Homework 1: Models and an Annotation Experiment

Due Feb 19 before class
This version: Feb 4, 3pm

CS 685, UMass Amherst, Spring 2020

This is a two-part assignment. The first part asks math questions about linear and neural net models, to build your intuition about how they work. The second part consists of an annotation experiment where, given a set of tweets, your task is to design an annotation experiment, collect annotations, and analyze and model the results. **Important: You will need to submit your peer-annotated tweets as a CSV to Moodle, 5 DAYS PRIOR TO THE DEADLINE, i.e. Feb 14 by midnight** (see timeline information there). Your PDF report and code will go on Gradescope. See the "deliverables" paragraph in the Annotation Overview section below.

1 Model Math Questions

Answer the questions below to the best of your ability, and show your work.

1.1 MLE for Naive Bayes

[INLP ch. 2, #3.] Derive the maximum-likelihood estimate for the parameter μ in Naive Bayes. This will require applying Lagrange multipliers, as mentioned in the chapter and detailed more in the appendix.

1.2 LogReg weights on data union

[INLP ch. 2, #5.]

1.3 Softmax and Sigmoid

[INLP ch. 3, #2.] Prove that softmax and sigmoid functions are equivalent when the number of possible labels is 2. Prove that both functions calculate the same values, i.e. for any weight matrix A, show how to construct a weight vector θ such that

$$Softmax(Az)[0] = \sigma(\theta \cdot z)$$

1.4 Class-indicative words under an averaged embedding classifier

Consider BOW logistic regression with vocabulary size V and K output classes, where x_w is the count of word w in the document, and the feature function is the wordcount-based cross-product we discussed in lecture and that is in your reading (here we are using indicator function notation):

$$f_{k,w}(x, y) = x_w \; 1\{y = k\}$$

We think of the function f as returning a vector (to dot product it against the weight vector β), though for convenience we subscript f and β with two variables. Therefore, the parameter weight $\beta_{k,w}$ denotes how much the (unnormalized) log probability of class k goes up, for each instance of the word w in the text.

One way to get insight into a model, is to ask it what words it thinks are the best statistical predictors for a class. Using a BOW logreg, one way we can do this is to look at the 10 words with highest $\beta_{w,k}$ weights for a particular output class k. This tells us what words that, according to the model, give the greatest increase in class k probability, when a single token of that word appears.

Now consider an averaged embedding classifier like we saw in lecture, where for each word type, there's a word vector $a_w \in \mathbb{R}^D$. The document representation x is calculated by averaging word vectors (a.k.a. embeddings) from the n tokens in the document,

$$x = \frac{1}{n} \sum_{i}^{n} a_{w_i}$$

and then used as the representation for multiclass logistic regression, where the parameter weights are $\beta \in \mathbb{R}^{DK}$.

For an output class k, we'd like to derive the top-10 most highly indicative words for k. Define this as before—which words, for each time they appear, most increase the probability of class k? For the BOW logreg, we got this by ranking words by $\beta_{k,w}$. What should we rank by for this new classifier, to get the analogous result? Please derive this formula and explain it.

Overview: Annotation and Modeling Experiment

You will be given a sample of 250 tweets, and then conduct an annotation task, from start to finish:

1. Task design: Come up with a sentence-level (tweet-level) classification NLP task for your tweets. Some examples include, but are not limited to: sentiment analysis, affect intensity, topic classification, etc. Write up annotator guidelines/instructions.
2. On your personal dataset, collect annotations from two classmates for your task. This must be done by Feb 14! In order to get everything done in time, we strongly suggest the following internal deadlines:
    - Feb 10: Researcher has found their annotators by now. (Maybe in Feb 5 lecture in person? Or on Piazza?)
    - Feb 10: Researcher sends annotation guidelines and spreadsheets to annotators.
    - Feb 13: Annotators hand off their completed spreadsheets back to researcher.
    - Feb 14: Researcher double checks basic data cleanliness (are all examples annotated? are all labels allowable ones?) and submits to Moodle.
3. Collect feedback from annotators about the task including annotation time and obstacles encountered.
4. Calculate inter-annotator agreement and other related statistics.
5. Aggregate results to create final dataset.
6. Perform NLP experiments on your new dataset! (Problem 3.)

Many more details on these steps are spelled out in the next section for Problem 2.

Roles: Every student in the class will take on two roles: **researcher** and **annotator**. As a **researcher**, you perform the above steps on your tweets. At the same time, you will also serve as an **annotator** for your fellow students.

Tasks: Every researcher will have a different set of tweets, and will define their *own* unique task. Therefore the researcher will have to draw up guidelines and explanations to allow their annotators to do a good job at reliably identifying whatever categories the researcher has a goal of collecting.

