Adobe Behaviour Simulation Challenge

Team 83
https://github.com/tea83/team83

Abstract

Simulating User Engagement on Social Media with Feature Extractors, Boosting Models, and LLMs This work tackles the dual challenge of behavior simulation (predicting user engagement on existing content) and content simulation (generating content that elicits desired engagement) by leveraging a powerful combination of multi modal feature extractors, gradient boosting regressors, ensemble neural networks, image captioning models and large language models (LLMs).

Keywords: Social media marketing, user engagement, behavior simulation, content simulation, feature extraction, boosting models, LLMs, image captioner, image generation.

1 Introduction

We leveraged a robust dataset from Twitter enterprise accounts, undertaking exploratory analysis and data cleaning. Processing text and images using pretrained models, we created a multi-modal dataset. An ensemble of regressors predicted user engagement, and media was converted to text for task two. The engineering of prompts and fine-tuning of a Large-Language model with LoRA and PEFT enhanced our approach.

In an additional stride, we developed an image generative model to complement content and optimize user engagement, aiming to drive desired Key Performance Indicators (KPIs).

2 Exploratory Data Analysis

Analyzing engagement in a large tweet corpus reveals varied patterns. While the average tweet receives 773 likes, a significant standard deviation of 4931 indicates a long-tail distribution. Noteworthy usernames like CNN and EuroLeague, and businesses such as IndependentNGR and AMCTheatres, dominate. Inferred companies span news outlets (CNN, CBC), tech giants (Cisco), and financial institutions. Daily trends hint at a potential weekly engagement pattern, with peak likes on Sundays (889) and a drop on Thursdays (675).

3 Data Preprocessing

We conducted a comprehensive text cleaning process, removing punctuation, stop words, URLs, and correcting spelling and grammatical errors to enhance text quality.

Temporal features, including day-of-week, hour-of-day, and time since the last company tweet, were extracted to consider temporal factors in user engagement. To facilitate a nuanced analysis, the inferred company name was appended to each cleaned tweet, enabling brand-specific investigation into engagement patterns. We systematically downloaded tweet images for a joint analysis of visual and textual elements, enhancing exploration of user engagement dynamics.

Step	**Description**
Text Cleaning	Remove noise, standardize formatting
Date-Time Features	Extract temporal attributes for analysis
Company Name Inferral	Append inferred company names to cleaned tweets
Image Download	Systematically download images for joint visual-textual analysis

Table 1: Data Preparation Steps

4 Task-1: Behavior Simulation

Given the content of a tweet (text, company, username, media URLs, timestamp), the task is to predict its user engagement, measured by likes.

4.1 Feature Engineering

We employ advanced techniques for text and visual embedding in the analysis of social media content. DistilBERT, a purpose-built pre-trained transformer model, is utilized to generate contextual representations of tweet text, capturing the intricate nuances of meaning and sentiment. Additionally, the US Encoder enhances text processing capabilities. For the visual component, two prominent image embedding models, EfficientNet and CLIP, are employed. EfficientNet excels in extracting high-level semantic features from images, while CLIP leverages multi-modal learning to bridge the semantic gap between textual and visual representations. The combined application of these models facilitates a comprehensive understanding of the visual content associated with each tweet.

We combined these features with other features like date-time and inferred company names.

4.2 Model Selection and Training

XGBoost, a robust gradient boosting model renowned for its adeptness in handling diverse features, was employed in this study to discern intricate interactions within the dataset. The model's hyperparameter tuning process was guided by the validation loss, ensuring optimal performance.

Parameter	**Value**
`max_depth`	5
`learning_rate`	0.01
`subsample`	0.8
`colsample_bytree`	0.6
`tree_method`	`gpu_hist`
`predictor`	`gpu_predictor_T4`
`random_state`	*42*

Table 2: XGBoost Parameters

In addition to XGBoost, the LGBM Regressor, another gradient boosting model valued for its efficiency and scalability with large datasets, was incorporated. Similar to the XGBoost implementation, the hyperparameters of the LGBM Regressor were fine-tuned based on the validation loss. We took the logithim of likes as our target column.

