Prize recipient documentation guide

Congratulations! You've gone up against dataheads from around the globe and emerged victorious! Laugh, dance, brush your shoulders off. You demonstrated serious skills, and helped make this world a better place in the process. Awesome job. Now you've finished in one of the top spots of the private leaderboard, which makes you eligible to receive a monetary prize following verification. You're almost there.

In accordance with the official competition rules, the DrivenData terms of use, and applicable State and Federal law, we both have some due diligence to take care of before we can announce winners and disburse prizes.

There are three steps left in the process which all involve you sending us materials. These steps must be completed and information received by DrivenData no later than December 1, 2020 or your prize may be forfeited and an alternate winner may be selected.

I. **Legal identity verification.**
You send us documentation so that we can verify your legal identity. We verify your eligibility to participate and then review the specific laws and rules about giving out prizes based upon your nationality and our tax reporting obligations. *Note: we are required by US Federal Law to withhold 30% of prize winnings for non-US individuals who do not already pay US taxes, unless exempted under an applicable income tax treaty.*

II. **Code submission and result reproducibility.**
You package up and send us your code, documentation of dependencies, and any other assets you used. We review your package and make sure that it works and that we can fully reproduce the workflow from raw data to a submission comparable to your best submission. We recommend our open source data science template as an effective structure for sharing code.

III. **Model documentation and write-up.**
You write up answers to our questionnaire, providing important context and documentation so that the beneficiary and the community get the most out of your work. You are also required to provide an academic explanation of your solution and publish it to arxiv.org or a similar platform under a permissive Creative Commons license without commercial restriction.

IV. **Winner Declaration of Eligibility and Liability/Publicity Release**
Each member of your team reviews, completes and signs a separate copy of the additional document provided to you, titled the 'Winner Declaration of Eligibility and Liability/Publicity Release'. We require a separate signed copy from you and each member of your team.

Please read this document carefully. Each section details exactly what is needed from you—the faster we can check all the boxes for our mutual responsibilities, the faster we can disburse your prize!

Thanks for your hard work, and congratulations for making it this far.

Best,
The DrivenData Team

I. Legal identity verification

Note: we are collecting this information in accordance with our competition rules and privacy policy. Virtually every government jurisdiction mandates that prize-awards are reported to tax authorities. Please see the official competition rules and our privacy policy for more details.

1. **Basic information.**
   Please provide us with the following information, numbered and in order.
   a. Full legal name:
   b. Date of birth:
   c. Citizenship:[1]
   d. Residential address (where you actually live and what you list on tax forms):
   e. Mailing address:[2]

   If for security reasons you would rather relay the following identification number over the phone or by other means, please let us know so that we can make arrangements.

   f. ***US only:*** Social security number (SSN) or taxpayer identification number (TIN)
   g. ***If not a US taxpayer:*** your local equivalent to a taxpayer identification number[3]
   h. ***If not a US taxpayer:*** Do you agree to have relevant tax reporting documents sent as an email attachment (our preference), or do you prefer to receive a paper copy by postal mail to the address above? *(Note: for added security, if we send as an email we will attach in a password-protected zip folder and send you the password in a separate email.)*

   ___ Email (our preference) ___ Postal Mail

2. **Documentation.**
   In your response e-mail, please attach a color scan or photograph of a currently valid legal identification document. Examples of acceptable forms of identification include:
   - Driver's license
   - Legal identification card

[1] From what country do you have a valid, current passport? Or, if you do not hold a passport, to which country would you file a request with the reasonable expectation that a passport would be issued?
[2] This is where we will send a check.
[3] If you are not a US citizen, what is the equivalent identification number that you would use to file your taxes? *Please notify us immediately if the laws of your country or locality do not permit companies to request or collect this information; in that case, we will figure out what information we are required to report to relevant tax authorities.*

- Passport

3. **Basic information for winner announcement.**
   Provide your preferred information for use in announcing the winners of the competition.
   - Name (first and last name or first name and last initial):
   - Hometown:
   - A recent picture of yourself or digital avatar:

II. Code submission and result reproducibility

You will need to submit a compressed archive of your code. You don't need to include the raw data that we provided, but everything else should be included and clearly organized. If the files are too large to be e-mailed, a Google Drive or Dropbox share (or other comparable method of transferring data) works.

