Behaviour Simulation Challenge

InterIIT Tech Meet 12.0

Team ID 80

Final Report

December 2023

Introduction

This report focuses on developing a system that predicts user engagement (likes) given a tweet, time posted, and username, and one that generates content likely to elicit desired key performance indicators (KPIs) from the audience when presented with a user query. This report encapsulates an overview of the methodologies and approaches harnessed to tackle the challenges of behavior simulation (Task-1) and content simulation (Task-2).

1. Behaviour Simulation

The primary objective of the initial task pertains to the anticipation of the number of likes based on the examination of tweets, including accompanying metadata and media content. The pipeline is demonstrated in Figure 1. Our initial goal is to solidify the textual and media input that machine learning models can comprehend using embeddings and to represent the other features (dates, views, duration, company etc.) in a suitable format. We then use a regression model to predict likes.

A. Main Approach

1.1 EDA

An examination of tweet frequency in relation to the number of likes revealed a concentration of instances with 0 likes. We excluded data points corresponding to these specific like counts to rectify this imbalance and employed a manual binning approach to identify and eliminate additional outliers. This approach aimed to achieve a more uniform distribution of likes and enhance the robustness of our dataset. Similar strategies were applied to cleanse data pertaining to company names and the months and days of tweet postings, ensuring a balanced representation across these parameters. The resulting dataset, comprises 1,61,099 instances. The meticulous curation process examined the correlation between various data points, including 'word count,' 'mention count,' 'hyperlink count,' and 'hashtag count,' with the number of likes.

1.2 Metadata

1.2.1 Time

Sine and Cosine Embeddings: *'day_sin,' 'day_cos,' 'year_sin,' 'year_cos'* encode cyclical patterns for the time of day and year from the 'datetime' column. Uptime, the time since the tweet, is calculated. Day of the week is encoded with sine and cosine functions *('week_sin' and 'week_cos')*, enhancing temporal pattern capture.

1.2.2 Inferred Company

We used Word2Vec to encode the company names. This will ensure that similar companies will have similar encodings. E.g., Toyota, and Porsche will have similar encodings.

1.2.3 Additional Features

We also generate additional metadata from the tweet's content, including the number of mentions, hashtags, hyperlink counts, and the duration and number of video views from the video link.
We generate the sentiment of each tweet using an off-the-shelf VADER sentiment analysis tool from NLTK to assess the sentiment of tweets.

1.3 Tweet

1.3.1 Tweet Content

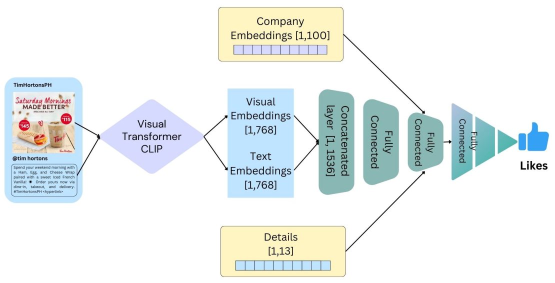
Company Embeddings [1,100]
TimHortonsPH
Saturday Mornings MADE BETTER
@tim hortons
Spend your weekend morning with a Ham, Egg, and Cheese Wrap paired with a sweet Iced French Vanilla! Order yours now via dine-in, takeout, and delivery. #TimHortonsPH <hyperlink>
Visual Transformer CLIP
Visual Embeddings [1,768]
Text Embeddings [1,768]
Concatenated layer [1, 1536]
Fully Connected
Fully Connected
Fully Connected
Likes
Details [1,13]

Figure 1 – Model Pipeline for Task 1

Our primary objective is to create embeddings for textual input in machine learning models. Initially, we combined tweet content, inferred company, and username for input. We used embedding methods like Google's universal sentence encoder and BERT, which were insufficient for capturing cross-semantic relationships between text and media in tweets [2].
We tried out **BLIP (Bidirectional Language-Image Pre-training)** and **CLIP (Contrastive Language–Language-image pre-training)** to address this limitation. We chose CLIP based on their superior results on the training dataset and optimal inference time. CLIP trains image and text encoders jointly, learning a multimodal embedding space by predicting pairings in a batch. It ensures that the image and text embeddings share a common embedding space, which was our initial goal [7].

