COL774: Assignment 4
Naive Bayes & Collaborative Filtering

Released on: 2nd October, 2024

Due date [Naive Bayes]: 7.00 PM, 13 October, 2024 [Sunday]

-
- This assignment has two main parts: 1) Naive Bayes and 2) Collaborative Filtering.
- All the announcements related to the assignment will be made on Piazza. The access code is wbd8avkblhb. You are strongly advised to look at the relevant Piazza posts regularly.
- You should use Python 3 for all your programming solutions.
- Your assignments will be auto-graded. Make sure you test your programs before submitting. We will use your code to train the model on the train set and predict on the test set.
- Input/output format, submission format, and other details are included. Your programs should be modular enough to accept specified parameters.
- You may submit the assignment **individually or in groups of 2**. No additional resource or library is allowed except for the ones specifically mentioned in the environment files. If you still wish to use some other library, please consult Piazza. If you choose to use any external resource or copy any part of this assignment, you will be awarded a D or F grade or **DISCO**.
- ~~You should submit work of your own. You should cite the source in the report if you choose to use any external resource for the competitive part. You will be awarded a D or F grade or **DISCO** in case of plagiarism.~~
- **Late submission policy**: There will be a 7% penalty for each late day. So, if you submit four days after the deadline, you will be graded out of 72% of the assignment's weightage. Any submission made after seven days would see a fixed penalty of 50%.
- Post your doubts on Piazza. Email Burouj Armgaan for doubts that may disclose implementation details.

Updates

- (8/10/24) Added checker files and some instructions. See section 1.7.
- (11/10/24) Use `quoting=3` while reading the files in pandas. Otherwise some rows are omitted.
- (11/10/24) Preprocessing in the non-competitive part: Just lowercase the data and split on whitespace.
- (11/10/24) Remove stopwords before stemming.
- (11/10/24) The same environment file will be used for the competitive and the non-competitive parts.
- (11/10/24) Time constraints: 5 minutes for each non-competitive part; 15 minutes for the non-competitive part.
- (11/10/24) Corrected the Bernoulli checker files.
- (11/10/24) Included more checker files: per-class word frequencies and word parameters.

1 Naive Bayes

In this assignment, we will implement Naive Bayes to detect fake news. We will use the LIAR dataset. LIAR is a publicly available dataset for fake news detection. A decade-long of 12.8K manually labeled short statements were collected in various contexts from politifact.com. It is a multi-class classification dataset where the labels indicate the truthfulness of the statement. It considers six fine-grained labels for the truthfulness ratings: `pants-fire, false, barely-true, half-true, mostly-true, true`. Find out more at the dataset's homepage. Link to the assignment dataset

Notes

- The checker scripts will be released soon.
- Use natural logarithms to handle underflow issues during probability calculations.
- Apply Laplace smoothing with a smoothing parameter with c=1 to adjust for zero probabilities in the feature counts.
- Do not shuffle the test set.

1.1 Non-competitive

In this part, we'll classify the input based on the news text alone (column 2 of the dataset).

1. (30 points) Implement the Naive Bayes using the **Bernoulli event model**, as indicated in section 4.2 of the cs229 lecture notes. Use uni-grams as features to classify each instance into one of the given categories. In an N-gram, we consider N consecutive words to be one feature. When N=1, i.e., each word is used as a feature, it is called a 1-gram or a uni-gram.
   The provided dataset is in the raw format. It includes words such as "of", "the", "and" called stop-words. These words may not be relevant for classification. In fact, their presence can sometimes hurt the classifier's performance by introducing noise into the data.
   Similarly, the raw data treats different forms of the same word separately, e.g., eating and eat would be treated as separate words. Merging such variations into a single word is called stemming. Perform stopword removal and stemming before using Naive Bayes. You can use the `nltk` library to remove stopwords and perform stemming.
2. (30 points) Repeat part 1, but this time implement Naive Bayes using the **Multinomial event model**, as indicated in section 4.2.2 of the cs229 lecture notes.
3. (20 points) Feature engineering is an essential component of Machine Learning. It refers to the process of manipulating existing features/constructing new features in order to help improve the performance of the prediction task. For example, instead of using each word as a feature, you may treat bi-grams (two consecutive words) as a feature. Learn new Bernoulli and Multinomial models that use both uni-grams and bi-grams.

1.2 Competitive

(20 points) The LIAR dataset provides rich features beyond the news text:

- Column 4: The subject(s).
- Column 5: The speaker.
- Column 6: The speaker's job title.
- Column 7: The state info.
- Column 8: The party affiliation.
- Columns 9-13: The total credit history count, including the current statement.
  1. 9: Barely true counts.
  2. 10: False counts.
  3. 11: Half-true counts.
  4. 12: Mostly true counts.
  5. 13: Pants on fire counts.
- Column 14: the context (venue/location of the speech or statement).

