CS 5350/6350 project: Old Bailey decisions

1 Introduction

The *Old Bailey* is the colloquial name for the *Central Criminal Court of England and Wales*, which deals with with major criminal cases in Greater London, and also sometimes from the rest of England and Wales. This court has existed in some form or another since the 16^{th} century.

The proceedings of this court have been digitized and available online via the *Old Bailey Proceedings Online* project (`https://www.oldbaileyonline.org/`). From the project website:

> The Old Bailey Proceedings Online makes available a fully searchable, digitised collection of all surviving editions of the Old Bailey Proceedings from 1674 to 1913, and of the Ordinary of Newgate's Accounts between 1676 and 1772. It allows access to over 197,000 trials and biographical details of approximately 2,500 men and women executed at Tyburn [...].

2 Task definition

Since all the text of the trials from 1674 to 1913 are available, we can ask the following text classification question: *Can we predict the decision of the court using the transcribed dialogue during a trial?*

The goal of this project is to explore classifiers that predict the outcomes of trials. That is, the instances for classification are the transcripts of trials, and the labels are either *guilty* (denoted by 0) or *not guilty* (denoted by 1).

To this end, you will use various feature representations of the data (including ones you may develop) and the different learning algorithms we see in class. Through the semester, you will be submitting predictions of various classifiers to Kaggle. The mechanics of the project are described in the last section.

3 Data

Each example for clasification is a single trial. In a single trial, there may be more than one defendant, and more than one charge. To simplify things, we have restricted ourselves to trials where there is exactly one defendant and one charge. As a result, each trial has exactly one outcome: either *guilty* (in our case label 0) or *not guilty* (in our case label 1).

3.1 Data splits

We have randomly selected a subset of the entire data (which originally consists of 197,000 examples). In this project, you will be working with 25,000 examples that have been split into three parts described below:

1. `train`: This is the training split of the data, on which you will be training models after hyper-parameter tuning. We have not split the training data into multiple folds for cross-validation; we expect you to do that on your own. The training split consists of 17,500 examples.

2. `test`: This is the test set that you can use to evaluate your models locally. The test set consists of 2,250 examples.

3. `eval`: This is the "evaluation" set with 5,250 examples. We have hidden the labels for these examples. The idea is that you use your models to make a prediction on these examples, and then upload the predictions to Kaggle, where your model's performance will be ranked on a leaderboard. Kaggle uses a random half these examples for a public leaderboard that will be visible to everyone in class, and the other half for a private leaderboard that is only visible to the instructors.

3.2 Feature representations of trials

Instead of having you work with the raw text directly, we have pre-processed the data and a few different feature sets.

1. `bag-of-words`: The bag of words representation represents the text of the trial as a set of its words, with their counts. That is, each dimension (i.e feature) corresponds to a word, and the value of the feature is the number of times the word occurs in the trial. To avoid the dimensionality from becoming too large, we have restricted the features to use the 10,000 most frequent words in the data.

2. `tfidf`: The `tfidf` representation is short for *term frequency–inverse document frequency*, which is a popular document representation that seeks to improve on the bag of words representation by weighting each feature in the `bag-of-words` representation such that frequent words are weighted lower. As with the `bag-of-words`, we use the 10,000 most frequent words in the data.

3. `glove`: The previous two feature representations are extremely sparse vectors, with only a small fraction of the 10,000 features being non-zero for any vector. The `glove` reprsentation represents a document by the average of its "word embeddings", which are vectors that are trained to capture a word's meaning. The word embeddings are dense, 300 dimensional vectors, and for the purpose of this project, each word is weighted by its `tfidf` score.

(The description of the tree feature representations here is deliberately brief. If you would like to know more about these representations, feel free to explore the corresponding Wikipedia articles which do a reasonable job of describing them, or use the office hours to discuss them. For the purpose of this class, you can think of these as three different feature representations of the same data.)

In addition to these features, we have also provide miscellaneous categorical attributes we have extracted from the trial data. This consists of the following features for each trial:

1. defendant age,
2. defendant gender,
3. number of victims,
4. genders of the victims,
5. offence category, and
6. offence subcategory.

