CS 5100 Foundations of AI

Project: Identifying Personal Attacks in Wikipedia Comments

Value: 20%

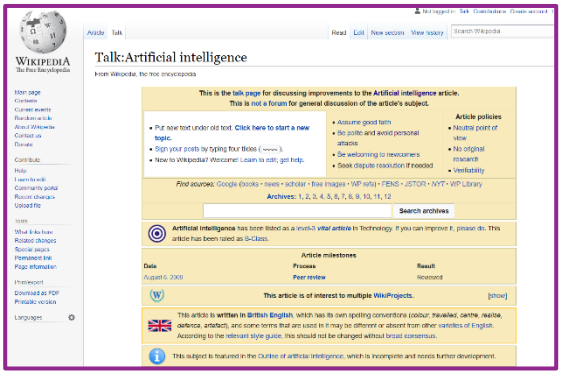
Note that there are no slip days or extensions possible for the project.

In this project, given a publicly available dataset, you will work on a classification task.

You will be using the "Wikipedia Talk Labels: Personal Attacks" data set, Version 6 of 22 Feb 2017, available from https://figshare.com/articles/Wikipedia_Talk_Labels_Personal_Attacks/4054689 . For convenience, the data has been made available on our Canvas site (details below).

I am grateful to the Wikimedia Foundation and Ellery Wulczyn, Nithum Thain and Lucas Dixon who worked on this dataset and related materials, for making this publicly available.

Here is a description of the data set from that page: "This data set includes over 100k labeled discussion comments from English Wikipedia. Each comment was labeled by multiple annotators via Crowdflower on whether it contains a personal attack. We also include some demographic data for each crowd-worker. See our wiki for documentation of the schema of each file and our research paper for documentation on the data collection and modeling methodology." Please note that this data set uses real data from Wikipedia comments. As such, these comments include some adult language.



You will develop a machine learning model to classify discussion comments on whether they contain a personal attack or not, using information from multiple (about 10) annotators. You will start learning about this project using the 'strawman' program in the Jupyter notebook referred to above. You will use a variety of mechanisms to improve upon the performance (accuracy) of the system. *The focus in the project will be on understanding what you are doing and why.* Note also in the rest of the document, when we mention result-accuracy, we mean **macro-averaged precision, recall and F1-scores**. [Note: We will stick with using macro-averaging, which has a higher penalty for when your model you create performs poorly with the minority (smaller) classes. That's what we want when we have an imbalanced data set.] It will definitely be useful to read the wiki cited above. The research paper cited above may also be of help.

The next sections describe what is expected from you.

Section 1. Get the tools ready

1. First make sure you have Python installed. Make sure you also have Jupyter installed, and you know how to use it. One of the outputs for the project will be a Jupyter notebook. See Jupyter Quick Start in Module 10 of our Canvas site for help on installing Jupyter and creating and using Jupyter notebooks. Alternatively, you can also use any IDE to develop Python programs, and you can import them from within Jupyter notebooks. But Jupyter makes it easy to develop Python code incrementally, and to document what you're doing as you go.

2. The Jupyter (iPython) notebook containing the strawman code is available from our course Canvas site. Download the Jupyter notebook **WikipediaTalkDataGettingStartedV3.ipynb** from the link **Final Project: Data files & strawman code** and save it on your laptop. This notebook has two parts: we will focus on the first part, titled "Building a classifier for personal attacks."

   Just because we have this code does not mean we're done with the project – we've just started! *You will work on multiple variations of this code, focused on improving performance and result-accuracy.*

3. The data is in two files. By default, the code downloads both these files from specific URLs every time you run the code. You may wish to download the files once and ensure that your code is changed to load the **local** copies each time afterwards. For convenience, these data files are also in our Canvas site at **Final Project: Data files & strawman code**

4. File ***attack_annotated_comments.tsv*** has the text of each discussion comment, along with a few other parameters. **rev_id** serves as the unique identifier for each comment. Read the wiki page for details of what each column represents. The image below shows a few rows of this file.

rev_id	comment	year	logged_in	ns	sample	split
37675	`-NEWLINE_TOKENThis is not ``creative`	2002	FALSE	article	random	train
44816	`NEWLINE_TOKENNEWLINE_TOKEN:: th	2002	FALSE	article	random	train
49851	NEWLINE_TOKENNEWLINE_TOKENTrue	2002	FALSE	article	random	train
89320	Next, maybe you could work on being l	2002	TRUE	article	random	dev
93890	This page will need disambiguation.	2002	TRUE	article	random	train
102817	NEWLINE_TOKEN-NEWLINE_TOKENNE	2002	TRUE	user	random	train
103624	I removed the following:NEWLINE_TOK	2002	TRUE	article	random	train

