CS6320 Assignment 1

Due: Sep 26, 2022, 11:59pm

Groups Groups are specified on eLearning and fixed (make sure you know your group name/number and members). You will submit reports as the same group to Gradescope. You should include (1) your group name which is specified on eLearning (e.g. "Assignment Groups01") on the top of the report; (2) each group member's name and netID. If we identify cases of students that do not contribute equally to the group's work, we will be penalizing the assignment.

1 Overall Goal

You will build an n-gram-based language model (with variations) and use it as a classifier for the Opinion Spam Classification task. You will follow the steps towards building the language model and testing.

Requirements&Advices (1) **The report is important!** The report is where you get to show that you understand not only *what* you are doing but also why and *how* you are doing it (e.g. data structure). So be clear, organized and concise; avoid vagueness and excess verbiage. Spend time doing error analysis for the model(s). This is how you understand the advantages and drawbacks of the systems you build. Using the LaTeX template below is recommended (you can also use Microsoft Word or Googld Doc if preferred, the final report should be a PDF); (2) You shall not share or make public any materials of the assignments.

Report Template: https://www.overleaf.com/read/djxcxcrrhstc

2 Dataset

You are given (via eLearning) an adapted version of the Deceptive Opinion Spam Corpus [1] which consists of truthful and deceptive hotel reviews of Chicago hotels [1]. Here is an example of a TRUTHFUL review:

> After Leaving some important documents in the room, I called and asked for the lost and found department... SEVEN TIMES over the course of a week. Eventually I had enough and asked for a manager and was put on hold then disconnected. Finally , a deceivingly friendly operator PROMISED me she would have someone call me back in a few minutes, It never happened. So Avoid this place because after they rip you off once they'll keep doing it later. They cannot be trusted at their word.

and a DECEPTIVE review from the training data:

> My husband and I stayed at the Omni after attending a wedding that took place there. We were delighted at the luxury of the rooms and the accommodation were wonderful. Everyone from the concierge to the housekeepers were friendly and professional. We were extremely pleased with the whole experience and look forward to our next trip to Chicago so we can stay there again. And, by the way, the wedding was absolutely gorgeous!

In the A1_DATASET folder, you will find two folders (i.e. `train`, `validation`) that contain the corpus for truthful and deceptive hotel reviews respectively. The `test` folder contains test data (reviews) and labels (truthful: 0, deceptive: 1). Note that **each line** in these files corresponds to **a single review**.

The project will proceed generally as follows in terms of system/code development, we require that the coding language to be *python*.

1. Write code to train unsmoothed unigram and bigram language models for an arbitrary corpus. We will use the provided corpora of train.txt (containing truthful and deceptive reviews) (see Section 3).
2. Implement smoothing and unknown word handling (see Section 4).
3. Implement the Perplexity calculation (see Section 5).

[1] You can find more information about the corpus on https://myleott.com/op-spam.

4. Using 1, 2 and 3, together with the provided training and validation sets, develop a language-model-based approach for Opinion Spam Detection (see Section 6).

3 Unsmoothed n-grams

To start, you will write a program that computes unsmoothed unigram and bigram probabilities from the training corpus. You should consider truthful and deceptive reviews as separate corpora and **generate a separate language model for each set of reviews**.

You are given a tokenized opinion spam corpus as input (see Section 2). You may want to do additional preprocessing, based on the design decisions you make [2]. Make sure to explain preprocessing decisions you make clearly in the report. Note that you may use existing tools just for the purpose of preprocessing but you must write the code for gathering n-gram counts and computing n-gram probabilities yourself. For example, consider the simple corpus consisting of the sole sentence:

the students like the assignment

Part of what your program would compute for a unigram and bigram model, for example, would be the following:

$$P(the) = 0.4,\ P(like) = 0.2$$
$$P(the|like) = 1.0,\ P(students|the) = 0.5$$

Preprocessing The `truthful.txt`, `deceptive.txt` and `test.txt` files included are already tokenized and hence it should be straightforward to obtain the tokens by using space as the delimiter. Feel free to do any other preprocessing that you might think is important for this corpus. Do not forget to describe and explain your pre-processing choices in your report.

4 Smoothing and unknown words

Firstly, you should implement (>=1) method to handle unknown words. Then You will need to implement (>=2) smoothing methods (e.g. Laplace, Add-k smoothing with different k). Teams can choose any method(s) that they want for each. **The report should make clear what methods were selected**, providing a description for any non-standard approach (e.g., an approach that was not covered in class or in the class).

