)

The word:

John

is represented using the features and values:

Feature	Value
1	1
5	1

We can encode case information using two labels:

Label	Meaning
0	Word is lower case
1	Word is upper case

Putting this together, the word *John* might be represented using the list of feature-value pairs and label:

(1 1) (5 1) 1.

1.2 Naive Bayes

The naive Bayes classifier works as follows. For each feature f_i, we count the number of times we see it with each label. In the previous example, we see feature 1 once with the label 1 and zero times with the label 0. For the two labels, call this $\text{freq}(f_i, 1)$ and $\text{freq}(f_i, 0)$.

We then estimate the probability $p(f_i \mid l)$ as:

$$p(f_i \mid l) = \frac{\text{freq}(f_i, l)}{\text{freq}(f_i, 0) + \text{freq}(f_i, 1)}$$

Because we will have many features F for each example, we multiply them all together:

$$p(l \mid F) = \prod_{i=1}^{n} p(f_i \mid l)$$

When working-out the correct label, we insert both labels into this formula and use the label that produces the highest probability. If we predict 1, we say the word is upper-case (and lower-case otherwise).

1.3 Data

You can use the following data (on DFS):

/user/miles/data/small.txt
/user/miles/data/large.txt
/user/miles/data/test.txt
/user/miles/data/test-truth.txt

The first two files contains sentences taken from the Web, one per line. The **small.txt** file is for developing and testing your code; when you are happy that it works, you should use the larger **large.txt**. The third file is for testing and is all lower case. The final file contains the associated real case. All of your actual results should be done using the larger file and where possible you should use Map Reduce and DFS.

1.4 Evaluation

Performance will be in terms of the number of words that are correctly capitalised. For example, if you produced:

John loves a mary

Then you have labelled three words correctly and so performance is 3/4. Note that numbers are always 'lower case' and any mixed cased words in the test data should be assumed to be just upper-case initial.

2 Tasks

For all tasks, you need to explain your approach and justify any decisions made. You should give evidence of testing and supply any code written.

1. Use the following set of features: the word itself, the last two letters, the last three letters, the first two letters, the first three letters.
2. **Task One:** Write a program on Unix which uses naive Bayes to assign the correct case to words. (This is to make sure you understand how naive Bayes works, before dealing with Hadoop and scaling). Call this the *Unix version.* **(5 marks)**
3. **Task Two:** Use Hadoop to write one or more Map Reductions which computes $P(f_i \mid l)$, using the supplied training data on DFS. Call this your *model.* **(10 marks)**
4. **Task Three:** Write one or more Map Reductions which converts the test file into features. **(2 marks)**
5. **Task Four:** Use Hadoop, your model and your feature representations to compute the most likely label for each word. **(5 marks)**

6. **Task Five:** Finally, use Hadoop to convert the lower-case words into their correct case and evaluate your results. Verify that your approach produces the same results as your Unix version **(3 marks**)

3 Marking Guidelines

Each part has marks allocated, which gives a guide to how much effort you should spend. Bonus marks will be awarded if you are creative, you use an efficient approach or you surprise me. Marks will be deducted if you are inefficient or make bad use of Hadoop.

4 Submission

Your submission should consist of one *plain text file*:

No Microsoft Word, open-document or PDF FILES

with sections for each part of this assignment –This is so that I know which part of the file answers which question.

Include all programs you have written **at the end the your report**, clearly marking which parts correspond with which section. Make sure you comment your code so that I can understand what you are doing.

Use the submit program to make the submssion. Name your file exc-mr.txt; you can submit it from the command line as:

```
submit exc 2 exc-mr.txt
```

The deadline is 1st **December, 4:00 pm**