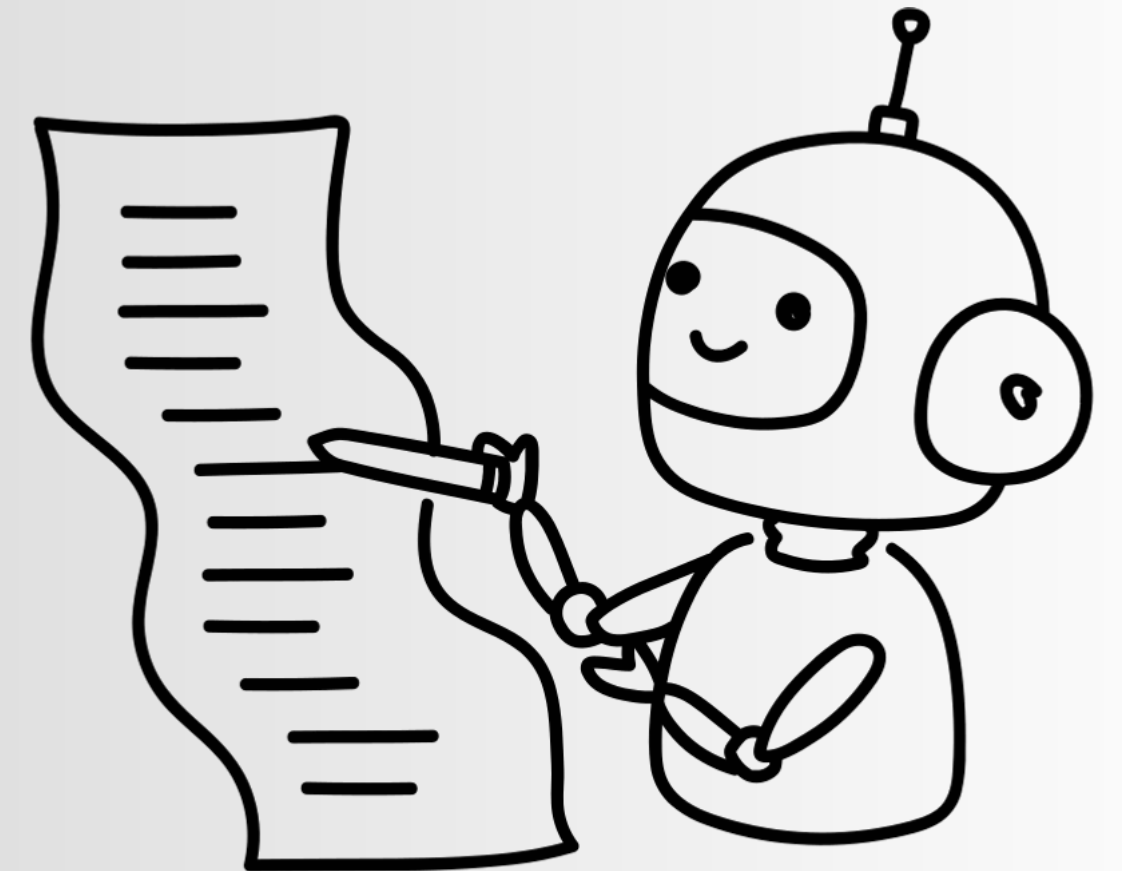


Image Captioning using Hugging Face Vision Models

B **L** **I** **P**



22224516 Gurung Yash
22224014 Khatri Priya
22224040 Sagar Bhusal

Table of Contents:

- Introduction
- Machine Learning Model
- Dataset Overview
- Methods and Workflow
- Correlation Matrix
- Results and Insights
- Conclusion

Image Captioning with Hugging Face Model



In this project, we explore image captioning using Hugging Face's BLIP model. Both the dataset and model were imported from Hugging Face.

