III. Model documentation and write-up

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

Riza Velioglu:

After receiving my bachelor's degree in Electrical & Electronics Engineering from Istanbul Bilgi University in 2018, I directly started to my master's degree in Intelligent Systems at Bielefeld University. Currently, writing the master's thesis and planning to graduate in February 2021. At the same time, I have been the Teaching Assistant of the courses; Intro to Machine Learning, Intro to Neural Networks, and Intro to Computer Algorithms for the previous 1.5 years. In addition, I am actively looking for PhD opportunities in the areas of Vision and Language Representations, Multi-Modal Deep Learning, Self-Driving Cars (Semantic Segmentation, Lane Detection, Image Classification, Object Detection, etc.)

Jewgeni Rose:

I completed my Master's degree in Computer Science at the University of Brunswick in 2017. After that, I started at the VW Group Innovation Center in Europe on the topic of digital assistants. At the same time, I began as a PhD student at the University of Bonn under the supervision of Prof. Dr. Jens Lehmann. My research area is Question Answering, specifically Context-aware Question Answering, whereby context is external knowledge (e.g. sensory data), chat history, or a user profile.

2. What motivated you to compete in this challenge?

(Riza)

I have a background on Image Processing and Natural Language Processing through university projects and as a personal interest. Therefore, Hateful Memes Challenge was a perfect chance for me to take part because it tries to leverage from joining the two modalities, that is multi-modality, which is perfect for someone working in both areas.

Moreover, I always try to do something that can have a direct impact in the real-world! With this challenge, it was possible to work on Hate Speech which we face in every day of our lives.

3. High level summary of your approach: what did you do and why?

The approach could be summed up as follows:

- Growing training set by finding similar datasets on the web,
- Extracting image features using object detection algorithms (Detectron),
- Fine-tuning a pre-trained V+L model (VisualBERT)
- Hyper-parameter search and applying Majority Voting Technique

Growing training set:

The more samples one has, the better score one will get! Therefore, we added more samples to the training data. We found out that there are 100 memes in *dev_seen* which are NOT in *dev_unseen*! First, we added those 100 memes to the training data.

Secondly, we searched for open-sourced datasets available on the web that is somewhat similar to Hateful Memes dataset. There was none because Hateful Memes was the first dataset on this regard. But we found the one called 'Memotion Dataset' which was planned to be one of the tasks at SemEval 2020, to be specific Task 8 ([Paper], [SemEval]). The dataset is open-sourced and can be downloaded [by this link].

In this dataset all 14k samples were annotated the class labels as *Humorous, Sarcasm, Offensive*, and *Motivational* and quantified the intensity to which a particular effect of a class is expressed. The effects of each of the class is given in the following table:

Humour	Sarcastic	Offensive	Motivational
Not Funny	Not Sarcastic	Not Offensive	Motivational
Funny	General	Slight Offensive	Not Motivational
Very Funny	Twisted Meaning	Very Offensive	
Hilarious	Very Twisted	Hateful Offensive	

For instance, a meme can be annotated as: *"Not Funny, Very Twisted, Hateful Offensive, Not Motivational"*. We added all the *"Hateful Offensive"* and *"Very Offensive"* to the training data as hateful memes and added *"Not Offensive"* ones as non-hateful memes. We discarded the *"Slight Offensive"* memes because we thought it might confuse the model. With this training data the model did not achieve a better score. Later we found out that the memes are not labelled correctly (in our opinion). Therefore, we labelled the dataset manually! We searched for the memes that are somewhat similar to the ones in Hateful Memes challenge considering the idea of the challenge. After cherry-picking the "similar" memes, we ended up having 328 memes. Then, we added those 328 memes to the training set.

Extracting image features using object detection algorithms (Detectron):

After collecting the whole dataset, we extracted our own image features using Facebook's Detectron. To be specific we used Mask R-CNN and ResNet 152 as the backbone network architecture.