1.3.2 Media Content

We initially used VGG19 transfer learning for Images to derive the features from tweet images by extracting features from the final layer. We also tried BLIP and CLIP to generate embeddings. However, we adopted CLIP for image embeddings to improve model performance by aligning image and text embeddings in the same space [5]. For videos, we created keyframes, sampled frames, and concatenated their features into a sequence for generating video embeddings [4]. However, due to high inference time and poor model performance, we ended up using just the thumbnails for each video to generate the embeddings.

An attempt was made to extract audio from videos and employ the Whisper (large) model for transcribing text and generating captions. However, this approach was excluded from the final solution owing to elevated inference times encountered per video.

1.4 The Multimodal Model

The model architecture consists of two parallel pathways for processing textual and image embeddings, each followed by dense layers with varying dimensions and activation functions. Additionally, numerical features are incorporated through a separate input pathway. These three pathways are concatenated to form the model's input tensor fed through additional dense layers to capture complex relationships within the data. The model is trained using a custom dataset generator implementing the Keras Sequence class, facilitating efficient loading of large datasets. A learning rate scheduler optimizes training and dynamically adjusts the learning rate during training epochs.

Other Approaches

1. **ResNet101-BERT-Time Model** [3] In this approach, the features of the image were extracted using ResNet101 model pretrained on ImageNet, while the tweet content was embedded using BERT model and features such as time, NIMA score of image, views and duration of the video were taken into account.

2. **VSCNN** [1] This model uses Transfer learning from VGG19 model to extract image feature along with image embeddings made with google universal encoder and meta features all concatenated together followed by dense network.

3. **Weighted Loss** [6] As already mentioned the dataset was highly biased, plotting it revealed it to be of the form

$$f(x) = e^{-x}$$

where x is the number of likes and $f(x)$ is the frequency of tweets corresponding to x likes. To counteract the effect, we tried training our regression network using a weighted loss, where the weights for the rmse score were taken as following

$$w_y = 1 - e^{-y} + \epsilon$$

where y is the target value and ϵ is a term to account for frequency of zero likes frequency.

Results and Conclusion

The results on the training set were quite unsatisfactory. This was because of the high sensitivity of likes to the follower count/verified status of the user, which was not being taken into account. Scraping could be used to fetch the user's followers and boost the accuracy, but that way proved to be financially expensive.

2. Content Simulation

In the second task, we aim to generate tweet text based on the provided metadata attributes, including company affiliation, username, media URL, and timestamp. This investigation focuses on the synthesis of textual content from essential tweet-related metadata.

2.1 Prompt

Our primary aim was to generate a prompt, which contained maximum features and relevant information of the tweet data that could help generate tweets as accurately as possible.

2.1.1 Image Captioning

We systematically evaluated diverse image captioning methodologies, including BLIP, BLIP2, Fastdup, GPT2, and MiniGPT4 models, we conducted rigorous experimentation to ascertain the model yielding optimal results and performance. Table 1 summarizes the performance of the models. Following extensive trials, our findings unequivocally identify the Fastdup model as the superior choice in image captioning, substantiating its superior efficacy and performance compared to alternative approaches.

Table 1 – Processing Times for Different Models

Model	Time	Model	Time
FastDup	125 ms	BLIP	590 ms
MiniGPT4	26 sec	BLIP2	501 ms
GPT2	2.47 sec		

Note: Inference times are per image

2.1.2 Video Captioning

Three approaches were explored. The first utilized BLIP for embeddings, which were then used for caption generation. The second, a more comprehensive strategy, involved denser video captioning. Keyframes were initially extracted, and their feature mean was captioned using BLIP, but this yielded subpar descriptions. We then used FastDup-BLIP on all keyframes, feeding the captions into a Language Model (LLaMa 2) to derive a dense caption for the entire video. Despite superior captions, this approach had high computational time and reduced tweet similarity scores. Ultimately, we opted to use only the video thumbnail for caption generation using FastDup-BLIP in the tweet prompt. Results are tabulated in Table 3.

2.1.3 Audio Captioning

For audio captioning, initial attempts using the MoviePy library and Whisper ASR models resulted in high inference times. Consequently, we reconsidered the methodology due to computational inefficiencies.

