CSCI-SHU 376 NLP Project Report: Meme Caption Generation

Yuchen Wang*
NYU Shanghai
yw3642@nyu.edu

Yichen Huang*
NYU Shanghai
yh2689@nyu.edu

Abstract

[1] Meme image captioning is a task with challenges distinct from those in conventional image captioning. In this project, we propose a pipeline generating formatted meme captions based on image input. Our pipeline consists of an encoder-decoder with a modified middle layer, a CLIP-model-assisted decoding strategy and a line break inserter. We train and test our model on self-scrapped data and demonstrate that our method outperforms the standard encoder-decoder approach.

1 Introduction

A meme is "an image, video, piece of text, etc., typically humorous in nature, that is copied and spread rapidly by internet users, often with slight variations"[5]. Memes are ubiquitous in today's online mediums, often commenting on cultural symbols, social ideas, or current events. In this project, we focus on memes where one picture is captioned with two lines of texts (Figure 1), a popular type of meme that is easily created and propagated. Given a meme image, we attempt to generate captions that capture the style and are specific to the input image. This task is significant for it involves not only a high level of visual and textual understanding but also a strong sense of creativity and diversity in the generation process.

We propose a pipeline consisting of an modified encoder-decoder model, a decoding strategy and a line break inserter. We then evaluate the performance of each component, both qualitative and quantitatively. We demonstrate that our method is superior to a transformer-based encoder-decoder baseline.

* equal contribution

[1] Our implementations: https://github.com/Zacchaeus14/CSCI-376-Project-Implementation

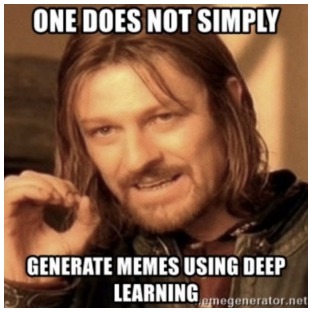

Figure 1: An example of the memes we work on

1.1 Challenges

Due to the unique features of meme images and captions, our task poses a distinct set of challenges compared with conventional image captioning. We discuss such challenges in the following section.

The image feature extractor should retrieve **higher-level representations**. In conventional image captioning, the desired captions mostly identify the most salient objects in the images and describe their relationships. By contrast, meme captions are often based on more subtle features, like facial expressions and body languages, that inspires a sense of humor.

The model should possess **prior knowledge**. Since most memes refer to popular cultures, it will be helpful for the model to have prior knowledge of the recurring tropes in such cultures.

Meme pictures and captions constitutes a **one-to-many mapping** where one image is mapped to multiple valid sets of captions (Figure 2). Since meme captions are not based on concrete physical features, it is common for an image to have more than 100 reference captions. Such captions usually share common features, including topics and the use of specific n-grams.

Each caption's **specificity** to the input image and **diversity** is more difficult to assure and measure since there are many high-frequency phrases

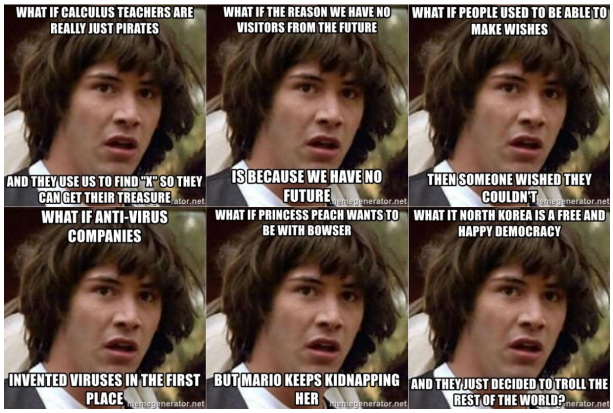

Figure 2: The Conspiracy Keanu meme with different captions. The captions share the same topic (conspiracy) and style (starting with "what if").

associated with almost all meme images.

The **line breaks** contain significant semantic meanings. In our interested format of captions, the upper part of the text often sets up the joke, whereas the lower part typically functions as the punchline. The arrangement of these two parts significantly influences the humorous effect and the quality of the generated captions.

2 Related Work

2.1 Encoder-Decoder Models for Image Captioning

Current image captioning methods mostly utilizes encoder-decoder models where the input image is encoded into a latent through an image encoder, which is then fed into a text decoder. The text decoder functions similar to a language model where the next output token is influenced by the previous tokens and the input image [2, 4].

Our work is inspired by Dank Learning [1], a system released in 2018 specializing in meme captioning. It utilizes a ResNet-based image encoder and an LSTM-based text decoder to generate captions similar in format as ours. Its generated results are passable in style. However, we note that it frequently outputs captions that are highly frequent but unspecific to the given image, which is an issue we aim to address.