Annotator etiquette: Everyone must collect annotations from at least two people. Therefore everyone should be ready to annotate for two other people. If someone asks you to annotate for them, please accept, unless you have accepted more than two annotation jobs already! You should try to do a good job doing annotations for your fellow students.

It's OK if this is hard: Designing an annotation task, and collecting annotations, is tricky. This assignment is small-scale as far as annotation projects go, but hopefully it illustrates the challenges that underlie it.

Datasets. We have collected samples of publicly available English tweets for all your experiments. They are available at: `http://hobbes.cs.umass.edu/~brenocon/anlp2020/hw1/`

- Personal250: A dataset of 250 tweets we gathered just for you. Every student, in their researcher role, has a *different* set of 250 tweets. You can look at your own as much as you like.

- CommonGround: The `Sample_1000` version is a dataset of tweets you can just look at, or use in examples for your annotation guidelines. If you want to play around with lots of unlabeled data, we also provide versions with 100,000 and 10 million tweets, respectively.

The format is, the text of one tweet is on each line. The original tweets sometimes had tab or newline characters; in our preprocessing we normalized all those to spaces. We did no other preprocessing. The text is encoded as UTF-8. Your web browser may not view it correctly; to view, we recommend using a smart text editor that lets you explicitly set the encoding (e.g. VS Code, Vim, Emacs, etc.), and when you view on the terminal, be aware your terminal software may not handle it either. In Python 3, you should be able to read the data correctly using the `encoding='utf8'` option for `open()` (see online docs). Finally, note the small CommonGround 1000 sample is deduplicated, but the larger CommonGround files are not. All tweets are from July 8 and 9, 2019.

2 Details: Annotations

2.1 Task design

Conceptualize and design a classification task for your tweets. Take a look at your Personal250 dataset. What's an interesting, but also reasonable way to annotate the classification problem for your tweets? It can be any binary or multiclass classification problem. Here are a few examples:

- The tweet's overall sentiment polarity (say, 3-class: positive, negative, neutral)
- Whether the tweet makes references to political news articles
- Does the tweet contain humor? Sarcasm?
- Is the tweet about a certain specific topic: politics, arts, food, . . .

A few guidelines:

- You must have a fixed set of categories. Please use no more than 5 categories. Fewer tends to be easier to annotate.
- Do not have any categories that are very rare—say, less than 5% incidence. Annotators tend to find rare classes difficult to annotate.
- For many category systems, it's often useful to have a catch-all category like "Not Applicable" or "Other" or "Unknown." Of course, those three category labels can mean different things in different situations, or you might even need more than one of them. (For sentiment, "neutral" is a kind of catch-all.)
- Don't do a really simple problem with an obvious shallow textual indicator, like "Is the tweet a retweet?" since the word 'RT:' indicates this. Or something like "Does this tweet contain a punctuation character?" This violates the spirit that we want you to do a real NLP problem. However, seemingly simple tasks might be more subtle than they first appear. For example, "Does this tweet contain an emoticon?" can sometimes be hard, if there are creative or complex ASCII-art emoticons (e.g. horizontal emoticons).

Write **annotation guidelines** for your task. These are instructions you will give your annotators, that they will read before doing the annotations. It should be a written document, perhaps a half page long (or more if you think necessary). The guidelines should include:

- A list of the categories under consideration, including the exact string you want them to use when typing into a spreadsheet.
- Descriptions of the categories and what they mean.
- Example tweets that are illustrative of the categories. **Do not use examples copied directly from your Personal250 dataset.** We specify this so that you can't make the task too easy. Please make up synthetic examples, or use examples drawn from the CommonGround dataset.
- A discussion of tricky corner cases, and criteria to help the annotator decide them. If you look at the data and think about how an annotators could do the task, you will realize a bunch of such issues!

You are in charge of defining your task! This process of boiling it down to something specific, actionable, and thus measurable, called *operationalization*. Everyone will be defining a different task, so feel free to make yours specific or unique in some way. But you will want to make it clear and as straightforward as possible for your annotators to do the task.

Deliverable: Include a copy of your annotation guidelines in your writeup.

2.2 Annotation collection

It's time to collect annotations! Find two "volunteers" from class (or even friends not from the class) to be your annotators. If you don't know anyone, just ask your neighbors from lecture and get each others' email addresses, or ask on Piazza. Everyone is required to annotate for someone else if asked, to make this easy! :)

Do not communicate with your annotators about your task too much beforehand. For this homework, we want you to communicate about the task primarily through the annotation guidelines document.

Please collect your annotations through a Google Sheets document with three columns:

1. Tweet text
2. Label from annotator
3. Notes from annotator

You should create two identical spreadsheets with just the text column, then send each, along with the guidelines, to each annotator. **Annotators should annotate independently of you, and of each other.**

(If you like, you could use other software for this. A web form interface is best, but it may be too much trouble to set up. We recommend Google Sheets because it usually works well enough.)