2.1.4 Prompt Engineering

The primary objective was to formulate a prompt that encompasses the maximum features of tweet data, aiming to assist the language model (LLM) in generating tweets with high accuracy. We explored various strategies, including incorporating multi-modal data (as outlined in sections 3.1.2 to 3.1.4) and integrating tweet metadata. Despite initial attempts to strengthen the prompt by emphasizing metadata, we found that this method did not yield highly accurate tweet generation. In an effort to enhance tweet accuracy, we experimented with providing relevant examples within the prompt, carefully selected to offer valuable insights to the LLM for tweet generation. This approach notably improved the similarity scores for the generated tweets. Additionally, we explored the use of dynamic examples in the prompt, where examples are automatically chosen based on the image and image caption.

2.2 Model for Tweet Generation

Various tweet generation models were tested, including open-source LLMs like LLaMa2 and Vicuna. However, these models exhibited longer generation times, lower evaluation scores, and posed deployment challenges due to their large size.

Subsequently, we transitioned to closed-source LLMs, specifically gpt-3.5-turbo and gpt-4-turbo. The closed-source models provided faster inference, accurate results, and ease of execution through API calls without the need for deployment. The gpt-3.5-turbo was selected as our generation model based on its superior similarity scores over an average of results over 20 samples, as detailed in Table 2.

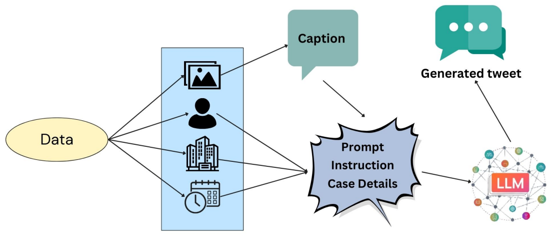

Figure 2 – Comparison of Inference time and similarity score

Future Work

In our current prompt construction strategy, we select examples from the training dataset. To address this limitation, we propose a novel approach involving dynamic input extraction from the training dataset, focusing on samples most likely related to the media content within the prompt, such as images or thumbnails. Our proposed methodology advocates for clustering the training data using embeddings from both tweet content and images. Representative examples from each cluster, reflecting thematic relevance, are then compared for similarity with the provided testing example through embeddings derived from the testing image and training content and image. Those training examples exhibiting the highest similarity are incorporated into the prompt, aiming to enhance its specificity and contextual relevance. This iterative refinement process leverages the nuanced relationships between textual and visual elements in the training dataset, holding promise for optimizing the model's performance in the targeted domain.

Table 2 – Comparison of Different Video Captioning Methodologies

Methodology	BLEU Score	ROUGE Score	Time for Inference
Thumbnail Captioning (fasdup blip)	0.221	0.16	0.15 seconds
Concatenate Output of Key Frames	0.231	0.15	12 seconds
Mean of Key Frames (fasdup blip)	0.254	0.16	9 seconds
Average of Embeddings of Key Frames (blip embeddings)	0.214	0.16	3 seconds
GPT-2 Video Captioning	0.25	0.14	96 seconds

References

[1] Fatma S. Abousaleh, Wen-Huang Cheng, Neng-Hao Yu, and Yu Tsao. Multimodal deep learning framework for image popularity prediction on social media. *IEEE Transactions on Cognitive and Developmental Systems*, 13(3):679–692, 2021.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*, 2019.

[3] Keyan Ding, Ronggang Wang, and Shiqi Wang. Social media popularity prediction: A multiple feature fusion approach with deep neural networks. In *Proceedings of the 27th ACM International Conference on Multimedia*, MM '19, page 2682–2686, New York, NY, USA, 2019. Association for Computing Machinery.

[4] Junnan Li, Dongxu Li, Caiming Xiong, and Steven C. H. Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International Conference on Machine Learning*, 2022.

[5] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR, 2021.

[6] Divish Rengasamy, Mina Jafari, Benjamin Rothwell, Xin Chen, and Grazziela P. Figueredo. Deep learning with dynamically weighted loss function for sensor-based prognostics and health management. *Sensors*, 20(3), 2020.

[7] Ke Wang, Mohit Bansal, and Jan-Michael Frahm. Retweet wars: Tweet popularity prediction via dynamic multimodal regression. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1842–1851, 2018.