5 Perplexity on Validation Set

Implement code to compute the perplexity of a "development set." ("development set" is just another way to refer to the validation set — part of a dataset that is distinct from the training portion.) Compute and report the perplexity of your model (with variations) on the it. Compute perplexity as follows:

$$PP = (\prod_{i=1}^{N} \frac{1}{P(w_i|w_{i-1},...,w_{i-n+1})})^{1/N}$$
$$= exp\ \frac{1}{N}\sum_{i=1}^{N} -log\ P(w_i|w_{i-1},...,w_{i-n+1})$$

where N is the total number of tokens in the test corpus and $P(w_i|w_{i-1},...,w_{i-n+1})$ is the n-gram probability of your model. Under the second definition above, perplexity is a function of the average (per-word) log probability: use this to avoid numerical computation errors.

As you experimented with more than one type of smoothing and unknown word handling (Sec 6), you should report and compare the perplexity results of experiments among some of them.

[2] you can refer to text tokenization and normalization mentioned in Sec 2.4 of J&M https://web.stanford.edu/~jurafsky/slp3/2.pdf

6 Language Model based Opinion Spam Classification

There are potentially many ways to use language models as predictors. The standard way to do so for our task is simply to measure the perplexity of the "truthful" vs. the "deceptive" language models on a given review: return the class (truthful, deceptive) associated with the model that produces the lower perplexity score. We don't recommend that you try fancy or advanced ideas without first implementing and evaluating the simple, straightforward one described above.

Methodology suggestion You are given plenty of labeled truthful and deceptive hotel reviews. To adjust your models for best performance/accuracy (e.g. tuning smoothing parameters) you should use the provided development sets to evaluate the performance of your model. The folder structure indicates the `train` and `test` sets as well as the classification labels (e.g. all the excerpts in `train/truthful.txt` are truthful reviews). You should use only the training data and validation data for developing your model.

Final Evaluation Generate your predictions for the excerpts in the test data (`./test/test.txt`) – the prediction values are integers using the following encoding: truthful: 0; deceptive: 1. Calculate the accuracy of your best model (developped on validation set) on the test set by comparing you predictions with `./test/test_labels.txt`, and report the accuracy results (the number of correct predictions divided by the number of excerpts in the test set).

7 Grading Rubrics

- Unigram and bigram probability computation (15%)
- Smoothing (15%)
- Unknown word handling (15%)
- Implementation of perplexity (15%)
- Opinion spam classification with language models (15%)
- Quality and clarity of the report (25%)
  - Group name, student names and netids of group members
  - Clear description of implementation details (10%)
  - Eval, Analysis and Findings – including but may not be limited to: (1) Report test set performance/accuracy in final evaluation; (2) how the smooothing strategy affects the performance on validation set; (3) Error analysis and comparison of language model based classifiers with specific examples, etc. (10%)
  - Details of programming library usage if any (3%)
  - Brief description of the contributions of each group member (1%)
  - Feedback for the project (1%) – too difficult/easy? how much time did you spend on it? does the project help with your understanding of the content? any other suggestions?

8 Report

You should submit a short document (approximately or fewer than 6 pages) that describes your work. Your report should contain a section for each of the Sections above as well as a short workflow section that explains *how* you divided up the work for the project among group members. Important: the role of this document is to show us that you understand and are able to communicate what you did, and how and why you did it. Come to ask the TA if you are not sure how to answer either of these questions!

You might think this means you must write a lot to explain something, it may be a bad sign. Describe the general approach, the data structures used (where relevant). Use examples, identify all design choices and justify them (e.g., how did you deal with unknown words? how did you perform smoothing?). Wherever possible, quantify your performance with evaluation metrics (e.g., report the classification performance on the development set). If you tried any extensions, perform measurable experiments to show whether or not your extensions had the desired effect and why?

Code Please include snippets of your code in the report, if you think it makes things clearer. Include only relevant portions of the code (such as a few important lines from a function that accomplish the task that you are describing a nearby paragraph, to help us understand your implementation). Including boilerplate code or tedious, unnecessary blocks (e.g., reading files) only makes your report cluttered and hard to follow, so avoid it.

9 What to Submit

- Submit your report to Gradescope (resubmissions to Gradescope REMOVES the Group structure. So be sure to re-create (add your members) your Group with every resubmission to Gradescope.)
- Submit your code on eLearning as a .zip or .tar file.

References

[1] Myle Ott, Yejin Choi, Claire Cardie, and Jeffrey T. Hancock. Finding deceptive opinion spam by any stretch of the imagination. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 309–319, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.