COMP 472 Project 2
Naive Bayes Classifier

Matteo Esposito[1], Matthew Liu[2], and Kabir Soni[3]

[1] 40024121 matteoesposito97@gmail.com
[2] 40029238 matthew.jx.liu@gmail.com
[3] 40033019 kabirsoni524@gmail.com

1 Introduction & Technical Details

This project was developed using python 3.7.4 64-bit.

1.1 Files

The file structure of our project is as follows:

Table 1. Files in project 1

Directory	Filename	Usage
`out/`	*	Trace and evaluation files for each run.
`out_BYOM/`	*	Trace and evaluation files for each BYOM run.
`src/`	`utils.py`	Helper functions for I/O and input parsing.
	`NBClassifier.py`	Naive Bayes Classifier class. A collection of methods used to implement Naive Bayes Classification.
	`Ngram.py`	Ngram class, used in all classifications.
	`BYOM.py`	Personalized model class.
	`main.py`	Reading training set, training classifier, predicting languages on test set and writing out results.

1.2 Packages

We used a total of 7 packages in our project, 5 existing, along with our 2 internal packages (board and node).

1. Existing
   – `shutil` and `os`: Folder and file management in the creation and deletion of output folders for our search and solution files.
   – `math`: Calculating log base 10 probabilities as part of the score function of each tweet.
   – `copy`: Used to create deep copies in the initialization of ngrams in the NBClassifier class.

- `decimal.Decimal`: Used to format the probability output for the trace file.
- `string`: Used to populate vocabulary 1 and 2 with ascii characters.

2. Internal
   - `NBClassifier`: Class that is used to represent the classifier, which takes a vocabulary selection, ngram size, delta/smoothing value and train/test file links.
   - `Ngram`: Class used to represent the Ngram used in the NBClassifier class.

1.3 NBClassifier and Ngram Classes

The **Ngram** class will be used in the NBClassifier class. It allows for a concise way to store a language, count/frequency table, probability table and language probability. This class stores the probability calculation, smoothing and language probability methods which are called in the train method of NBClassifier.

The **NBClassifier** class will have as main attributes a language, count table, probability table and size (n), which will all be stored in a size-n Ngram. It will also hold a language probability. It is from the NBClassifier object in `main.py` that we will call the train and predict methods.

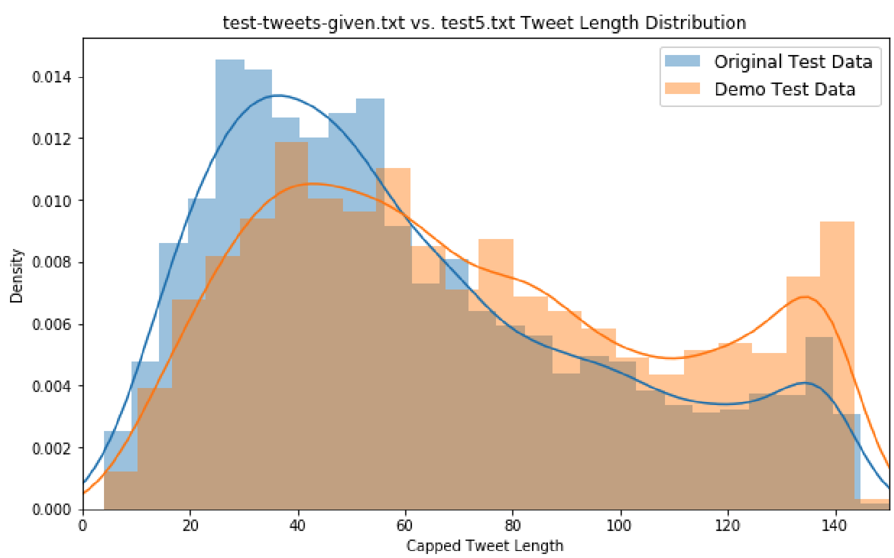
Note on vocabulary creation: Vocabularies 0 and 1 are generated by combining ascii character sets from the `string` library. Vocabulary 3 is created by... Matt add something here that talks about the creation of vocab2 for Ngrams

2 Dataset Impact & Analysis

In our exploratory data analysis we assess 2 characteristics of the test sets, namely tweet length and language frequency.