Here's the overall concept: **please set this archive up as if it were a finished open source project**, with clear instructions, dependencies and requirements identified, and code structured logically with an obvious point of entry. Again, we have a data science project template which may be helpful.

Note: *please follow these instructions carefully*. The spirit and purpose of the competition (and the reason for offering prizes) is to give our beneficiary organizations the best possible solution *along with working code they can actually use*. In accordance with the competition rules, if we can't get your code working and reproduce your results with a reasonable effort, or if your entry is too disorganized to be practically usable, then your entry may be disqualified!

At a minimum, **this means the inclusion of an extremely clear README** that details all of the steps necessary to get to your submission from a fresh system with no dependencies (e.g. a brand new Linux, Mac OS X, or Windows installation depending on what environment you choose to develop under) and no other data aside from the raw data you downloaded from us.

This will probably entail the following:

- Necessary tools and requirements (e.g. "You must install Word2Vec 0.1c" or "Install the required Python packages in `requirements.txt`").
  - **All requirements should be clearly documented**, for instance in either a requirements.txt file with versions specified or environment.yml file.
- The series of commands, in order, that would get a reasonably experienced and savvy user from your code to a finished submission.
  - **Ideally, you will have a main point of entry to your code** such as an executable script that runs all steps of the pipeline in a deterministic fashion. A well-constructed IPython notebook or R script meets this standard.
  - **The next best thing is a list of specific, manual steps** outlining what must be done. For example, "First, open Word2Vec and set the following parameters. [...] Take the output file from Word2Vec and run the script `src/make_preds.R` with the following parameters [...]". *(The limitations of this approach should be clear to any experienced data scientist!)*

    - **Make sure to include your trained model(s)** in your archive as well so that inferencing can be run without needed to train everything from scratch.
- Any other instructions necessary to end up with your winning submission file (or comparable — we understand that certain parts of model fitting are stochastic and won't result in exactly the same parameters every time).

As set forth in the Official Rules, you are required to publish your model and source code used to generate the model to a public GitHub repository under a permissive open source license approved by the Open Source Initiative "Approved OSS License."

III. Model documentation and write-up

Information included in this section may be shared publicly with challenge results. You can respond to these questions in an e-mail or as an attached file. Please number your responses.

1. **Who are you (mini-bio) and what do you do professionally?**

   Ekaterina: I am an Assistant Professor at the Institute for Logic, Language and Computation (ILLC) at the University of Amsterdam, where I lead the Amsterdam Natural Language Understanding Lab. Previously, I was a Leverhulme Early Career Fellow at the University of Cambridge and a Research Scientist at the University of California, Berkeley. I received my PhD in Computer Science from the University of Cambridge. My research is in the area of natural language processing, with a specific focus on machine learning for natural language understanding tasks. My current interests include few-shot learning and meta-learning, joint modelling of language and vision, multilingual NLP and societal applications of NLP, such as hate speech and misinformation detection.

   Helen: I am an Assistant Professor at King's College London and a Visiting Researcher at the University of Cambridge. Previously, I was an Affiliated Lecturer and a Senior Research Associate at the Department of Computer Science and Technology of the University of Cambridge, a Fellow and Director of Studies in Computer Science at Murray Edwards College, and a Newton Trust Teaching Fellow at Girton College, Cambridge. I hold a PhD in Natural Language and Information Processing from the University of Cambridge. Current research interests include transfer and multi-task learning, few-shot learning, continual learning, dialogue systems, and explainable machine learning, as well as machine learning for real-world applications and non-canonical forms of language.

   George: I am currently working as part of a machine learning research team for a medical diagnosis start-up. Previously, I was a short-term Visiting Researcher at King's College London working on multimodal approaches for hate speech detection. I hold a PhD in Pure Mathematics from the University of Glasgow and Master's degree in Applied Mathematics from the University of Cambridge.

   Nithin: I am a recent graduate from the MSc AI program at the University of Amsterdam. I worked on meta-learning for few-shot and continual learning in NLP as part of the master thesis.

Phillip: I'm a first-year PhD student at the University of Amsterdam working on temporal causality and deep learning. Before starting the PhD, I have completed a Master degree in Artificial Intelligence at the University of Amsterdam.

Santhosh: I am a Machine Learning Engineer (NLP) at Slimmer AI, and I am part of the team responsible for building and maintaining various AI products used by Springer Nature. I graduated from University of Amsterdam with a Masters degree in Artificial Intelligence in 2019 and my thesis was on joint modelling of emotion and abusive language detection.

