Assignment 2 – Text Classification Using Logistic Regression

CSCI-LING 5832 – Spring 2025

Due 2/23/2025

Disclaimer: All code turned in is expected to be python. You are given the choice of making a script file (*.py) or a jupyter notebook (*.ipynb). Whichever route you choose, please ensure your final submission is clean and well-organized – enough so as to be quickly evaluated by your TAs.

Sentiment Classification

In this assignment, you are tasked with implementing a binary logistic regression classifier for sentiment classification on hotel reviews. The input to the model will be a text review, and the output label is a 1 or 0, corresponding to positive sentiment (i.e., a good review) and negative sentiment (a bad review).

Data

The training data for this task consists of a collection of short hotel reviews. The data is formatted as one review per line. Each line starts with a unique identifier for the review (e.g. ID-2001) followed by a tab and the text of the review. The reviews are not tokenized or sentence segmented. The positive reviews and negative reviews appear in separate files: `hotelPosT-train.txt` and `hotelNegT-train.txt`.

1. We have provided some starter functions in `util.py`; use those to load the data and explore it. What is the distribution of the training samples—how many positive samples and how many negative samples are there?

2. Split the data into training and development sets, so that 80% of the data is in the training set, and 20% is in the development set. (Feel free to use a library such as `sklearn`.)

Evaluation Metrics

We will use precision, recall, and F_1 to measure the output of our classifier. Remember from Chapter 4 of the textbook that accuracy generally isn't used for text classification tasks. Figure 1, reproduced below, visualizes precision and recall metrics next to a confusion matrix. For this assignment, you will also implement F_1-score, which is the harmonic mean of precision and recall:

$$F_1 = \frac{2PR}{P+R} \tag{1}$$

where P = precision and R = recall.

3. Implement precision, recall, and F_1

```
Example:
In: precision(predicted_labels, true_labels)
Out: 1.0
In: recall(predicted_labels, true_labels)
Out: 0.5
In: f1(predicted_labels, true_labels)
Out: 0.667
```

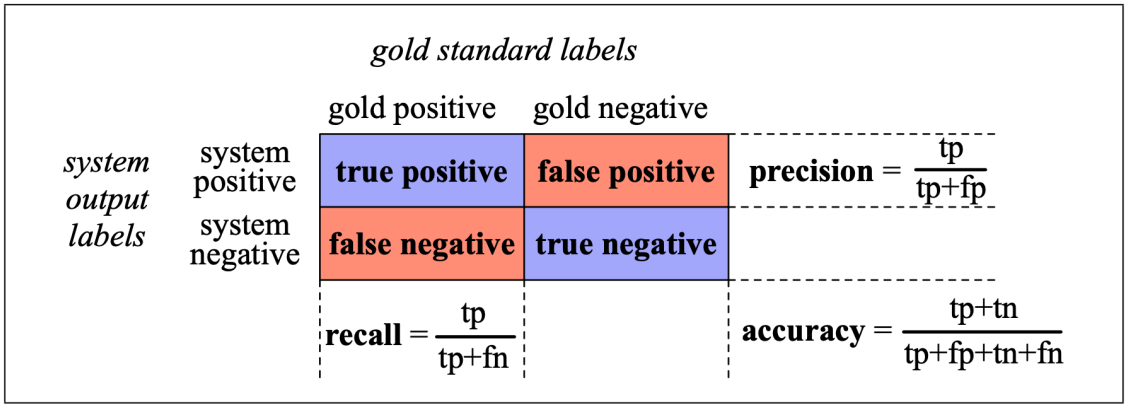

Figure 1: Confusion matrix showing precision, recall, and accuracy.

Logistic Regression on Hand-engineered Features

The input to our logistic regression classifier will be a vector of hand-engineered features of each review, rather than the text of the review itself. The classifier will learn to output 0 or 1 based on these features. For the features, we will refer to the sample given in Section 5.2.1 of the textbook, reproduced in Figure 2 below.

4. Write a function that takes a text snippet and outputs its feature vector, based on Figure 2. Use the files `positive-words.txt` and `negative-words.txt` as the positive and negative lexicon.

```
Example:
In: featurize_text(test_string)
Out: [3.0, 2.0, 1.0, 3.0, 0.0, 4.18965482711792]
```

5. Write a function that takes all the feature vectors in the training set and returns normalized feature vectors based on the below equation, which transforms each feature value so that they lie between 0 and 1:

$$x'_i = \frac{x_i - \min x_i}{\max x_i - \min x_i} \tag{2}$$

where $i \in [1, 6]$ so that each feature vector $[x_1, x_2, ..., x_6]$ is transformed into $[x'_1, x'_2, ..., x'_6]$ based on the maximum and minimum for each feature value x_i across the entire dataset.

```
Example:
In: normalize(all_feature_vectors)
Out: [[0.3, 0.6, 0.9, 0.06, 0.0, 0.7], ...]
```

Defining and training the classifier

Now that we have defined functions for processing input data and scoring model output, it's time to implement our classifier. We recommend referencing `A2sample.py` which contains some scaffolding for a simple PyTorch logistic regression classifier, as well as a simple training loop.