Fine-tuning a pre-trained V+L model (VisualBERT):

After trying different models and different pre-trained VisualBERT models, we found out that the model which was pre-trained on the Masked Conceptual Captions gave the best score. The pre-trained model is provided by Facebook's MMF and reachable via this link (pretrained key= *visual_bert.pretrained.cc.full*).

Hyper-parameter search and applying Majority Voting Technique:

We then did a hyper-parameter search and ended up having multiple models having different ROC-AUC scores on *"dev_unseen"* dataset. We sorted them by the ROC score and took the first 27 models (27 is chosen arbitrarily). Then, predictions are collected from each of the models and the majority voting technique is applied: the 'class' of a data point is determined by the majority voted class. For instance, if 15 models predicted Class1 for a sample and 12 models predicted Class0, the sample is labelled as Class1 cause it's the majority voted class. The 'proba' which stands for the probability that a sample belongs to a class is then determined as follows:

- if the majority class is 1, then the 'proba' is the **maximum** among all the 27 models
- else if the majority class is 0, then the 'proba' is the **minimum** among all the 27 models

We would argue that this technique works because it brings the "experts of the experts"! Imagine that one model is very good at -in other words it's expert in- detecting hate speech towards Woman but might not be an expert in detecting hate speech towards Religion. Then we might have another expert which has the opposite case. By using the majority voting technique, we bring such experts together and benefit from them as a whole.

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

- Extracting image features: we run the extract_features_vmb.py, which uses Mask R-CNN and ResNeXt-152 as the backbone network architecture to collect 100 features per image.

```
python extract_features_vmb.py \
        --config_file "https://dl.fbaipublicfiles.com/pythia/detectron_model/detectron_model_x152.yaml" \
        --model_name "X-152" \
        --output_folder "/home/features/" \
        --image_dir "/root/.cache/torch/mmf/data/datasets/hateful_memes/defaults/images/img/" \
        --num_features 100 \
```

- Hyper-parameter Search: the process is started with the following code.

```
feats_dir = os.path.join(home, "features")                    # Define where image features are
train_dir = "hateful_memes/defaults/annotations/train.jsonl"  # Define where train.jsonl is
python sweep.py --home $home --feats_dir $feats_dir --train $train_dir   # Start hyper-parameter search
```

which simply iterates through the possible hyper-parameters in *sweep.py#L18-L29:*

```
...
i = 1     # for each iteration a folder will be created whose name is 'i'

for bs in [32, 64, 80]:
   for lr in [0.3, 0.6]:
      for w_steps in [250, 500]:
         for w_type in ['warmup_cosine', 'warmup_linear']:
            for w_factor in [0.1, 0.3]:
               for w_iter in [500, 1000]:
                  # call the bash script which start the training
                  rc = call(f"./sweep.sh {str(i)} {lr} {w_steps} {w_type} {w_factor} {w_iter} {bs} {feats_dir} {train_dir}", shell=True)
                  i+=1
```

- Applying Majority Voting Technique: after the search, we end up having multiple models. First, we choose top *n* models (we chose *n*=27) and collect predictions for all those models. Second, we apply the majority voting technique:

```
 # Store all the prediction folders
folders = [i for i in os.listdir("save/preds") if i.startswith("hateful_memes")]
preds = pd.DataFrame()

for folder in folders:
   pred = [i for i in os.listdir(f"save/preds/{folder}/reports/") if i.endswith(".csv")]
   pred = pd.read_csv(f"save/preds/{folder}/reports/{pred[0]}")
   preds = pd.concat([preds, pred], axis=1)

submission = pred
np_df = np.asarray(preds)

for idx, row in enumerate(np_df[:,:]):
   probas = row[1::3]
   labels = row[2::3]

   if sum(labels) > 13:
      submission.loc[idx, 'label']=1
      submission.loc[idx, 'proba']=probas.max()
```

```
else:
    submission.loc[idx, 'label']=0
    submission.loc[idx, 'proba']=probas.min()
```