Shantanu: I am currently working as an AI Research Scientist at ZS. Before this role, I did my Masters in Artificial Intelligence from University of Amsterdam. My masters thesis was on graph-based modeling of online communities for misinformation detection in collaboration with Facebook AI and King's College London.

2. **What motivated you to compete in this challenge?**
   We had never got a chance to work on a multi-modal problem and we saw this challenge as the perfect opportunity to get acquainted with the field and solve an important social problem in the process. Learning jointly from the two modalities poses its own unique set of challenges and this competition was a great learning experience. Also, the monetary reward for top 5 teams served as an incentive to be competitive and improve our solution to the best of our abilities.

3. **High level summary of your approach: what did you do and why?**
   The late-fusion baselines for the task were shown to underperform whereas the early fusion systems seemed promising. Thus, we experimented with two early-fusion pretrained models, namely UNITER, OSCAR as well as LXMERT. We found that UNITER performed the best, which could perhaps be attributed to its diverse set of pretraining tasks. We observed that simple fine-tuning on the HatefuMemes dataset led to poor performance on text confounders in the training set whereas the model could overfit on the image confounders and non-confounders. Thus, we decided to upsample these text confounders to give additional weight to these examples. Additionally, we added more weight to the hateful examples to combat the class imbalance. Furthermore, we performed cross-validation style training for better generalizability of the model. The final prediction was obtained as a weighted ensemble where the ensemble weights were optimized using an evolutionary algorithm (EA) on the development set predictions. We also observed that the dev_seen split had a higher percentage of truly multimodal examples than the training set. Thus, we included part of the dev_seen set into the cross-validation training. Specifically, for each fold, we split the dev_seen set in two halves, and used one part for training, and the other for testing. Thereby, we included text confounder pairs with different labels together to increase the confounder amount for training. This means that examples that make a confounder pair are not split between the train and test set. Finally, the predictions on the test parts were used for the EA optimization.

4. **Copy and paste the 3 most impactful parts of your code and explain what each does and how it helped your model.**

    1. The use of pretrained UNITER as the base model was one of the most impactful choices that helped us improve over the provided baselines. Here we show the model initialization using the pretrained model:

```
def init_model(self):
    if self.pretrained_model_file:
        checkpoint = torch.load(self.pretrained_model_file)
        LOGGER.info('Using pretrained UNITER base model {}'.format(self.pretrained_model_file))
        base_model = UniterForPretraining.from_pretrained(self.config['config'],
                                                          state_dict=checkpoint['model_state_dict'],
                                                          img_dim=IMG_DIM,
                                                          img_label_dim=IMG_LABEL_DIM)
        self.model = MemeUniter(uniter_model=base_model.uniter,
                                hidden_size=base_model.uniter.config.hidden_size,
                                n_classes=self.config['n_classes'])
    else:
        self.load_model()
```

    2. Using cross-validation style training along with a weighted ensemble where the weights are optimized using an evolutionary algorithm also helped increase the performance further. Here we show the call to the ensemble optimization algorithm:

```
if EA_IMPORTED:
    logger.info("Starting EA to find optimal weights...")
    ea_score, ea_config = EA_ensemble_finder(eval_func=eval_func,
                                             num_weights=len(dev_preds),
                                             individual_scores=dev_scores)
```

    3. Including part of the dev_seen set in cross-validation was also an important choice that provided a further boost in performance. Here we show the call to the function that creates the cross-validation split. By passing use_dev_set = True, we include the dev set while generating the splits.

```
if not os.path.isdir(crossval_path) or len(glob(os.path.join(crossval_path, "*.jsonl"))) == 0:
    logger.info("Creating cross-validation splits for dev size %i" % dev_size)
    generate_crossval_splits(config['data_path'], dev_size=dev_size, use_dev_set=use_dev_set)
```

5. **Please provide the machine specs and time you used to run your model.**

- **CPU (model):** Intel Xeon Gold 5118 Processor (16.5M Cache, 2.30 GHz)
- **GPU (model or N/A):** Single Nvidia TitanRTX
- **Memory (GB):** 45 GB
- **OS:** Debian GNU/Linux 10 (buster)
- **Train duration:** 6 hours
- **Inference duration:** 2 mins

6. **Please list any external data or pre-trained models you used to develop your winning submission.**
   External Data: None
   Pretrained model: UNITER(base)
   - Paper: UNITER
   - **Official GitHub:** UNITER code

