Statistical Approaches to NLP
PA 1 – Writeup

José Andrés Molina

September 21, 2016

1 Train Function

1.1 The Model

The baseline train function stored as a model a double of dictionaries, where one dictionary contained conditional probabilities of the type $P(\text{feature}|\text{label})$ for all the features and labels, and the second contained a the probabilities of the labels themselves, or $P(\text{label})$ for all the labels.

This was later replaced with defaultdicts of joint probabilities, or $P(\text{feature, label})$ for all features for all labels, instead as they are easier to read in the code and faster to calculate. (Defaultdicts also look cleaner than code than having if-statements checking for existence.) Storing joint probabilities makes it so that there is no need to store the probabilities of the labels themselves, and requires just one dictionary of probabilities in the model instead of two.

Original the model just stored (and in classify later compared) the counts, but this was quickly replaced with probabilities in the baseline and later with log probabilities for the final version.

1.2 Smoothing

The baseline originally had no smoothing, but this was quickly substituted for +1.0 Laplace smoothing. Then later with tuning over empirical trials, this number ultimately settled upon was +0.1 smoothing.

The smoothing was done over the set of all the features already seen but not with a particular label, as well using 'UNK' as a catch-all for any features not seen in the train set.

2 Classify Function

The classify method in the baseline took the probabilities of all features in the instance trying to classify given all the labels, multiplied them together, multiplied by the probability of the labels, and then took the label with the highest probability.

$$\text{"most likely label"} := \underset{l \in L}{\arg\max}\Big(P(l) \prod_{f \in F} P(f|l)\Big)$$

In the above formula, f refers to a feature, F to the set of features, l to a label, and L to a set of labels.

The ultimate form of this method instead uses joint log probabilities, and merely adds them up for all features, and returns the argmax of that.

$$\text{"most likely label"} := \underset{l \in L}{\arg\max}\Big(\sum_{f \in F} \log\big(P(f, l)\big)\Big)$$

3 Feature Engineering

3.1 Even-Odd Corpus

The even odd corpus reaches 100% accuracy and F_1 without a need to change the feature, as this feature "$x \mod 2 = 0$" is a perfect indicator of even numbers.

3.2 Names Corpus

For this particular dataset, just taking the first and last letters of a name as features were the best indicators of the gender. Other NLTK features (letters it contains, number of each letter it contains, etc.) were found to actually slightly decrease the accuracy, as well as the length of the name as a feature.

3.3 Blogs Corpus

3.3.1 Imbalanced Blog Corpus

No matter what features were changed in the BagOfWords class, the classifier would always label all as 'M' (male) to receive a 91% accuracy because the data was so skewed towards male.

3.3.2 Balanced Blog Corpus

For the balanced corpus, a multitude of features attempted affected it. Most of the features actually ended up decreasing the accuracy when compared to a simple split bag-of-words approach. Among those features were trigrams, removal of stopwords, lemmatization (using NLTKs lemmatizer), part-of-speech tagging (using NLTK as well). It would make sense that in a small corpus, trigrams would not yield enough information as a feature to be useful, and for similar reasons lemmatization might also just get rid of useful information by decreasing the number of features. Removing stopwords intuitively should just focus the features towards more useful and descriptive words, but also limits the features and ended up decreasing results. Part Of Speech tagging also intuitively feels like it should help as it has been noted that women tend to use more adjectives than men for example, but ended up not working as it should. Perhaps limiting this or counting different POS might have affected it instead and may be a good thing to try in the future. One last thing attempted that slightly decreased accuracy (by about 1%) was returning a set of the words instead of a list and removing duplicates.

A few things however increased the accuracy. The first attempted one and the simplest was making everything lowercase, which slightly improved the results from the very start. Tokenizing words and removing punctuation slightly increased accuracy as did added the length features, average word length and number of words (i.e. length of document). However, adding a list of bigrams found in the document significantly improved results (by over 5%) and was a major part in what pushed the classifier's accuracy on this corpus over 72% accuracy.

4 F Measures

4.1 Balanced Blog Corpus

The following are the recall, precision, and F_1 results per label for the balanced blog corpus. The accuracy was 72.84% and the average F_1 was 72.75%.

Male recall: 79.82%	Female recall: 66.10%
Male precision: 69.46%	Female precision: 77.22%
Male F_1: 74.28%	Female F_1: 71.23%

Most interestingly about these data is that the Male label had better recall but worse precision and the Female label had better precision but worse recall. Both had comparable F_1 measures however.

4.2 Imbalanced Blog Corpus

The following are the recall, precision, and F_1 results per label for the imbalanced blog corpus. The accuracy was 91.05% and average F_1: 47.65%.

Male recall: 100.0%	Female recall: 0.0%
Male precision: 91.05%	Female precision: 0.0%
Male F_1: 95.31%	Female F_1: 0.0%

The Naïve Bayes Classifier labeled all of the test documents in this dataset as "Male", and since it was skewed 90+% towards it, received a 90+% accuracy. The F_1 however is horrendous as it did not label a single "Female" document correctly. This is a good example of why accuracy as a measure can be misleading and why F_1 is a better metric for these data.

4.3 Even-Odd Corpus

The following are the recall, precision, and F_1 results per label for the even-odd corpus. The accuracy was 100.0%, and the average F_1 was also 100.0%.

False recall: 100.0%	True recall: 100.0%
False precision: 100.0%	True precision: 100.0%
False F_1: 100.0%	True F_1: 100.0%

4.4 Names Corpus

The following are the recall, precision, and F_1 results per label for the names corpus. The accuracy was 75.20% and the average F_1 was 70.05%.

Male recall: 45.93%	Female recall: 92.19%
Male precision: 77.35%	Female precision: 74.60%
Male F_1: 57.64%	Female F_1: 82.47%

Interestingly, the recall (and therefore F_1) for "Female" are significantly better than for "Male" for this dataset.