2.1 Tweet Length

Rounding down all tweets with length larger than 150 charaters down to 150 (as they make up less than 0.5% of the observations), we get the following capped tweet length distribution curves for the provided and demo test sets of tweets.



The main observation we can notice from the overlayed distribution plots is that the demo test set contains a greater proportion of large tweets (tweets with length over ~ 70 characters). The effect this could have on the accuracy of our classifications is that we are introducing a larger amount of unigrams, bigrams and trigrams from the tweet being assessed by the classifier and therefore introducing the potential for a greater amount of incorrectly labelled character patterns.

2.2 Language Frequency

Generating a frequency table of the languages in the provided test dataset we observe the following:

```
orig_test.lang.value_counts()

    es    3926
    pt    2020
    en     505
    eu     376
    ca      75
    gl       1
```

We notice that there is a single observation in the 'gl' class. This will have an effect on our precision, recall and f1 values. In the case where we do not correctly classify that observation into the 'g1' class, our true positive value for the gl class will be 0, making precision and recall also 0 and yielding an f1 value of 0/0 (which we set to 0 in this case).

Generating a frequency table of the languages in the demo test dataset we observe the following:

```
demo_test.lang.value_counts()

    es    4572
    ca    1387
    gl     504
    en     483
    pt      96
```

We notice that there are no observation in the 'eu' class. Every classification of a test tweet into the 'eu' class will result in a direct increase in the number of false positives, affecting our validation metrics negatively (decreasing precision and f1).

3 Our Model (BYOM)

Kabir

4 Result & Experiment Analysis

Check in appendix for all necessary confusion matrices

5 Team Responsibilities

The breakdown of tasks were as follows:

1. Matteo Esposito

  - Project structure
  - NBClassifier & Ngram class
  - utility functions `utils.py`
2. Matthew Liu
  - NBClassifier & Ngram class
  - utility functions `utils.py`
3. Kabir Soni
  - Personalized model (BYOM)

References

1. S.J. Russel, P. Norvig: Artificial Intelligence: A Modern Approach. 3rd edn. Pearson, Harlow (1994)

6 Appendices

6.1 Appendix A - Confusion Matrices (Given Test Dataset `test-tweets-given.txt`)

Table 2. Cross tabulation of predictions, $V = 0, n = 1, \delta = 0$

		Predicted					
		ca	en	es	eu	gl	pt
Actual	ca	24	3	44	1	1	2
	en	7	287	205	11	0	6
	es	44	140	3635	63	3	88
	eu	3	16	107	253	0	1
	gl	0	0	1	0	0	0
	pt	16	62	1365	11	1	600

Table 3. Cross tabulation of predictions, $V = 1, n = 2, \delta = 0.5$

		Predicted					
		ca	en	es	eu	gl	pt
Actual	ca	59	1	14	0	0	1
	en	8	443	50	7	0	8
	es	110	66	3584	84	41	88
	eu	5	6	41	323	0	5
	gl	0	0	0	0	0	1
	pt	61	23	397	8	38	1528

Table 4. Cross tabulation of predictions, $V = 1, n = 3, \delta = 1$

		Predicted					
		ca	en	es	eu	gl	pt
Actual	ca	53	1	21	0	0	0
	en	4	386	121	1	0	4
	es	10	15	3918	6	1	23
	eu	5	4	140	230	0	1
	gl	0	0	1	0	0	0
	pt	16	8	480	0	1	1550

Table 5. Cross tabulation of predictions, $V = 2, n = 2, \delta = 0.3$

		Predicted					
		ca	en	es	eu	gl	pt
Actual	ca	62	1	12	0	0	0
	en	9	438	60	7	0	2
	es	70	55	3721	57	11	59
	eu	4	8	51	311	0	6
	gl	0	0	0	0	0	1
	pt	47	20	340	6	14	1628

