Programming Assignment 2: Semi-supervised Text Classification

CSE 156: Statistical NLP: Spring 2019

University of California, San Diego

Due: April 29, 2019

In this programming assignment, you will build a `semi-supervised` text classifier. Many real-world applications contain a small number of labeled data points (instances) but a large number of unlabeled ones. Machine learning algorithms can utilize unlabeled data are called semi-supervised.

The particular type of text classification to be performed for this homework is `sentiment classification`. The objective is to classify the sentiment of user reviews into `negative` and `positive` reviews, and to make the task more interesting (i.e. difficult), we have split the reviews into segments of around 140 characters. Notice that this might make it prohibitively difficult or impossible detect the sentiment of some segments – this can be ignored for our purposes.

1 Preliminaries

1.1 Data and Code

The data you will use is in the tarball `sentiment.tar.gz`, which contains the following files:

- `train.tsv:` Reviews and their associated labels to be used for training.
- `dev.tsv:` Reviews and their associated labels, to be used for development (do not use for training).
- `unlabeled.tsv:` Reviews without labels, to be used for semi-supervised training

We have made some initial code available for you to develop your submission. The file `classify.py` contains a basic logistic regression classifier from `sklearn`.
The file `sentiment.py` contains methods for loading the dataset (without untarring the data file), creating basic features, and calling the methods to train and evaluate the classifier. It also contains the code to output the submission file for Kaggle from a classifier (`sentiment-pred.csv`) and the code that was used to generate the solution file, and a basic benchmark (`sentiment-basic.csv`).

You will use Kaggle to keep track of the performance of your models. The test data is only available on Kaggle, you will have access to the test data indirectly by seeing how your methods perform. Follow the link below to make submissions on Kaggle.
`https://www.kaggle.com/t/db2e92def4d544daa8d96b8faa7ca46f`

2 Supervised: Improve the Basic Classifier (50%)

As described above, we have included a basic logistic regression classifier and included its predictions as a benchmark on Kaggle. This classifier uses the default hyperparameters for regularization and optimization from *sklearn* and uses the basic *CountVectorizer* to create the features.
Your task is to familiarize yourself with the data and the classification task by improving this supervised text classifier. There are a number of different things you can try, such as hyperparameter search (using accuracy on the development data), different tokenization, adding/removing features, TF-IDF weighting (using only the training data), and so on. The primary guideline here is to utilize only your intuitions and the training data, and the performance on the development data, in order to make these modifications. Use of unlabeled data or any other external

resource is not allowed.

You are not allowed to use a different classifier other than logistic regression (and as much as possible, use the scikit-learn implementation). The aim of this exercise is to train you to do feature engineering and semi-supervised learning algorithms, not from changing the underlying classifier.

In the writeup, describe the changes, your reasoning behind them, and the results. You should use figures, examples, tables, and graphs to illustrate your ideas and results, for example plotting accuracy as the regularization strength is varied. Submit the relevant code in a folder called *sup*. The Kaggle submission description should start with"Supervised:" (you are free to add whatever else afterwards).

3 Semi-supervised: Exploiting the Unlabeled Data

In the second part of this homework, you will focus on improving your supervised classifier by leveraging a large collection of unlabeled reviews that are available to you. There are many different approaches to do this, we will describe two options below. You are free to choose your favorite. Note that semi-supervised learning is tricky, and its not always easy to get a substantially better accuracy than the supervised classifier. *Do what you can.*

As with the previous submission, you have to describe the approach that you are using, and provide the plots, tables, and graphs to analyze and discuss the results in the write-up. For example, a useful plot for you to include might be the performance of your classifier (on development data) as the amount of unlabeled data is varied, i.e. 0% (same as supervised), 10%,20%,...,100%. Part of the analysis should also include some error analysis and examples of the highest and lowest weighted features for each of the labels. You should also refer to the relevant readings and published papers, and cite them as appropriate, when describing your approach. Include the relevant portions of your code that your approach in a folder called *semisup*. The description for the submission on Kaggle should start with "Semi-supervised:" (you are free to add whatever else afterwards).