5. Please provide the machine specs and time you used to run your model.
    - CPU (model): Intel(R) Xeon(R) CPU @ 2.30GHz
    - GPU (model or N/A): 4xNVIDIA RTX 2080 Ti (11GB)
    - Memory (GB): 16GB
    - OS: Ubuntu 16.04
    - Train duration: ≈ 1.5 days
    - Inference duration: ≈ 38 minutes
6. Please list any external data or pre-trained models you used to develop your winning submission.
    - We used VisualBERT which was pre-trained on Conceptual Captions dataset. The pre-trained model is available at MMF, with 'visual_bert.pretrained.cc.full' as the pre-trained key.
    - We added a portion of Memotion Dataset (Paper, Dataset) to the training data.
7. What are some other things you tried that didn't necessarily make it into the final workflow (quick overview)?
    - Using accuracy metric than Cross-Entropy for training
    - VisualBERT models which are pre-trained on different datasets
    - Adding all the samples and annotations from the Memotion Dataset to the training set
    - Different encoders for image feature extraction
8. Did you use any tools for data preparation or exploratory data analysis that aren't listed in your code submission?

No. All the code is provided under the GitHub repository.

9. How did you evaluate performance of the model other than the provided metric, if at all?

We tried to maximize accuracy, but the resulting models did not achieve better scores on the development set.

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.)?

If the specified GPU RAM is not available, one can reduce the batch_size.