File ***attack_annotations.tsv*** has ratings from about 10 annotators on each comment, on different kinds of attacks. The image below shows a sample of the first few lines of the file, *after it has been sorted on rev_id,* showing all the annotations for the comment with rev_id 37675. We are interested only in the ratings in the (final) '*attack*' column in each row. We can ignore almost all other columns except rev_id (, maybe worker_id,) and attack in this file. If the value in the attack column is 0, then the annotator is judging that the associated comment is **not** a personal attack. If the value of attack is 1, then the annotator judges that it **is** a personal attack.

rev_id	worker_id	quoting_attack	recipient_attack	third_party_attack	other_attack	attack
37675	1362	0	0	0	0	0
37675	2408	0	0	0	0	0
37675	1493	0	0	0	0	0
37675	1439	0	0	0	0	0
37675	170	0	0	0	0	0
37675	176	0	0	0	0	0
37675	481	0	0	0	0	0
37675	487	0	0	0	0	0
37675	578	0	0	0	0	0
37675	1127	0	0	0	0	0

Rev_id is the column that links the data in the two files. You will need to link the data from the first file to the *attack* ratings in the second file. You will also need to decide how to deal with multiple annotator's ratings for each comment (rev_id). How do you decide, based on all annotations for a comment, if it is a personal attack or not?

Section 2. Get an understanding of the data and the code, and then run the strawman

1. Take a sample (say 100 rows of data) from each file. **Visually examine the data** to get a feel for the data.

2. Decide how you will use the attack information from multiple annotators. Integrate (join) the information from the two data files and see how the percentages of attacks and non-attacks vary with different parameters (year, logged_in, ns, etc.). You can do this by plotting bar charts either using Python or using Excel or similar tools. **Visualizing such relationships is a very good way to start learning from data.**

3. **Read the code and try and understand it.** There may be portions you have to look up and understand as you go. That is ok – there's lots of new material here.

4. **Run the notebook,** especially the part titled "Building a classifier for personal attacks". You may want to experiment with altering some of the code parameters to see how this change the results.

5. Take a look at the metrics generated by the *classification_report* method in the Jupyter notebook, which shows how good the machine learned model is. We will work to improve these metrics by improving every aspect of developing the model: data cleaning, feature extraction, modeling etc. Your focus should be on **macro-averaged precision, recall and F1-scores.**

Section 3. Create the Best "Personal Attacks Classifier" you can

The following steps may be done in any order you prefer, though step #1 is best done first, and step #5 is better done after step #4. Keep track of all that you do, as you will need to document trials/experiments. You will benefit from reading the paper on this dataset available at https://arxiv.org/pdf/1610.08914.pdf

1. **Text cleanup**: Often, the biggest chunk of time and effort spent in machine learning and big data projects is to get the data cleaned up and ready to be used. The strawman code does some simple cleaning of data, replacing newlines and tabs with spaces. See if you can come up with other, better ways you can clean up the data to improve performance and accuracy.

2. **Metrics**: In the strawman code, we have metrics for precision & recall, F1 score etc. You must add confusion matrix as well. See *sklearn.metrics.precision_recall_fscore_support* as an alternative to classification_report (if you prefer this), and also see *sklearn.metrics.confusion_matrix*. Note that the rows in the *attack_annotated_comments.tsv* file have a 'split' column indicating if they are in the training, test or validation (dev) set. **Combine the training and test sets and use k-fold cross-validation**. What value of k do you think will be good? Why?

3. **Feature Extraction**: This is where you decide which features you will use to classify the comments and related data. Start with using a bag of words representation using words from the comments by themselves (word unigrams). Try word and character n-grams, individually, and together, in various combinations. The strawman code uses two feature extraction methods (*CountVectorizer* and *TfidfTransformer*). You could use *TfidfVectorizer* as a single alternative to the combination of these two, to generate features. You may need to use *FeatureUnion* as well.
   Try adding other meaningful non-word features and see how they affect the accuracy. For example, is '*year*' useful as a feature? Why or why not? Your Section 2 visualization may also give you some useful information to decide this.

4. **Modeling the data**: Try **at least** 3 ML modeling methods (other than the *DecisionTreeClassifier* module used in the strawman code), to see if any of them helps improve system accuracy. Here are some methods you could try: *LogisticRegression*, *MultinomialNB*, *RandomForestClassifier*, *LinearSVC* (Linear Support Vector Machines) and Multilayer Perceptrons (*MLPClassifier*). Feel free to try other methods too.
   If you check the data, you will find the number of comments that are personal attacks are not the same as the number of comments that are *not* personal attacks. In other words, this data is not *balanced.* There are well-known techniques to address "imbalanced" classes of data in machine learning. Look up these techniques on the web. Check out this article, for example, and see how you can improve your results on imbalanced data.

