CS4740: Intro to NLP
Project1: Language Modeling and Opinion Spam Classification

Due **Friday, Sept 27, 11:59PM**

1 Overall Goal

In this project we will build an **n-gram-based language model** and also investigate a feature-based Naive Bayes model for the **Opinion Spam Classification** task.

The project is divided into parts. First, we will get started with the steps towards building the language model using a **Deceptive Opinion Spam Corpus**. After that, you will (a) use this language model to classify hotel reviews as truthful or deceptive and (b) build a feature-based Naive Bayes model to perform the same classification task.

Logistics: You should work in groups of 2 or 3 students. Students in the same group will get the same grade. Thus, you should make sure that everyone in your group contributes to the project. Also, **remember to form groups on BOTH CMS and Gradescope** or not all group members will receive grades!!! You can use the Teammate Search facility in Piazza for finding your project partners.

Advice: 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. So be clear, organized and concise; avoid vagueness and excess verbiage. Spend time doing error analysis for the models. This is how you understand the advantages and drawbacks of the systems you build. The reports should read like a research paper, not a lab report or list of facts or specifications.

2 Dataset

You are given (via Piazza) a **Deceptive Opinion Spam Corpus**[1] which consists of truthful and deceptive hotel reviews of 20 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

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

> 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 accomodations 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 DATASET folder, you will find two folders that contain the training corpus for truthful and deceptive hotel reviews, respectively. Note that **each line** in these files **corresponds to a single review**.

As outlined above, your ultimate goal for the project is to **compare the performance of a language-modeling vs. a classification-based approach to predicting, for a given review, whether it is truthful or deceptive**. We hope that you find this task interesting! It is definitely a challenging task and it can be difficult for even humans to discriminate between truthful and deceptive reviews. Hopefully, we will make some progress in this project to investigate which models perform better for this task. If you want to better understand how this dataset was constructed and alternative methods for the Opinion Spam Detection task, we recommend you to read the paper [1].

The project will proceed generally as follows in terms of system/code development:

1. Write code to train unsmoothed unigram and bigram language models for an arbitrary corpus. (We will use the provided corpora of truthful reviews and deceptive reviews. (see Section 3)

2. Implement smoothing and unknown word handling. (see Section 4)

3. Implement the Perplexity calculation. (see Section 5)

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 6.1)

5. Apply your best language-model-based opinion spam classifier (from 4) to the provided test set. Submit the results to the on-line Kaggle competition. (see Section 6.2)

6. Use any existing implementation of Naive Bayes (and the training and validation sets) to create an opinion spam classifier. Apply your best NB classifier to the provided test set. Submit the results to the separate Kaggle competition (for NB classifiers). (see Section 6.3)

3 Unsmoothed n-grams

To start, you will write a program that computes **unsmoothed unigram and bigram probabilities** from an arbitrary corpus. You can use any programming language(s) you like. 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 2.1). You may want to do additional preprocessing, based on the design decisions you make. (See lecture slides and readings on text tokenization and normalization.) 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 liked the assignment

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

$$
\begin{aligned}
&P(the) = 0.4\\
&P(liked) = 0.2\\
&P(the|liked) = 1.0\\
&P(students|the) = 0.5
\end{aligned}
$$

3.1 Preprocessing

The `truthful.txt` and `deceptive.txt` files included in each of the subfolders in the DATASET folder 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

For the deceptive opinion classification task (see section 6), you will need to implement smoothing and and a method to handle unknown words in the test data. 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 readings. You should use the provided validation sets to experiment with different smoothing/unknown word handling methods if you wish to see which one is more effective for this task.

5 Perplexity

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 and the test portion.) **Compute and report the perplexity of each of the language models (one trained on truthful reviews and deceptive reviews) on the `development corpora`.** Compute perplexity as follows:

$$PP = \left(\prod_{i}^{N} \frac{1}{P(W_i|W_{i-1}, \ldots, W_{i-n+1})} \right)^{\frac{1}{N}}$$
$$= \exp \frac{1}{N} \sum_{i}^{N} - \log P(W_i|W_{i-1}, \ldots, W_{i-n+1})$$

where N is the total number of tokens in the test corpus and P($W_i|W_{i-1}, \ldots, W_{i-n+1}$) 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.

If you experimented with more than one type of smoothing or unknown word handling, it would be a good idea to report, compare and discuss the results of experiments with each.

6 Opinion Spam Classification

In this part of the project, you will use a language model based method and Naive Bayes to automatically predict whether a given hotel review is truthful or deceptive.