Incorporate these features into your Naive Bayes model to improve performance by providing more context and information about the statements. Explain your approach in a report.

1.3 Running your code

Use this argument parser:

```
from argparse import ArgumentParser

parser = ArgumentParser()
parser.add_argument("--train", type=str, required=True)
parser.add_argument("--test",  type=str, required=True)
parser.add_argument("--out",   type=str, required=True)
parser.add_argument("--stop",  type=str, required=True)

args = parser.parse_args()

print("Train file:", args.train)
print("Test file:", args.test)
print("Output file:", args.out),
print("Stopwords file:", args.stop)
```

We'll use these commands to run your code:

```
# Non-competitive
# Binomial
python nb1_1.py --train <path to train.csv> --test <path to val.csv>
--out <path to nb1_1.txt> --stop <path to stopwords.txt>

# Multinomial
python nb1_2.py --train <path to train.csv> --test <path to val.csv>
--out <path to nb1_2.txt> --stop <path to stopwords.txt>

# Binomial bigrams
python nb1_3a.py --train <path to train.csv> --test <path to val.csv>
--out <path to nb1_3.txt> --stop <path to stopwords.txt>

# Multinomial bigrams
python nb1_3b.py --train <path to train.csv> --test <path to val.csv>
--out <path to nb1_3.txt> --stop <path to stopwords.txt>

# Competitive
python nb2.py --train <path to train.csv> --test <path to val.csv>
--out <path to nb2.txt> --stop <path to stopwords.txt>
```

1.4 Output format

Output the predicted test data labels in a text file exactly as spelled in the dataset. Each line should have one prediction. Here's an example:

```
false
false
half-true
mostly-true
true
barely-true
true
pants-fire
```

1.5 Heads-up

- Handle any blank lines at the end when reading and writing to text files.
- When dealing with lists or NumPy arrays, remember that assigning a list/array to a variable doesn't create a copy but creates a reference. This holds when passing them as function arguments. Create copies to avoid this.

```
a = [1, 2, 3]
b = a
b[0] = 100 # This will change "a"

c = [1, 2, 3]
d = np.array([1, 2, 3])
def func(l):
    l[0] = 200

func(c) # This will change "c"
func(d) # This will change "d"

from copy import deepcopy
b = deepcopy(a)
b[0] = -1 # This won't change "a"
```

1.6 Submission Instructions

Submit a zip file named `kerberos_1_kerberos_2_A4.zip` on Moodle, e.g., `csz228001_cs5200424.zip`. Unzipping it should return a folder with the same name as the zip file containing just your Python scripts. There should not be any sub-folders. Submit your report as a PDF on Gradescope.

Only one teammate should submit on Moodle and Gradescope. On Gradescope, mark your teammate before submitting.

1.7 Checker Files and more

Checker files

- The checker files, environment files, and stopwords are up. Find them here.
- Each checker file is a NumPy array of size (n, c), where n is the dataset size and c is the number of classes. The classes follow this order: `pants-fire, false, barely-true, half-true, mostly-true, true`. Entry (i, j) refers to the product of word parameters and the class parameter for data instance i:

$$\text{Entry}_{(i,j)} = \phi_{y=j} \prod_{k=0}^{d} \phi_{k|y=j}$$

- You can take the argmax of a matrix to get the predictions and map them to the original labels to compute any metrics.

Environment

- We have provided two environment files: one for a `conda` environment and one for a `venv` environment. You may use either one.
- You'll need Anaconda or Miniconda for this. Create a conda environment as follows:

```
pip cache purge
conda env create -f requirements.yml
conda activate col774_a4
```

- For `venv`:

```
python3 -m venv .col774_a4
source .col774/bin/activate
pip cache purge
pip install -r requirements.txt
deactivate
```

- Note that we'll use this environment while grading.

Notes

- Choose the first occurrence to resolve tie-breaks when taking the argmax (This is the default behavior of `np.argmax(my_array)` and `my_list.index(max(my_list))`. To compare against the checker files order your classes as follows: `pants-fire, false, barely-true, half-true, mostly-true, true`, so that if argmax comes out to be 2, you can map it to `barely-true`.
- Convert all the text to lowercase before stopword removal and stemming.
- To avoid any download issues while grading, we are providing you with a file containing the stopwords. Section 1.3 has been altered to account for the stopwords file.
- Use nltk's `PorterStemmer` for stemming.
- Do not use the test data's labels anywhere in your code. We may pass dummy labels in their place while grading to avoid any cheating attempts.
- In section 1.1, part 3, you're supposed to do it for both cases: Bernoulli and Multinomial. See updated section 1.3.

2 Collaborative Filtering

Will be released soon.