   Experiment with using different ML methods to understand why some perform better. More experimentation, with different ML methods, is good. *Make sure you stop and think about the results you get from each run with each method.* Finally, pick the model that improves metrics overall for this problem.

5. **Tuning hyperparameters**: Once you find a classifier model that gives you great results, try optimizing the classifier's parameters – we call this hyperparameter tuning. For example, assume that *LinearSVC* gave you the best results with default hyperparameters. *LinearSVC* has hyperparameters like *C, class_weight* and *kernel*. A different combination of values for these parameters may improve performance even more. How would you identify the best combination(s)? Check out hyper-parameter tuning methods such as *GridSearchCV* and *RandomizedSearchCV*. Again, keep track of your work.

Once you are happy with the results you are getting, you can finish up the report and submit.

Section 4. Preparing your submission

Your submission will consist of two parts:

1. A Jupyter notebook (filename: **yourFirstName_yourLastName_Project.ipynb**) containing **your best version of the classifier to solve this problem**. This notebook must include appropriate comments and explanatory text about how you went about developing this classifier, especially about the questions (a) through (k) below. The notebook should show both the code and the results from running the code. This notebook does not need to include the code for all the experiments you tried; we need the code only for the best version overall. But you must document the other methods you tried out -- just list in the notebook all the ML methods you tried out, the parameters you used for each, and the best results with each method.

   This notebook should clearly include answers for the following questions:
   a. What did you learn from visualizing the data as suggested in Section 2 above?
   b. What text cleaning methods did you try? Which are the ones you included in the final code?
   c. What are the features you considered using? Which features did you use in the final code?
   d. How did you decide to use the 'attack' information from different annotators? Did you average them, or use a number threshold, or did you use some other method to use this information?
   e. Did you add any special optimizations to your code? If so, describe them briefly.
   f. What are the ML methods you tried out, and what were your best results with each method? Which was the best ML method you saw before tuning hyperparameters?
   g. What hyperparameter tuning did you do, and by how many percentage points did your accuracy go up because of hyper-parameter tuning?

h. What did you learn from the different metrics? How useful was it to use cross-validation?
i. What are your best final Result Metrics? What is the increase in accuracy compared to the strawman figure? Which model gave you this performance?
j. What is the most interesting thing you learned from doing the report?
k. What was the hardest thing to do?

The notebook and its contents will be evaluated for a total of 15 points, of which up to 2 points will be based on the best performance (in terms of recall and precision) you were able to achieve.

2. Slide deck: A slide deck, (filename: **yourFirstName_yourLastName_ProjectSlides.pptx**) less than or equal to 5 slides, in Microsoft PowerPoint format. The deck must answer questions in items (a) to (i) in item 1 of submissions above. Clarity will be much appreciated. You do not have to detail the problem specification in the deck – let's assume the audience knows that. In a couple of weeks, we will list some guidelines for this slide deck.

A third part of your project will be the presentation of your slide deck. On Dec 9th, during class, in person, each of you will make a brief (3-4 minute) presentation of your project, using the slide deck you submitted. Each slide deck and presentation will be evaluated for a maximum of 5 points.

We do not want anything fancy, but we will ask that each presentation be clear and audible. Do practice your presentation to make sure you finish in time and do a good job. In PowerPoint, you may wish to use the Rehearse Timings feature (under the Slide Show menu) to get your timing right. If possible, present it to friends or family and get their feedback to improve your presentation. There's a preview feature in PowerPoint (also under the Slide Show menu) "Rehearse with Coach", which provides feedback on your presentation -- try it out if you can! Do not wait till the last minute to practice your presentation.

Section 5. Submission

1. When you have all components of your submission ready, please name the files using the format:
   yourFirstName_yourLastName_Project.ipynb , and
   yourFirstName_yourLastName_ProjectSlides.pptx

2. Zip up the files into one .zip file, again with the **yourFirstName_yourLastName_Project** name and submit the zip file via Canvas.

   Note that there are no slip days or extensions possible for the project.

3. Please also email a copy of your slides to your instructor **via Piazza.**

You will be evaluated on the content (not on style – this is not a writing course), but readability and correct spelling will help you get higher scores.

Please do not plagiarize. Keep in mind the Academic Integrity policy of the University.

Working on projects and making presentations is increasingly an integral part of any job in our industry. I hope this will be a good learning experience for you!

[End of Assignment]