6.1 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. You may explore other ways as well. Regardless, **make sure to clearly explain your method, and why you chose it, in the report**. We don’t recommend that you try fancy or advanced ideas without first implementing and evaluating the simple, straightforward one described above. (Good life advice in general!) As in the Perplexity section, **if you experimented with more than one classification approach, it would be a good idea to report, compare and discuss the results of experiments with each.**

Methodology suggestion. You are given plenty of labeled truthful and deceptive hotel reviews. To adjust your models for best performance (e.g. tuning smoothing parameters, or choosing between design options) 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. **Submit your predictions for the excerpts in the test data on Kaggle.** See Section 6.2 below for the format. Keep in mind that the evaluation metric used on Kaggle is accuracy — the number of correct predictions divided by the number of excerpts in the test set.

6.2 Kaggle submission format

Apart from your report, you will submit a **.csv** file to Kaggle. There should be one prediction on each line for each speech inside the test folder. Each line should contain an Id (the line number of the review in the test set) and the category predicted by your algorithm for the review, separated by a comma. The prediction values are integers using the following encoding: truthful: 0; deceptive: 1. So your output file should look like:

```
Id,Prediction
0,0
1,0
2,1
...
```

6.3 Feature-based Naive Bayes for Opinion Spam Classification

As an alternative to the above, in this part, you will use Naive Bayes method to classify truthful vs. deceptive hotel reviews.

One of the most important steps of building a classifier is to think about what kind of features would be important to extract from the text for the given task. One simplest yet effective way of representing text is by using **Bag of Words** or **Bag of n-grams** representation which represents the occurrence of words within a document. Note that this representation is called "bag" of words since it omits information about the order of the words in the document. We expect you implement and experiment with various features for this classification task. Note that you **have to** experiment with at least **unigram and higher degree n-gram features** to see how they perform for this task.

Implementation Details. You can use existing libraries for the Naive Bayes classifier implementation and feature extraction. However, you have to clearly report for which parts of the project you implemented yourself vs. used a library and include the name of the libraries you experimented with.

Suggestions for the implementation. You can consider using `scikit-learn` library for the implementation of Naive Bayes classifier[2] and feature extraction[3].

For submission, **run your model on the files in the test folder and submit your predictions on Kaggle**. Make sure to include your group name for Kaggle submission as well as the screen shot of the Kaggle score in your report. Otherwise, you will lose the points for Kaggle submission.

[2] https://scikit-learn.org/stable/modules/naive_bayes.html

[3] https://scikit-learn.org/stable/modules/feature_extraction.html

7 Grading rubric

- Unigram and bigram probability computation (10%)
- Smoothing (10%)
- Unknown word handling (10%)
- Implementation of perplexity (10%)
- Opinion spam classification with language models (10%)
- Opinion spam classification with Naive Bayes (15%)
  - Incorporating Naive Bayes model (5%)
  - Incorporating bag of words and ngram features (5%)
  - Implementation of additional linguistic features (5%)
- Model and feature selection using validation corpus[4] (5%)
- Quality and clarity of the report (25%)
  - Names and netids of group members
  - Kaggle group name!!!!
  - Clear description of implementation details (8%)
  - Motivation for feature choices and showing which features are important to discriminate between truthful vs. deceptive reviews for Naive Bayes classifier (6%)
  - Error analysis and comparison of language model based classifier with Naive Bayes, with specific examples (6%)
  - Details of programming library usage (3%)
  - Brief description of the contributions of each group member (1%)
  - Feedback for the project (1%)
    - did you find the project too difficult/easy? how much time did you spend on it? does the project help with your understanding of the content? any other suggestions?
- Submission to Kaggle (5%)
  - You will not be given any points if (i) more than one member of your group makes submissions to Kaggle OR (ii) your report does not include your Kaggle group name or screen shot for the score.

[4] Note that we want you to think about modeling decisions (e.g. hyperparameters, features etc.) that you can optimize using the validation corpus for both language model based and Naive Bayes classification methods. Please explain modeling choices and how you use the validation corpus to make modeling decisions clearly in your report.

8 What to submit

You must create your Group on both CMS and Gradescope. Note that, at least as of last year, resubmissions to Gradescope REMOVED the Group structure. So be sure to re-create your Group with every resubmission to Gradescope.

- Submit your **report** to Gradescope.
- Submit your **code** on CMS as a .zip or .tar file.
- We will have separate Kaggle challenges for the language modelling vs. Naive Bayes opinion spam classification. We will notify you about the submission link later via Piazza. Note that in order to be graded as a group, **you must form groups with your partner(s) on Kaggle.** You will have a limited number of Kaggle submissions per group — 5 predictions daily. If you do not submit your prediction as a group, your Kaggle submission will be invalid.

9 The report

You should submit a short document (approximately 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 see us if you're 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 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? If something doesn't work or performs surprisingly, try to look deeper and explain why.

Code. You may 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 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.

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 - Volume 1, HLT ’11, pages 309–319, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.