3.1 Expanding the Labeled Data (50%)

One popular approach to semi-supervised learning is to iteratively expand the labeled data (by using the classifier to predict on the unlabeled data) and retrain the classifier. The following pseudo-code illustrates this idea.

require D_u (unlabeled data), D_l (labeled data)
  $\hat{D}_l \leftarrow D_l$
  loop
    $\theta \leftarrow \text{TRAIN}(\hat{D}_l)$
    $\hat{D}_u \leftarrow \text{PREDICT}(D_u, \theta)$
    $\hat{D}_l \leftarrow \text{EXPAND}(D_u)$
    if STOP(_) **then return** θ
    end if
  end loop

There are many variations of this algorithm, known as self-training, that you can use. The first choice is to decide which labels to include in $\hat{D}_l$ in every iteration: every prediction? most "confident" predictions? predictions with a "soft" label (kind of like EM)? only points that an ensemble of simple classifiers agree on? nearest neighbors of previously labeled points? or something else? The second choice is to decide the stopping criterion, should it be a fixed number

of iterations (determined using development data)? should it be when the set of labels stop changing (or only a small proportion changes)? or something else?

When analyzing this approach, we expect you to include the accuracy and the size of the labeled set (if appropriate) as they vary across iterations. It may also be useful to identify a few features/words whose weight changed significantly, and hypothesize why that might have happened.

3.2 Bonus: Designing Better Features (+10%)

The primary concern when using a small set of labeled data for training text classifiers is the sparsity of the vocabulary in the training data; irrelevant words might look incredibly discriminative for a label, while relevant words may not even appear in the training data, because of small sample and high dimensional statistics. A way to counteract this is to utilize the corpus of unlabeled text to learn something about the word semantics, and use it to identify the words that are likely to indicate the same sentiment. In order words, knowledge from unlabeled documents can allow us to spread the labels to words that do not even appear in the training data.

We are being vague here in order to not provide a specific solution. If you will explore this direction, you will have to consider how to represent the word contexts, how to represent/encode the notion of similar words (fixed or hierarchical clusters? low-dimensional embeddings?) and so on.
Your analysis should include why you picked a certain strategy (why you thought it would be a good idea). You should also include examples of words and/or reviews where the propagation of information helped (where it worked) and hurt.

4 Submission Instructions

Submit your work on Gradescope.

- **Code:** You will submit your code (the folders **sup** and **semisup**) together with a neatly written README file to instruct how to run your code with different settings. We assume that you always follow good practice of coding (commenting, structuring), and these factors are not central to your grade.
- **Report:** (use the filename A2.pdf): your writeup should be **four pages long, or less, in pdf** (reasonable font sizes). **You should have three separate sections in your report, the first on supervised learning, the second on semi-supervised learning**. Additionally, under a third section called "Kaggle", please include your **Kaggle user name, Kaggle display name, and email**. This will allow us to lookup your performance on the test data, and verify that it matches what is in the report.

  Part of the training we aim to give you in this class includes practice with technical writing. Organize your report as neatly as possible, and articulate your thoughts as clearly as possible. We prefer quality over quantity. Do not flood the report with tangential information such as low-level documentation of your code that belongs in code comments or the README. Similarly, when discussing the experimental results, do not copy and paste the entire system output directly to the report. Instead, create tables and figures to organize the experimental results.

5 Potential FAQ

Question 1: **Is the grade based on the performance of my submissions relative to the others in the class?** No, you will be evaluated on the quality and creativity of your

write up, but it is expected that your submissions should outperform the simple baseline we have provided. If you do get unsatisfactory results, for example your semi-supervised approach performs worse than the supervised classifier, we would expect a discussion and analysis in the writeup (and you can get full credit).

Question 2: Then why have the class leaderboard? The leaderboard is provided for you to be able to evaluate how good your results are relative to others. Some might find it useful. Similar settings are found in academia, where progress is evaluated performance on standardized datasets, for example ImageNet in Computer Vision.

Question 3: I do not find this leaderboard motivating, will there be other assignments with a leaderboard? No,this will be the only assignment with a leaderboard.

Question 4: This assignment seems too open-ended, and I can see spending many whole days improving my classifier. How will I know I am done? The short, but perhaps unsatisfactory, answer is: when you feel you have put an adequate effort. For each part, this would mean you decided on one non-trivial approach you would like to use, implemented it, analyzed the results, and presented the approach in insightful way. There is going to be diminishing returns in accuracy with the amount of effort you put in, we would discourage solely using the accuracy gain in deciding when you are done.

Question 5: I am not sure what is a valid external resource that I can use, and what is not? External resource are not allowed. Using additional labeled text from elsewhere will be considered cheating.

Question 6: My code is taking too long to run. What should I do? By most standards, this is a small dataset. If you are facing efficiency issues, you may need to analyze your code for bottlenecks and address them, for example caching feature calculations if runtime feature computations are taking too long. However, we expect you to waste too much time optimizing your code; if you are struggling, it would be better to randomly sub-sample the unlabeled documents

6 Acknowledgments

This assignment is adapted with some changes from similar assignments made available by Noah A. Smith, and Sameer Singh.