An ensemble strategy was adopted to enhance predictive capabilities. Rather than relying on a singular model, the ensemble approach weighted predictions from each regressor according to their respective validation losses. This nuanced averaging of strengths and mitigation of individual weaknesses aimed at improving overall prediction accuracy and generalization.

4.3 Results

Model	**Validation RMSE on log(1+likes)**
Model 1	0.1153
Model 2	0.1708

Table 3: RMSE on Validation for Model 1, Model 2, and Combined Ensemble.

Model 1, a fusion of CLIP with MPNet, achieved an RMSE of 0.1153 on validation, showcasing strong performance in capturing intricate textual and visual relationships. In contrast, Model 2, combining EfficientNet and DistilBERT, yielded an RMSE of 0.1708, leveraging the efficient scaling of convolutional networks for images and distilled knowledge from BERT for text. This RMSE was measured on the logarithm of likes.

The ensemble of Model 1 and Model 2 demonstrated robust results, blending their complementary strengths to capture diverse aspects of the data, contributing to overall model performance. Inference took 33 seconds and 31 seconds on the given test set.

5 Task 2: Content Simulation2

Given the tweet metadata (company, username, media URL, timestamp), generate the tweet text.

5.1 Utilizing the Media Data

Recognizing the value of visual information, we employed BLIP2, a powerful image captioner, to automatically generate textual descriptions of the downloaded images. This provided us with a rich source of additional features beyond the original tweet text. These features captured the visual aspects of the content, allowing us to incorporate them into the LLM for more contextually relevant and engaging content generation.

5.2 Prompt Engineering

In guiding the Language Model (LLM) toward specific content creation objectives, we employed prompt engineering techniques. Detailed prompts were crafted, incorporating elements such as the image caption from BLIP2 for rich visual context, date-time information for temporal relevance, the initial number of likes as an indicator of audience preference, and the inferred company name for additional brand context. This amalgamation aimed to provide the LLM with a holistic understanding of desired content characteristics, facilitating the generation of tweets that align with brand identity, engage the audience, and fulfill specific content creation goals.

5.3 Model Selection and Training with LoRA and PEFT

For fine-tuning Bloom 7b, a powerful language model with 176B parameters, we employed two advanced techniques. Firstly, PEFT (Prefix-Tuning with Early Fine-tuning) was applied, targeting the early layers of the model with domain-specific data while keeping later layers more generalizable. This approach leveraged the model's pre-trained knowledge, adapting it specifically to the context of social media content generation. Secondly, LoRA (Low-Rank Adaptive Weights) was utilized to enhance efficiency during fine-tuning by reducing the number of parameters updated, resulting in improved training efficiency and potentially faster model adaptation to new data.

Configuration	Value
`r (Attention heads)`	16
`lora_alpha`	32
`lora_dropout`	0.05
`bias`	`none`
`task_type`	`CAUSAL_LM`

Training Arguments	Value
`per_device_train_batch_size`	4
`gradient_accumulation_steps`	4
`warmup_steps`	100
`max_steps`	550
`learning_rate`	2×10^{-4}
`fp16`	`True`
`logging_steps`	1
`output_dir`	`outputs`

Table 4: Model Configuration and Training Arguments

5.4 Results

The finetuned Bloom 7b model achieved a loss of 1.8126 after 1100 iterations over the data, showcasing its effectiveness in predicting text. On the test dataset it took about 8.1 seconds per iteration.

Input Prompt	Output Prompt
CBC with username CBCManitoba posted a tweet at 2018-10-01 15:09:14 with an image that is a bride and groom on bicycles with a man in a cowboy hat with a number of likes 41	The bride and groom are riding their bikes to their wedding in Manitoba.

Table 5: Example Input and Output Prompts for the Bloom 7b Model.

6 Image Generation Strategy

Our image generation involves a two-step strategy. Firstly, prompt engineering guides the Stable Diffusion XL model to align with specific content goals, generating images seamlessly integrated with tweet narratives to engage the target audience.