Table 6. Cross tabulation of predictions (BYOM), $V = 1, n = 2, \delta = 0.5$

		Predicted					
		ca	en	es	eu	gl	pt
Actual	ca	51	1	23	0	0	0
	en	3	406	102	2	0	3
	es	11	16	3914	7	0	25
	eu	2	4	109	264	0	1
	gl	0	0	1	0	0	0
	pt	17	9	412	0	2	1615

6.2 Appendix B - Confusion Matrices (Demo Dataset `test5.txt`)

Table 7. Cross tabulation of predictions, $V = 0, n = 1, \delta = 0$

		Predicted					
		ca	en	es	eu	gl	pt
Actual	ca	418	52	868	5	5	43
	en	7	299	171	2	0	4
	es	51	164	4217	20	5	132
	gl	7	8	441	1	8	41
	pt	0	5	61	10	0	20

Table 8. Cross tabulation of predictions, $V = 1, n = 2, \delta = 0.5$

		Predicted					
		ca	en	es	eu	gl	pt
Actual	ca	1064	23	256	7	9	32
	en	34	380	59	2	0	8
	es	150	99	4072	32	107	129
	gl	19	9	266	4	122	86
	pt	4	1	19	1	2	69

Table 9. Cross tabulation of predictions, $V = 1, n = 3, \delta = 1$

		Predicted					
		ca	en	es	eu	gl	pt
Actual	ca	1041	4	338	0	0	8
	en	21	329	131	0	0	2
	es	20	40	4480	1	6	42
	gl	3	1	452	0	19	31
	pt	2	0	29	0	0	65

Table 10. Cross tabulation of predictions, $V = 2, n = 2, \delta = 0.3$

		Predicted					
		ca	en	es	eu	gl	pt
Actual	ca	1107	18	239	7	2	18
	en	29	374	72	1	0	7
	es	114	84	4227	20	52	92
	gl	11	6	326	1	106	56
	pt	1	0	18	1	0	76

Table 11. Cross tabulation of predictions (BYOM), $V = 1, n = 2, \delta = 0.5$

		Predicted					
		ca	en	es	eu	gl	pt
Actual	ca	1071	7	305	0	0	8
	en	22	342	116	0	0	3
	es	19	44	4465	1	13	47
	gl	4	2	415	0	42	43
	pt	2	0	25	0	0	69

6.3 Appendix C - Miscellaneous Code

Confusion Matrix

```python
import pandas as pd

# Go through all the trace files generated during the demo.
for file in ['trace_0_1_0.txt', 'trace_1_2_0.5.txt', 'trace_1_3_1.txt', 'trace_2_2_0.3.txt', 'trace_1_BYOM_0.5.txt']:
    df = pd.read_csv(f'total_out/{file}', delimiter="	", names=('id','lang','prob','pred_lang','res'))
    print(pd.crosstab(df['lang'], df['pred_lang']).to_latex())
```

Distribution Plot

```python
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt

orig_test = pd.read_csv(f'input/test-tweets-given.txt', delimiter="\t", names=('id','user','lang','tweet'))
demo_test = pd.read_csv(f'input/test5.txt', delimiter="\t", names=('id','user','lang','tweet'))

fig, ax = plt.subplots(figsize=(10,6))

# Cap data at tweet length of 150 and create individual plots.
orig_test['tlen'] = orig_test['tweet'].apply(len)
orig_test['tlen_capped'] = np.where(orig_test['tlen'] > 150, 150, orig_test['tlen'])
sns.distplot(orig_test.tlen_capped, ax=ax, label="Original Test Data")
```

```

demo_test['tlen'] = demo_test['tweet'].apply(len)
demo_test['tlen_capped'] = np.where(demo_test['tlen'] > 150, 150,
    demo_test['tlen'])
sns.distplot(demo_test.tlen_capped, ax=ax, label="Demo Test Data")

# Settings
plt.xlim(0, 150)
plt.title('test-tweets-given.txt vs. test5.txt Tweet Length Distribution
    ')
plt.xlabel('Capped Tweet Length')
plt.legend(prop={'size': 12})
plt.ylabel('Density')
```