11. Do you have any useful charts, graphs, or visualizations from the process?

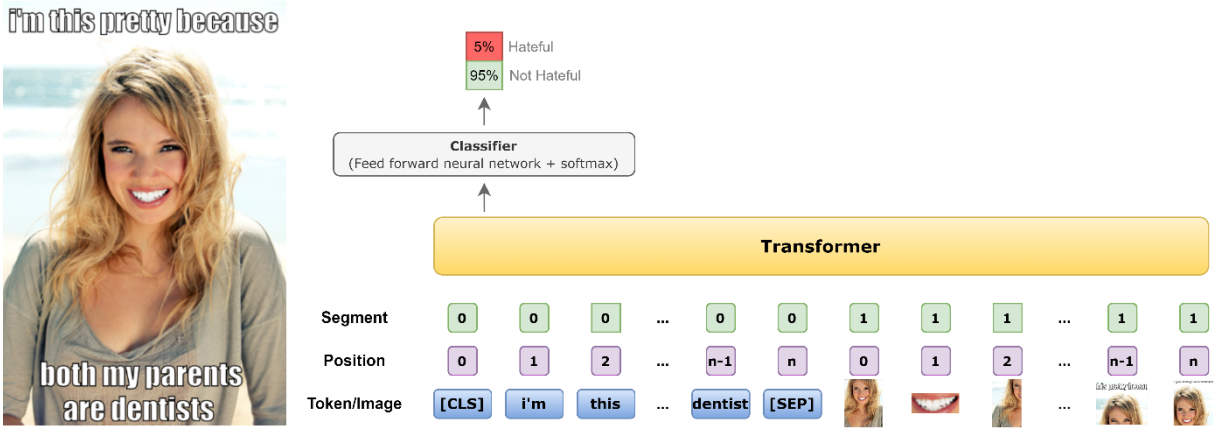

Figure 1: An illustration of the multimodal transformer architecture

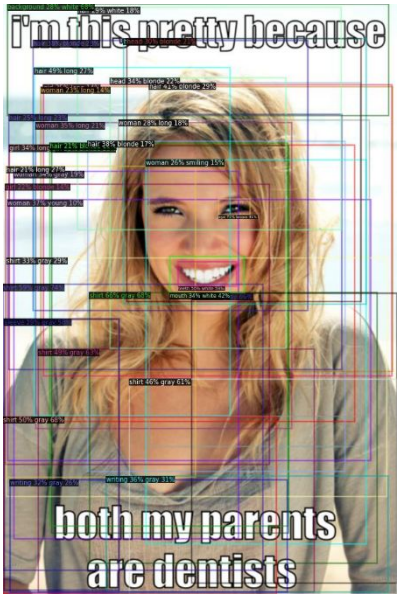

Figure 2:An example of a processed image

Hyperparameter Sweeps								Phase 2-Validation	
id	lr_ratio	warmup_factor	warmup_iterations	lr	batch_size	scheduler.num_warmup_steps	scheduler.type	ROC-AUC	Acc.
18	0.3	0.2	1.00E+03	5.00E-05	32	2000	warmup_linear	0.7521	0.7093
19	0.3	0.3	1.00E+03	5.00E-05	80	250	warmup_cosine	0.7516	0.6963
8	0.7	0.1	1.00E+02	5.00E-05	80	50	warmup_cosine	0.7502	0.7074
16	0.3	0.7	1.00E+03	5.00E-05	80	500	warmup_cosine	0.75	0.7074
7	0.6	0.3	1.00E+03	5.00E-05	80	500	warmup_cosine	0.7497	0.7167
21	0.6	0.3	1.00E+03	5.00E-05	80	500	warmup_linear	0.7449	0.7037
6	0.3	0.05	5.00E+02	5.00E-05	80	250	warmup_cosine	0.7447	0.7037
15	0.6	0.3	1.00E+03	5.00E-05	80	500	warmup_cosine	0.7444	0.7019
0	0.3	0.2	2.50E+02	5.00E-05	80	250	warmup_cosine	0.7437	0.7093
2	0.3	0.05	1.00E+03	5.00E-05	80	250	warmup_cosine	0.7432	0.7019
9	0.3	0.1	1.00E+03	5.00E-05	80	250	warmup_cosine	0.7427	0.7111
20	0.6	0.5	1.00E+03	5.00E-05	80	250	warmup_cosine	0.7419	0.7
13	0.3	0.7	1.00E+03	5.00E-05	80	250	warmup_linear	0.7402	0.7185
23	0.3	0.05	2.50E+02	5.00E-05	80	250	warmup_cosine	0.7402	0.6833
17	0.3	0.1	2.50E+02	5.00E-05	80	250	warmup_linear	0.7402	0.7111
22	0.3	0.5	1.00E+03	5.00E-05	80	250	warmup_linear	0.7401	0.6926
1	0.6	0.3	1.00E+03	5.00E-05	80	250	warmup_cosine	0.7396	0.7111
11	0.3	0.05	5.00E+02	5.00E-05	80	500	warmup_cosine	0.7396	0.7037
24	0.7	0.1	2.50E+02	5.00E-05	80	50	warmup_cosine	0.7391	0.6963
5	0.7	0.3	2.50E+02	5.00E-05	80	50	warmup_linear	0.7391	0.6963
25	0.6	0.3	1.00E+03	5.00E-05	80	250	warmup_cosine	0.7386	0.6926
26	0.6	0.7	1.00E+03	5.00E-05	80	250	warmup_cosine	0.7381	0.7
14	0.3	0.05	2.50E+02	5.00E-05	80	250	warmup_linear	0.7381	0.7111
12	0.3	0.3	1.00E+03	5.00E-05	80	500	warmup_linear	0.7378	0.7093
4	0.3	0.05	2.50E+02	5.00E-05	80	500	warmup_cosine	0.7376	0.7056
3	0.3	0.05	5.00E+02	5.00E-05	80	250	warmup_linear	0.7375	0.7093
10	0.3	0.1	5.00E+02	5.00E-05	80	250	warmup_cosine	0.7368	0.6981

Figure 3:Majority Voting models derived from VisualBERT CC

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?

- Providing the object tags, as used in OSCAR,
- giving access to a knowledge source, as most of the memes require an understanding of real-world events,
- using other V+L models such as: OSCAR, UNITER, ERNIE-ViL and growing the majority voting models,

would be some of the techniques, methods that may result in a more accurate model.

13. Provide a link to the GitHub repository for your model and source code.

Please find the GitHub repository at https://github.com/rizavelioglu/hateful_memes-hate_detectron

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

Please find the paper at https://arxiv.org/abs/2012.12975