You are free to explore using these categorical features to improve, or even build your classifiers. *In fact, you are also welcome to try feature space expansions neural networks, and other non-linear methods.*

3.3 Data files

We have three data splits (`train`, `test`, `eval`), and four different kinds of features (`bag-of-words`, `tfidf`, `glove`, `misc`). Each of these are available in a different file. Other than the `misc` data, all other features are in the libSVM format. The `misc` data is in CSV files.

All the data is available in the `data` directory, organized as follows:

Filename	Feature	Data split
data/bag-of-words/bow.train.libsvm	bag-of-words	train
data/bag-of-words/bow.test.libsvm		test
data/bag-of-words/bow.eval.anon.libsvm		eval
data/tfidf/tfidf.train.libsvm	tfidf	train
data/tfidf/tfidf.test.libsvm		test
data/tfidf/tfidf.eval.anon.libsvm		eval
data/glove/glove.train.libsvm	glove	train
data/glove/glove.test.libsvm		test
data/glove/glove.eval.anon.libsvm		eval
data/misc-attributes/misc-attributes-train.csv	misc	train
data/misc-attributes/misc-attributes-test.csv		test
data/misc-attributes/misc-attributes-eval.csv		eval

In addition, the directory also contains a file called `data/eval.ids`. This file has as many rows as the data.eval.anon file. Each line consists of an example id, that uniquely identifies the evaluation example. The ids from this file will be used to match your uploaded predictions on Kaggle.

4 Evaluation

Since the data is constructed to be balanced, we will use standard accuracy to evaluate classifiers.

The examples are all split randomly among the three splits. So we expect that the cross-validation performance on the training set and the accuracy scores on the test set and the public and private splits of the evaluation set will be similar.

5 Submission format

Kaggle accepts a csv file with your predictions on the examples in the evaluation data. There should be a header line containing `example_id,label`. Each subsequent line should consist of two entries: The example id (from the file `data/eval.ids`) and the prediction (0 or 1).

We have provided two sample solutions for your reference:

1. `sample-submissions/all-positive.csv`: Where all examples are labeled as positive.
2. `sample-submissions/all-negative.csv`: Where all examples are labeled as negative.

6 Project rules

You should work *individually* on the project. You will have to submit *at least* six different non-trivial submissions to Kaggle. Here are the rules for these submissions:

1. You should use at least two feature sets. For example, you could train a classifier on one feature set, and the same classifier on a different one. These would count as different submissions. You are free to use the feature representations provided by us, or construct your own.
2. You should use at least five *different* learning algorithms we see in class. You cannot use any machine learning library for these five algorithms, and instead implement them by yourself.

1. For at most one of your six submissions, you are welcome to use a machine learning library such as PyTorch, TensorFlow or `scikit-learn`.

6.1 Milestones

The project is organized into several milestones summarized below. The deadlines and submissions will be managed via Canvas.

1. **Project information** (10 points): You will need to have registered for the Kaggle competition and made a dummy submission. You should also submit your kaggle user details on Canvas.

2. **Project checkpoint 1** (15 points): For this milestone, you will need to have downloaded the data, and also perhaps run some initial pre-processing on it. You should also have made at least one non-dummy submission on Kaggle. You should submit a one page report on Canvas that describes what you did so far, descriptive statistics about the dataset, and your plan till the next milestone.

3. **Project checkpoint 2** (30 points): This milestone is similar to the previous one. You will need to have made at least two additional submissions to Kaggle. You should submit a one-page report detailing updates after the first milestone, any challenges you have faced, and your plan for the rest of the semester.

4. Final report (45 points): By this time, you should have made at least six non-dummy submissions on Kaggle totally. You should submit a final report of at most six pages that is structured like a small research paper. Broadly speaking it should describe:

   (a) An overview of the project.
   (b) What are the important ideas you explored?
   (c) What ideas from the class did you use?
   (d) What did you learn?
   (e) A summary and discussion of results
   (f) If you had much more time, how would you continue the project?

   Each of these components will be equally weighted in the report grade.