7. **What are some other things you tried that didn't necessarily make it into the final workflow (quick overview)?**

   - We tried UNITER-large as a next step to using UNITER-base in our initial experiments. Overfitting was more severe and led to lower performance on the test set.
   - We tried the OSCAR-base and -large as well as LXMERT as our base model. Performance was lower than UNITER.
   - We tried a warm-up phase for the pretrained UNITER model as a domain-adaptation step to the hateful memes dataset. We used the UNITER pretraining tasks of MLM, ITM and MRFR on this dataset. We followed it by supervised learning for hateful meme detection task using this new "warmed-up" pretrained model. This gave no improvements.
   - We extracted fine-grained labels (eg, recognizing people as Omani, Haiti, etc) using YOLO9000 for the images in the hateful meme dataset. We then concatenated these labels to the meme text during training to give the model precise information of the subjects in the image. Lead to no improvements.
   - Looking at the offensive memes, we realized that most of them targeted social groups such as women, disabled, Muslims, Jews, etc. In order to leverage that information, we tried to use the Social Bias Frames dataset to detect the social minority being referred to in the meme through the meme text. To that effect, we first tried using it in an MTL fashion (primary task - hateful meme detection, auxiliary task - social minority detection) which gave no improvements. Next we trained a purely text model (RoBERTa) for the social minority detection using the Social Bias Frames dataset and then appended these labels to the meme text during training UNITER. Again, no improvements in test scores.
   - We tried a similar MTL approach with the GoEmotions dataset in an attempt to supplement the model with the emotions associated with each meme (primary task - hateful meme detection, auxiliary task - emotion detection). Lead to no improvement.
   - We trained the model using margin ranking loss instead of binary cross-entropy loss. Training was performed on pairs of data points - pairs of text confounders and random pairs of non-confounders. Lead to no improvements.
   - Experiments showed that the model performed really well on image confounders while poorly on text confounders indicating that it was utilizing mostly the textual information of the input and ignoring the image features. Thus, we tried to split the internal self-attention of the UNITER transformer into 4 independent chunks - text->text, text->img, img->text, img->img -- to enable different attention dropout probabilities on different subsets of the input. We hoped that a higher text2text

dropout with lower img2text and text2img dropout will help, as we overfit on the text input. No improvements were noted under different values of dropouts.

8. **Did you use any tools for data preparation or exploratory data analysis that aren't listed in your code submission?**
   We extracted our own Faster R-CNN features, as opposed to using the ones provided in the MMF library. We have included the code for the same in the submission.

9. **How did you evaluate performance of the model other than the provided metric, if at all?**
   We monitored precision, recall, F1 on the hateful class, AUCROC and accuracy during training. However, the best model was chosen based on just auroc and accuracy.

10. **Anything we should watch out for or be aware of in using your model (e.g. code quirks, memory requirements, numerical stability issues, etc.)?**
    Choose batch size that fits into GPU memory. For larger batch size use the *--gradient_accumulation* parameter (see code). Apart from that no quirks or instability issues. Given the hardware specifications and the configuration provided in the readme file, our approach is perfectly reproducible.

11. **Do you have any useful charts, graphs, or visualizations from the process?**
    We visualized the evaluation metrics and the loss during training using Tensorboard. Nothing else apart from that.

12. **If you were to continue working on this problem for the next year, what methods or techniques might you try in order to build on your work so far? Are there other fields or features you felt would have been very helpful to have?**
    - Looking at the errors our model makes on the dev_seen set, we find that it lacks in its image understanding capability. Some of the false negatives are memes that are hateful because of the following in the images - black people, differently-abled people, Hitler, Nazi symbolism, Anne Frank, domestic abuse indicators etc. We believe that the Faster R-CNN features do not encode finer details such as these. Thus, training an image feature extractor on a different dataset that has information on racial profiles, disabilities, abuse, well-known persons (e.g., Hitler) etc could help in increasing the performance.
    - Yet another possibility is to fine-tune (parts of) the Faster R-CNN backbone while training UNITER, in an end-to-end fashion. This could perhaps combat the domain shift between ImageNet and the current dataset.

13. **Provide a link to the GitHub repository for your model and source code.**
    https://github.com/Nithin-Holla/meme_challenge

14. **Provide a link to your published academic explanation of your solution.**
    In progress.