Assignment 3: Naive Bayes Classifiers

11-411/611 Natural Language Processing

Due: Oct 4, 2022

Introduction

In this assignment, you will create a ngram (upto bi-gram for this assignment) character based Naive Bayes text classifier. Your classifier will distinguish between 6 different languages.

Data

The training dataset includes 53,856 different sentences distributed among 6 different languages which include Hausa, Indonesisan, Manobo, Tagalog, Swahili and Nahuatl. The dev set (for testing locally) includes 17,791 different sentences from the same 6 languages.
The train and dev files have one sentence per line. The language of the sentence is given first, followed by a tab character, followed by the sentence (each word is separated by whitespace).

Task: Programming [100 points]

Subtask 1: Implementation [50 points]

In your file `naivebayes.py`, create a Naive Bayes classifier using the provided template.

1. For each sentence, Naive Bayes classifier returns the class $\hat{c}$ which has the maximum posterior probability. By applying Bayes' rule we can get:

$$\hat{c} = \underset{c \in C}{argmax} P(c|sentence) = \underset{c \in C}{argmax} \frac{P(sentence|c)P(c)}{P(sentence)}$$

   We can drop the denominator $P(sentence)$.

2. Prior and Likelihood: the Naive Bayes model assumes that all ngrams in a sentence are independent of one another given the class. We can write the likelihood of a sentence as:

$$P(f_1, ..., f_n|c) = \prod_{i=1}^{n} P(f_i|c)$$

where f_i is the i^{th} term in the sentence and c is the class of sentence. You will need to write program to estimate the multinomial distributions P(term—class) and the prior P(c) from the training dataset.

3. Use add-one smoothing when estimating probability. Make sure to add the size of the vocabulary (number of unique character ngrams in all classes) to the denominator when normalizing.

4. Probability calculations are done in log space, to avoid underflow and increase speed.

5. You do not need to deal with the problem of new bigrams.

Subtask 2: Evaluation [50 points]

The points have been divided depending on the performance of your model on the test set (on gradescope). All metrics have been already implemented in the handout.

The rubrik is as follows:

- 10 points — macro-averaged F1 $>$ 0.90
- 10 points — micro-averaged F1 $>$ 0.90
- 15 points — macro-averaged F1 $>$ 0.98
- 15 points — micro-averaged F1 $>$ 0.98

For your reference, Macro-Averaged F1-score and Macro-Averaged F1-score are calculated as follows. You are encouraged to go through the implementations of all metrics in the handout for your understanding.

- Micro-Averaged F1-score(MF)

$$MF = \frac{2 * MP * MR}{MP + MR}$$

  where MP is Micro-Averaged Precision and MR is Micro-Averaged Recall.

- Macro-Averaged F1-score(MAF)

$$MAF = \frac{2 * MAP * MAR}{MAP + MAR}$$

  where MAP is Macro-Averaged Precision, MAR is Macro-Averaged Recall.

To evaluate your implementation, train a Naive Bayes classifier using the training data `train.txt`. Use the dev file `dev.txt` to evaluate your classifier locally. The code can be run using the command

```
python3 naivebayes.py train.tsv dev.tsv
```

Submission Guidelines

Please submit a a file called `naivebayes.py` which implements your classifier and reports MP, MR, MF, MAP, MAR, MAF.