2.2 CLIP

Proposed by Radford et al.[7] at OpenAI, CLIP (Contrastive Language-Image Pre-Training) is a neural network trained on a variety of (image, text) pairs. It can be instructed in natural language to predict the most relevant text snippet, given an image, without directly optimizing for the task, similarly to the zero-shot capabilities of GPT-2 and 3. CLIP's design of effectively connecting images and text brings about its proximity with our project. Also, as we observe, CLIP has three advantages. First, it captures high-level features of images and text. Second, it utilizes prior knowledge learned from pre-training. Third, it generalizes well on unseen data.

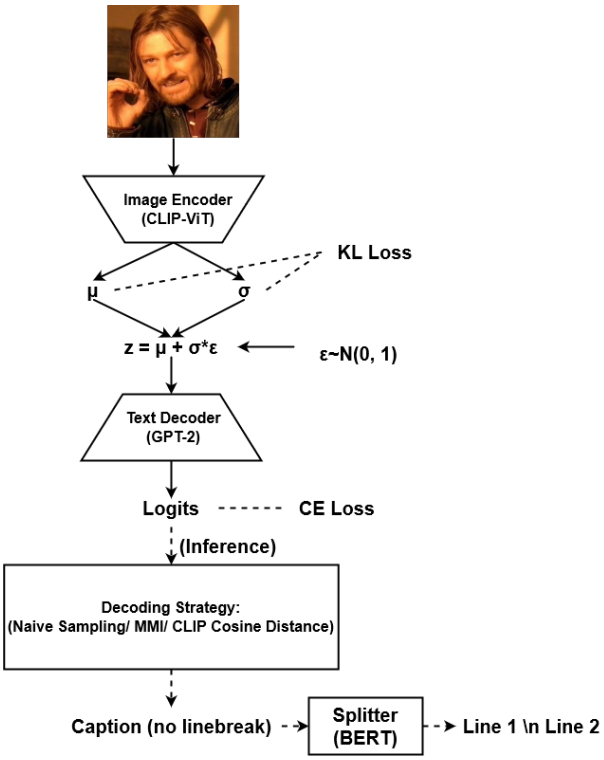

Figure 3: A diagram of the proposed pipeline.

3 Method

Our pipeline consists of an encoder-decoder model that generates probabilities for each word token given the input images and the previous tokens, a decoding strategy that select tokens given such probabilities, and a splitter that identifies the position to insert the line break given the text token sequence. We alter the standard approach for the encoder-decoder model by introducing sampling through reparameterization in the bottleneck layer, similar to that in the typical variation autoencoder. For decoding, we explore three strategies: naïve sampling, maximum mutual likelihood decoding and CLIP score ranking. For the splitter, we finetune a BERT model to predict the index of the last token before the line break as a classification task.

3.1 Encoder-Decoder

3.1.1 CLIP Image Encoder

For the image encoder, we use a vision transformer pre-trained as a part of the CLIP model. The encoder generates a hidden vector h_{img} give an image.

3.1.2 The Bottleneck Layer

Instead of directing passing h_{img} into the text decoder, we adopt an approach commonly seen in variational autoencoders. We use separate linear transformations to obtain a mean μ and log variance $\log \sigma^2$ from h_{img}. We then use the reparameterization trick to acquire a latent z sampled from a normal distribution with corresponding mean and log variance as shown in the formula below. The intuition behind this approach is that the sampling process helps the model capture the one-to-many mapping between the images and the captions.

$$z = \mu + \sigma\epsilon, \epsilon \sim \mathcal{N}(0, 1)$$

3.1.3 GPT-2 Text Decoder

The decoder is based on a pre-trained distilgpt2 model. The model produces a hidden vector h_{txt} based on previous word tokens. We then add h_{txt} and z before passing them into a linear language model head to produce a logit for each word token in our vocabulary.

3.1.4 Training

The encoder-decoder model is trained with a hybrid loss consisting of the cross-entropy loss between the generated captions and a reference caption and the Kullback Leibler divergence between the learned latent distribution and the standard normal distribution. For the CE loss, we consider only one reference caption at each pass.

$$Loss = CE(\hat{y}, y) + \beta D_{KL}(q(z|x)||\mathcal{N}(0, 1))$$

3.2 Decoding Strategies

Aside from the naïve sampling approach where we input the logits into a softmax function and directly sample from the probabilities, we explore two other approaches: Maximum Mutual Likelihood Decoding and CLIP Score Matching with the aim of encouraging diversity and specificity to the input image.