Deliverable: in your Feb 14 Moodle submission, include two CSV files, one for each annotator.

2.3 Annotator Feedback

Ask your annotators for general feedback, and how long it took them. Ideally, try to interview them in a meeting or call; but written feedback might be ok too.

You may be able to get some of this from the spreadsheet—Google Sheets tracks edit times to a certain extent, and you may be able to summarize some of the general issues encountered by your annotators if they left information in the 'Notes' column.

Deliverable: In your writeup, report the times and summarize the important issues in your annotation task. Did the annotators find it easy or hard? What were the biggest issues? Did the two annotators have similar or different experiences? What would you do differently if you were to revise your task?

If this were a real annotation project, what you just did would be considered the first pilot experiment, and you would work more with your annotators and iteratively refine your guidelines. But for this homework, one round is enough!

2.4 Inter-annotator agreement

Now you'll conduct quantitative analysis of the agreement between annotators. **Deliverable:** in your writeup, report:

1. The confusion matrix between the two annotators. Rows = category choices for Annotator1, Columns = category choices by Annotator2, Cells = the number of tweets with that pair of labels from the two annotators. The categories should be in the same order for both rows and columns, so the diagonal cells correspond to cases with agreement.

2. The observed agreement rate, the random chance agreement rate, and chance-adjusted agreement rate (kappa). Explain the calculations you conduct.

2.5 Aggregate to final dataset

Create the final annotated dataset, by combining the two sets of annotations. For cases where your two annotators agree, you should just use that label. But where they disagree, you will have to adjudicate and decide who's right.

Deliverable: What principles (if any) did you use to make these decisions? (Explain in your writeup). Also, include a copy of your final dataset in CSV format, as 'final.csv' in your code/data submission.

3 NLP experiments

OK great, you have some labeled data. What about NLP modeling?

Conduct experiments to build and test classifiers on your dataset. This will be a little difficult, since it is very small; fancier machine learning methods may not work well.

Since the dataset is so small, please report evaluations from cross-validation, since that more efficiently uses all datapoints for evaluation. (A static train/test split is often better to use, if you have a large enough dataset.) Make sure to describe decisions you made in how cross-validation is set up. (If you use premade crossval routines, like from scikit-learn, you must describe what they do, and how evaluation metrics are calculated, in a manner specific enough to be interpretable and replicable.) Also describe any hyperparameter tuning or selection that you conducted.

3.1 Tokenization

Choose a word tokenizer for your experiments. (The INLP reading has suggestions.) Justify it.

3.2 BOW LogReg

Create a logistic regression model based on features of the presence/count of words. (Word counts aren't much different than word presence indicators, since tweets are so short.)

3.2.1 Feature preproc.

Describe the features and preprocessing you do. You probably want to clean things a bit more beyond simply lowercasing the words. Justify your decisions.

3.2.2 Results

Report overall accuracy, as well as the precision and recall for each class.

We recommend you use the scikit-learn version of LogisticRegression with L2 regularization, and probably with DictVectorizer to aid construction of the feature vectors.

3.2.3 Model insight

Look at your features' weights. Show the top-10 highest weighted words for each category. What do they indicate, if anything?

3.3 Word embedding LogReg

3.3.1 Description

Use Twitter-specific GLOVE word embeddings. Pretrained GLOVE word embeddings can be downloaded from here: `https://nlp.stanford.edu/projects/glove/` and we recommend using the "Twitter" version (glove.twitter.27B) because it is pretrained on similar source of text to the data that you are using for the assignment. Construct a logistic regression classifier based on averaged word embeddings. Describe any particular details that are necessary to specify the model—for example, what you do for out-of-vocabulary terms, empty tweets, preprocessing to match the pretrained embeddings' vocabulary, etc.

3.3.2 Results

Include this model's results in the overall results table in the next subsection.

3.3.3 Model insight

For each output class, calculate and report the top-10 most indicative words for the class (using the method you developed in Question 1.4). What do they indicate, if anything?

3.4 Better models

Experiment a bit to create a better model than the two baseline models above. You may want to try adding heuristics, patterns, or keywords for specific features or rules. You may want to incorporate external resources, like sentiment lexicons, or Twitter-specific resources, like other word embeddings or clusters (Brendan is most familiar with these ones: `http://www.cs.cmu.edu/~ark/TweetNLP/#resources`). You may also want to try different ML models. Note that with only 250 examples, many of the fanciest models won't work well. If you are looking at ELMO/BERT, they might work in "frozen" (embedding extraction) mode.

3.5 Overall results

Create a results table with one row for each model, and including both the word feature logreg, the word embedding logreg, and better model(s) you develop. Report (as columns) the overall accuracy, as well as the precision and recall for each class.

Which model is better, if it is clear? What do these findings suggest? Report, explain, and interpret your results.