INLS 613 Text Data Mining
Homework 3

Due: Wednesday April 12th (before class) via Sakai

1 Motivation

Predictive analysis of text requires labeled data for training and testing. Usually, labeled data is produced by human annotators. Traditionally, human annotators have consisted of trained experts who go through the process of putting together a coding manual, code data until they achieve an acceptable level of inter-annotator agreement, and finally work independently to produce a large enough set of labeled data for training and testing. While this careful process tends to produce high-quality data, it has two main drawbacks: it is time-consuming and expensive.

One alternative to using highly-trained assessors is to use less expensive assessors who are given less instruction and little training. Services like Amazon's Mechanical Turk provide access to such a pool of assessors.[1] Of course, there is a catch! Using less-expensive, non-expert assessors can lead to noisy data, for several reasons: an assessor may not completely understand the task, may be biased in favor of a particular class value (e.g., *positive* or *negative* review), or may not do the task carefully in an attempt to maximize their revenue. However, because the work can be done for less, it is oftentimes feasible (and advisable) to collect *redundant* judgements (i.e., multiple judgements per instance). Redundant judgements can serve multiple purposes. They can be used to distinguish between reliable/unreliable assessors and they can be used to distinguish between easy instances (those with a clear majority label) and difficult instances (those where assessors largely disagreed). For example, if an assessor tends disagree with the majority vote label, we could down-weight his/her assessments in deciding on a final *true* label for training and testing, or we could ignore his/her assessments altogether. Likewise, when training the model, we could place more weight on instances with a higher level of agreement (i.e., clear-cut cases).

The goal for this homework is to expose you to the task of training a model using noisy *redundant* labels from multiple assessors. This has been a popular research topic in recent years. For a nice example of this type of work, see Sheng *et al.* [2].[2]

2 Assignment

This is a fairly open-ended assignment. The prediction task is classifying music album reviews into *positive* and *negative* sentiment. The data originates from the full dataset used in Blitzer *et al.* [1]. Download the dataset from: `http://ils.unc.edu/courses/2023_spring/inls613_001/hw/hw3_data.zip`.

After uncompressing hw3_data.zip, you will notice two files: `music.train.annotators.csv` and `music.test.csv`. Both files are in comma-separated format, so they can be opened using most spreadsheet applications (e.g., Microsoft Excel) and they can be opened in LightSIDE.

- `music.train.annotators.csv`: This file contains 1000 music album reviews that have been *redundantly* labeled by eight hypothetical, non-expert annotators: a_1, a_2, a_3, ..., a_8.

[1] If you are not familiar with Amazon's Mechanical Turk, start here `http://en.wikipedia.org/wiki/Amazon_Mechanical_Turk` and here `https://www.mturk.com/mturk/welcome`

[2] The third author of this paper, Panos Ipeirotis, has a done a lot of work on methods for obtaining reliable data from non-expert assessors.

Every annotator labeled every review. There is no gold standard. All you are given are these eight sets of noisy annotations. While all a_{1-8} annotators can be considered noisy annotators, they are not equal. Some may be more reliable than others (in general), some may be more reliable in predicting one class than the other, and some may be biased towards a particular class. As explained in more detailed below, your goal is to somehow use these annotations to maximize prediction accuracy on the test set. There are lots of things you could do!

- `music.test.csv`: This file contains 1000 music album reviews with gold standard labels. This is the test set. You can assume that these labels were produced by the "expert" (yet expensive) annotator we wish we had for the training set. Your goal is be try a few different ways of combining the noisy labels while training a model and to test your accuracy on this test set. **The test set should only be used for test purposes.**

2.1 Details

Try three different methods of combining the annotations from a_{1-8} and, for each method, evaluate its performance on the test set in terms of accuracy. In this case, accuracy is a good metric: the test set is roughly balanced and we'll assume that we care equally about detecting positive and negative reviews.

There are lots of things you could do (more on this below). For this homework, only try three, and, for each method, answer the following (90%):

1. Provide a thorough description of what you did. (10%)
2. Provide a well-grounded motivation for why you thought it would work. (10%)
3. Provide the accuracy of your method on the test set and discuss how well it worked and why you think that is. (10%)

Finally, taking into consideration everything you tried and whether or not it worked, provide a discussion of your overall results. Did you notice any trends? Do you have any ideas for why these trends occurred? What did you learn? (10%)

2.2 Things you might try

- Aggregate the eight redundant labels into one "true" label. There are many ways of doing this: majority vote, *weighted* majority vote (favoring labels from annotators that you predict to be more reliable), etc.
- Instance weighting. Most learning algorithms can accommodate instance weighting in some form or another. With Naive Bayes, for example, you can easily weight instances differently by simply replicating them in the training set. For example, suppose you want to weight instance i_1 twice as much as instance i_2, for whatever reason. You can do this by adding i_1 to the training set twice and i_2 only once. Or, if you want to assign different instances a weight of 0.10, 0.50, or 1.0, then you can do so by adding those with a weight of 0.10 once, those with a weight of 0.50 five times, and those with a weight of 1.0 ten times. If you go this route, it's up to you to decide how to weight instances differently

- Predicting the reliability of an annotator. There are many ways of identifying and down-weighting (or completely ignoring) unreliable assessors, for example: (1) you could compare each assessor with the majority vote, (2) you could test an annotator's consistency by training and testing a model using the assessor's own annotations (using cross-validation on the training set), or (3) you could explore the features that correlate with the assessor's positive/negative predictions and see whether they make sense to you. **Note that you cannot test the reliability of each individual annotator by training a model on his/her annotations and then applying it to the test set. The test set should only be used three times (once for each of your three strategies).**

3 Submission

Please submit your report via Sakai (Word and PDF formats only) and be sure to include your best accuracy.

References

[1] J. Blitzer, M. Dredze, and F. Pereira. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 440–447. Association for Computational Linguistics, 2007. 2

[2] V. S. Sheng, F. Provost, and P. G. Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 614–622, New York, NY, USA, 2008. ACM. 1