3.2.1 MMI Decoding

Proposed in [3], MMI decoding maximizes the mutual information between the image and captions. We choose the MMI-AntiLM approach proposed by the original authors where we maximize $\log P(Cap|Img) - \lambda \log P(Cap)$ where the prior $P(Cap)$ is obtained using an unmodified pre-trained GPT-2 language model with the same vocabulary as ours. We then perform beam search based on the MMI scores. During our experiments, we have found that this approach leads to incoherent outputs containing mostly high-frequency words (like "pizza") (Figure 4). For our purpose, such an approach seems outperformed by naïve sampling and hence we do not include it in our evaluations.

```
pizza  in in  a on and and,, a even a  and and your and and
pizza  in in  a on and and,, a even a a  and and your and
pizza  in in  a on and and,, a even a a  and and your a
pizza  in in  a on and and,, a even a  and and your and in
pizza  in in  a on and and,, a even a your a  and and and
pizza  in in  a on and and,, a even a  and and your the a
pizza  in in  a on and and,, a even a  and and your a in
pizza  in in  a on and and,, a even a  and and your and a
pizza  in in  a on and and,, a even your a a  and and and
pizza  in in  a on and and,, a even a  and and your a a
pizza  in in  a on and and,, a even a  and and your and
pizza  in in  a on and and,, a even a your a  a and and
pizza  in in  a on and and,, a even a  and your and and and
pizza  in in  a on and and,, a even a  and your and and in
pizza  in in  a on and and,, a even a your a  and and in
```

Figure 4: Top outputs using MMI decoding where beam size $= 50$ and $\lambda = 0.3$.

3.2.2 CLIP Score Matching

We introduce a method to rank captions based on their compatibility with a given image. We fine-tune a pair of CLIP image and text encoders to maximize the dot product between the image and text embeddings.

$$CLIPscore = h_{\text{img}} \cdot h_{\text{txt}}$$

We utilize contrastive training similar to that used in the original paper, where we maximize the CLIP score for matching image and texts and minimize that for mismatching ones. In the decoding process, we first generate K candidate captions using naïve sampling. We calculate the CLIP score between each candidate and the input image and choose the one with the top score.

3.3 BERT Caption Splitter

We fine-tune a pre-trained BERT model for caption splitting. For a batch B of sequences with length N, an additional linear layer projects the last hidden states with shape (B, D) to a B-shape

vector at each word position. The resulted (B, N)-shape vector $\hat{y}$ is passed into the cross-entropy loss with the label y of the same dimension for back-propagation. For each sample i in the batch, the label $\mathbf{y_i}$ is a one-hot vector where $\mathbf{y_{i,j}} = 1$ only if the sentence i splits after position j.

4 Data

4.1 Data Source

We fetch our data ourselves from the Internet with a `Python` scrapper script. The data source we adopt is `memegenerator.net`, where people upload new images or post meme captions on an existing image. We choose this website because it organizes memes in a scrapper-friendly way, and the quantity of the data is enough for our project. We fetch $n_I = 3000$ images, denoted as $\{I_i\}_{0 \le i \le n_I - 1}$. Each image I_i corresponds to $\min(u_i, n_C)$ most popular (measured by number of upvotes) captions, where u_i is the total number of captions available for image I_i and $n_C = 225$. Each caption $C_{i,j}$ contains one and only one line break, splitting the two lines. The resulted dataset $\mathcal{D} = \{(I_i, C_{i,j})\}_{\substack{0 \le i \le n_I - 1 \\ 0 \le j \le \min(u_i, n_C) - 1}}$.

4.2 Preprocessing

4.2.1 Preprocessing Image

All images fetched from `memegenerator.net` have resolution 300×300 and are RGB. Before feeding them into the CLIP Encoder, we use a built-in preprocessor provided in CLIP's official source code. The preprocessor executes the following precedures sequentially:

1. Resizing the images to 224×224 to fit the CLIP model;
2. Cropping the images at center, which is trival on our dataset where all images are quadrate;
3. Converting the images into RGB, which is trival on our dataset where all images are RGB;
4. Convert the images to `PyTorch Tensor`;
5. Normalizing the images with parameters mean μ[2] and standard deviation σ[3].

[2] $\mu = (0.48145466, 0.4578275, 0.40821073)$
[3] $\sigma = (0.26862954, 0.26130258, 0.27577711)$

4.2.2 Preprocessing Text

The captions from `memegenerator.net` are noisy. For instance, some samples are non-English, etc. we adopt a cleaning pipeline to remove three types of captions:

- Captions that include characters that do not belong to the ASCII Table;
- Captions with length beyond a preferred range $[min\_len, max\_len]$[4];
- Captions that include words which are exceedingly long[5];
- Captions without any alphabetical character.

The cleaned dataset has 13% fewer captions. They are subsequently tokenized by different tokenizers depending on the model.