6. Train a model for 100 epochs with a batch size of 16. Record the loss on the training set and the development set for each epoch. If the training loss decreases, the gradient descent is working as it should. If the development loss decreases, the learning is working as it should—it is generalizing well enough that it can slowly classify more and more unseen examples correctly.

7. Calculate the precision, recall, and F_1 on the entirety of the development set using your trained model. Remember that model output is probability, which should be converted to a binary 0/1 label for our metric functions.

```
Example:
dev_probs = model(all_dev_feature_vectors)
predicted_dev_labels = [model.logprob2label(p) for p in dev_probs]
f1(predicted_dev_labels, true_dev_labels)
```

Var	Definition	Value in Fig. 5.2
x_1	count(positive lexicon words $\in$ doc)	3
x_2	count(negative lexicon words $\in$ doc)	2
x_3	$\begin{cases} 1 & \text{if "no"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	1
x_4	count(1st and 2nd pronouns $\in$ doc)	3
x_5	$\begin{cases} 1 & \text{if "!"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	0
x_6	ln(word count of doc)	$\ln(66) = 4.19$

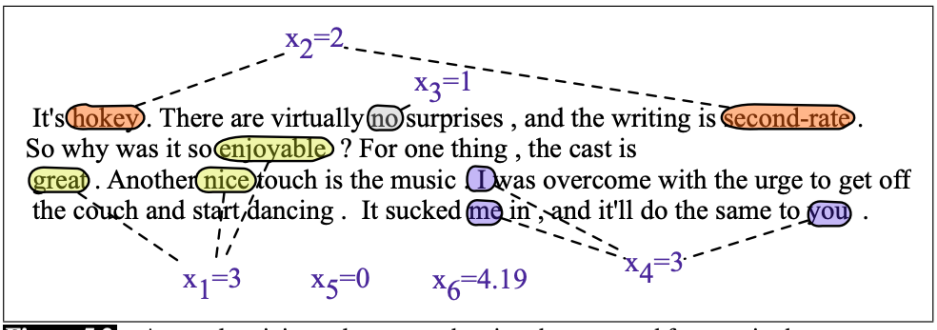

Figure 5.2 A sample mini test document showing the extracted features in the vector x.

Figure 2: Instructions for a 6-dimensional feature vector $[x_1, x_2, ..., x_6]$ for sentiment classification.

8. Experiment with different settings: for example, switch up the number of epochs, batch size, or learning rate. Run your best-performing model (i.e. the model with the highest F_1 on the development set) on the test data.

Written Report

The last item you will be required to turn in is a written report (minimum 1 page) answering the following questions about this assignment and the concepts it is meant to cover.

1. Describe the details for your best performing model: parameters such as number of epochs, batch size, and learning rate. Compare and contrast with other parameter settings you used—feel free to use graphs or tables to compare. Why do you think it performed better than the other models?

2. What do you think would happen if you didn't normalize the feature vectors? Write down a guess for what you think would happen, and then run an experiment to test your intuitions and report back what you learned.

3. What would happen if you removed one of the features entirely and used a 5-dimensional feature vector? Choose one feature and remove it from your vector. Then, run another experiment and see what happens. Does the test F_1 go up or down? Does the model converge slower, or faster? Report which feature you removed and what you learned.

4. Review Section 4.10, p. 18 of the textbook and then consider the resources we used for this task: for instance, the training data and the positive and negative lexicons. Did you notice any biases present in these resources? Can you think of any harms or unintended consequences (harmful or not) that this classifier could cause? There is no correct answer; just write a couple of sentences reflecting on this prompt.

Rubric

Base Points	Item
10	Data loading and splitting
10	Evaluation metrics
20	Featurizing and normalizing data
10	Train loss decreases
10	Dev loss decreases
20	Ran metrics on dev set and test set
20	Written Report
Point Multipliers	**Criteria**
1	All expected functionality is present. Report answers questions in a near-perfect manner.
0.75	Minor errors not impacting general functionality. Output may slightly deviate from what is expected. Report answers questions well with minor errors.
0.5	Errors demonstrating a lack of understanding. Some functionality missing. Report demonstrates lack of understanding.
0.25	Significant errors affecting the functionality of the item. Report has significant errors and does not answer questions in an acceptable manner.
0	No work present or code does not run. No report.