Next, we fine-tune the Stable Diffusion XL model using Low-Rank Adaptive Weights (LoRA). This advanced technique adjusts parameters, enhancing training efficiency and adapting to social media nuances. The synergy of prompt engineering and LoRA fine-tuning optimizes content relevance and engagement, producing visually compelling images tailored to surpass desired metrics.

Future work may explore techniques like neural architecture search or ensemble methods to further enhance image generation efficiency and performance.

Limitations

Our current set of regressors and embedders do not capture the full context of audio or video data. This limitation could be addressed by incorporating multimodal encoders into our model, coupled with increased computational power and time for training.

Additionally, the Language Model (LLM) we employ exhibits a certain level of slowness during inference and demands substantial computational resources. To mitigate this limitation, optimization strategies such as model pruning, quantization, or employing more efficient hardware can be explored. Further improvements may also be achieved with algorithmic enhancements and parallelization techniques.

References

a. Efficientnet pytorch model zoo. https://pytorch.org/hub/nvidia_deeplearningexamples_efficientnet/.

b. Efficientnet tensorflow hub model. https://blog.tensorflow.org/2019/02/effective-tensorflow-20-best-practices.html.

Google ai blog. https://blog.research.google/.

Hugging face model hub. https://huggingface.co/models?language=ai.

Papers with code. https://paperswithcode.com/.

Askaydevs. 2019. Distilbert github repository. https://github.com/askaydevs/distillbert-qa.

Authors. 2022a. Bit: Learning deep representations for image and text retrieval. *Google AI*.

Authors. 2022b. Bloom: Language models are multi-modal and many-modal. *BigScience*.

Authors. 2022c. Lora: Low-rank optimization for efficiently encoding large language models. *Google AI*.

Authors. 2022d. Point-efficient transformers. *Google AI*.

beta-eto code. 2022. Blip2 github repository. https://github.com/beta-eto-code/bx.model.

bigscience workshop. 2022. Bloom 7b github repository. https://github.com/bigscience-workshop/model_card.

dmlc. Xgboost github repository. https://github.com/dmlc/xgboost.

Hugging Face. 2023a. Blip2 hugging face model card. https://huggingface.co/docs/transformers/main/model_doc/bit.

Hugging Face. 2023b. Distilbert hugging face model card. https://huggingface.co/docs/transformers/model_doc/distilbert.

microsoft. Lightgbm github repository. https://github.com/microsoft/LightGBM.

Microsoft. Lora github repository. https://github.com/microsoft/LoRA.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert for efficient inference. *Google AI*.

Mingxing Tan and Quoc V. Le. 2020. Efficientnet: Rethinking model scaling for convolutional neural networks. *Google AI*.

A References

- DistilBert:
  - Paper: (Sanh et al., 2019)
  - GitHub repository: (Askaydevs, 2019)
  - Hugging Face model card: (Face, 2023b)
- BLIP2:
  - Paper: (Authors, 2022a)
  - GitHub repository: (beta-eto code, 2022)
  - Hugging Face model card: (Face, 2023a)
- CLIP:
  - Paper: (?)
  - GitHub repository: (?)
  - Hugging Face model card: (?)
- Universal sentence encoder:
  - Paper: (?)
  - TensorFlow Hub model: (?)
- LORA:
  - Paper: (Authors, 2022c)
  - GitHub repository: (Microsoft)
  - Hugging Face model card: (Not yet available)
- PEFT:
  - Paper: (Authors, 2022d)
  - GitHub repository: (Not yet available)
  - Hugging Face model card: (Not yet available)
- Bloom 7b:
  - Paper: (Authors, 2022b)
  - GitHub repository: (bigscience workshop, 2022)
  - Hugging Face model card: (Not yet available)
- EfficientNet:
  - Paper: (Tan and Le, 2020)
  - TensorFlow Hub model: (eff, b)
  - PyTorch model zoo: (eff, a)
- XGBoost:
  - Website: (?)
  - Documentation: (?)

- GitHub repository: (dmlc)

- LightGBM:
  - Website: (**?**)
  - Documentation: (**?**)
  - GitHub repository: (microsoft)

- Additional resources:
  - Hugging Face model hub: (hug)
  - Papers with Code: (pap)
  - Google AI Blog: (goo)