5 Experiments and Evaluation

We evaluate the performance of four methods: a baseline model with standard encoder-decoder structure (no sampling in the bottleneck layer) decoded using naïve sampling, our purposed encoder-decoder model respective with naïve sampling and CLIP score matching and finally, a sampling-only pipeline where we use CLIP score matching to find the most suitable caption from our training set. Our training/validation/test split in terms of the number of images is 2400:300:300. The encoder-decoder, fine-tuned CLIP model for decoding and the BERT splitter are trained separately. We evaluate the overall quality of the generated captions using BLEU-1, BLEU-2 and BLEU-3 [6], all with uniform weights. We also evaluate the diversity using unigram and bigram counts where we generate one caption for each of the 300 test images and count the unique unigrams and bigrams (Table 1).

Our results show that the proposed model with CLIP score matching achieves the best BLEU scores. We note that the BLEU scores, especially BLEU-3, is relatively low for all approaches compared to top results in conventional image captioning, indicating that none of the evaluated approaches can perfectly predict more extended meme captions. In terms of diversity, all approaches are on a comparable level except the sampling and CLIP score matching approach,

[4] $min\_len = 3, max\_len = 128$
[5] threshold is 16

Method	BLEU-1	BLEU-2	BLEU-3	Unigram Count	Bigram Count
Baseline	0.479	0.164	0.046	**2505**	**1335**
Proposed (Naive sampling)	0.491	0.158	0.050	2439	1229
Proposed (CLIP score matching)	**0.524**	**0.183**	0.071	2299	1249
Sampling + CLIP score matching	0.362	0.143	**0.077**	1359	85

Table 1: BLEU scores and n-gram counts for the entire pipeline.

Method	Precision
Finetuned CLIP	**0.545**
Unmodified CLIP	0.116

Table 2: Precision of the finetuned CLIP model used in the decoding. We tested the model on 1000 images. For each image, the model should identify 1 matching caption from 127 irrelevant captions.

Method	Precision
Finetuned BERT	**0.823**
Naive	0.196

Table 3: Precision of the finetuned BERT caption splitter. We compare our model with the naive approach where we always insert the line break in the middle of the caption.

which scores notably low. Qualitatively, we found that the sampling and CLIP score matching generates the best-quality results with coherent and relevant captions. Thanks to the finetuned CLIP model, the approach generalizes very well to unseen images. The obvious downside of this approach is the limited diversity due to the lack of any generation process.

The proposed model with CLIP score matching generates satisfactory results that are usually limitedly coherent and only captures a general sentiment conveyed by the input image. We attribute the limit in its performance to the noises in our dataset and the challenging training process.

6 Conclusion and Future Work

We have introduced a complete pipeline that generates formatted meme captions based on image inputs. We demonstrated that our approach outperforms the standard encoder-decoder model. We expect further performance improvement if we apply better datasets and training. In particular, we note that our dataset is extremely noisy with low-quality or even grammatically incoherent reference captions. We also observe inappropriate elements like swearing and racism in our dataset. These challenges suggest a need for better data sources and stricter data cleaning.

Figure 5: Example outputs using the sampling and CLIP score matching method.

To further improve our approach, since we find that one image can correspond to captions of different topics, clustering the captions by topic and applying conditional generation can be a way to address the one-to-many mapping problem and introduce controllability. We also see ways to apply our approach to more complex examples with multiple sub-images and text fields.

Figure 6: Example outputs using the proposed encoder-decoder with CLIP score matching.

References

[1] Abel L Peirson V au2 and E Meltem Tolunay. *Dank Learning: Generating Memes Using Deep Neural Networks*. 2018. arXiv: 1806.04510 [cs.CL].

[2] Marcella Cornia et al. *Meshed-Memory Transformer for Image Captioning*. 2020. arXiv: 1912.08226 [cs.CV].

[3] Jiwei Li et al. *A Diversity-Promoting Objective Function for Neural Conversation Models*. 2016. arXiv: 1510.03055 [cs.CL].

[4] Xiujun Li et al. *Oscar: Object-Semantics Aligned Pre-training for Vision-Language Tasks*. 2020. arXiv: 2004.06165 [cs.CV].

[5] *MEME: Definition of MEME by Oxford Dictionary on Lexico.com also meaning of MEME*. URL: https://www.lexico.com/definition/meme.

[6] Kishore Papineni et al. "Bleu: a Method for Automatic Evaluation of Machine Translation". In: *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, July 2002, pp. 311–318. DOI: 10.3115/1073083.1073135. URL: https://www.aclweb.org/anthology/P02-1040.

[7] Alec Radford et al. "Learning Transferable Visual Models From Natural Language Supervision". In: *CoRR* abs/2103.00020 (2021). arXiv: 2103.00020. URL: https://arxiv.org/abs/2103.00020.