- **Dataset:** We used the flickr8k dataset
- **Model:** BLIP

Machine Learning Model

BLIP Model Overview

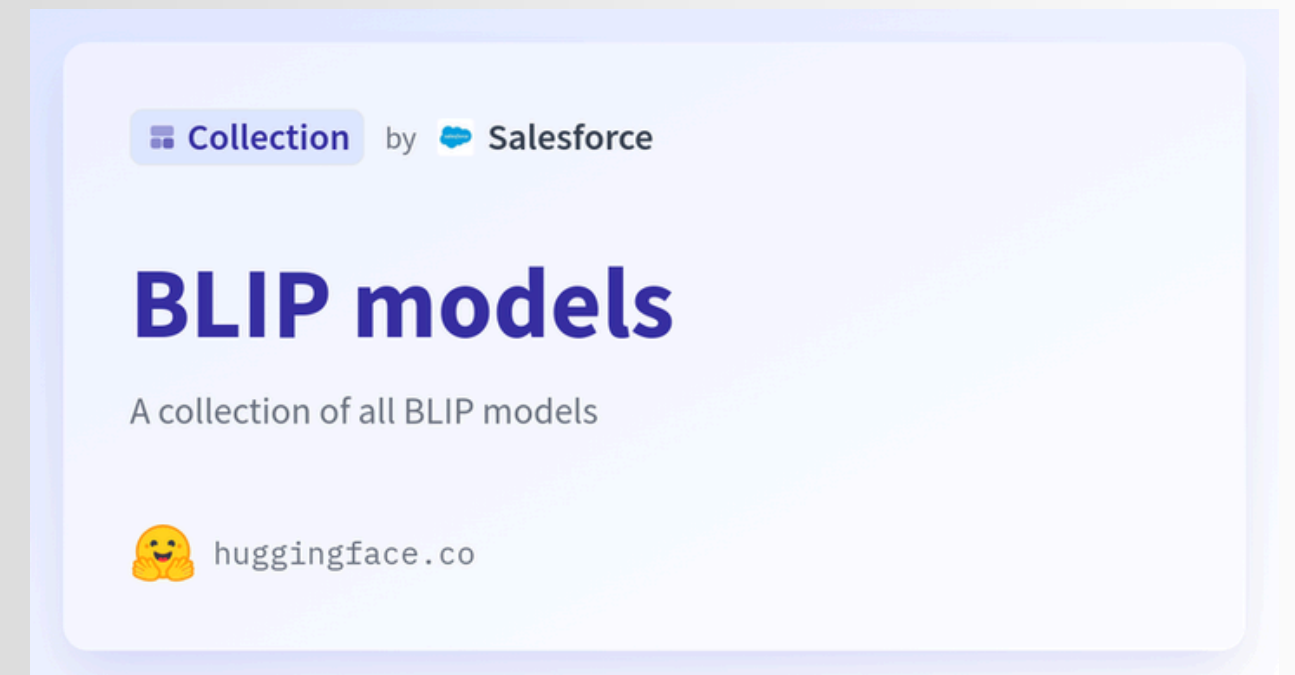
- **Creator:** The BLIP model was developed by Salesforce.

- **Purpose:**

BLIP (Bootstrapped Language-Image Pre-training) is a unified vision-language model designed for tasks like image captioning, image-text retrieval, and visual question answering.

- **Project Utility:**

BLIP's pre-trained framework enables us to efficiently generate captions for images by leveraging its state-of-the-art performance in vision-language tasks.



Key Features of BLIP

BLIP is a pre-training framework aimed at unified vision-language understanding and generation. It achieves state-of-the-art results across seven key vision-language tasks:

- Image-text retrieval
- Image captioning
- Visual question answering
- Visual reasoning
- Visual dialogue
- Zero-shot text-video retrieval
- Zero-shot video question answering

Methods and Workflow

1. Loading the Pre-trained BLIP Model

✓ Loading the pre-trained BLIP image captioning model.

```
[ ] # Use a pipeline as a high-level helper
    from transformers import pipeline

    pipe = pipeline("image-to-text", model="Salesforce/blip-image-captioning-large")
```

+ Code

+ Text

2. Data Preprocessing

✓ Pre-Process the Text

```
# Example of text preprocessing
def preprocess_text(text):
    # Lowercasing and other preprocessing if needed
    text = text.lower() # Convert to lowercase
    return text

# Apply preprocessing to the dataset (if needed)
dataset = ds.map(lambda x: {'text': preprocess_text(x['text'])})
dataset[:5]
```

make the text in the captions all in lower case

```
import re

def remove_special_characters_and_numbers(text):
    # Remove special characters and numbers, keeping only letters and spaces
    text = re.sub(r'^a-zA-Z\s', '', text)
    return text

# Apply the function to the dataset
dataset = dataset.map(lambda x: {'text': remove_special_characters_and_numbers(x['text'])})
dataset[:5]
```

Removing all the special characters from the captions

3. Dataset Sampling

✓ Performing BLIP image captioning model.

```
sample_ds = dataset[0:1200]
sample_ds

'a man lays on a bench while his dog sits by him ',
'a man in an orange hat starring at something ',
'a child playing on a rope net ',
'a black and white dog is running in a grassy garden surround
'a dog shakes its head near the shore a red ball next to it
```

4. Caption Generation

```
results = pipe(sample_ds['image'])  
results
```

- Generated captions using BLIP were treated as predicted_labels.
- Original captions were retained as true_labels for comparison.

```
true_labels = sample_ds['text'] # Actual labels (captions)  
  
# Predicted labels  
predicted_labels = [result[0]['generated_text'] for result in results]  
  
# Create a DataFrame to compare true and predicted labels  
comparison_df = pd.DataFrame({  
    'True Label': true_labels,  
    'Predicted Label Captions': predicted_labels,  
})  
comparison_df
```

5. Evaluation Using Confusion Matrix

```
!pip install scikit-learn  
import seaborn as sns  
import matplotlib.pyplot as plt  
from sklearn.metrics import confusion_matrix  
true_labels_simplified = ['Positive' if 'dog' in label.lower() else 'Negative' for label in true_labels]  
predicted_labels_simplified = ['Positive' if 'dog' in label.lower() else 'Negative' for label in predicted_labels]  
  
cm = confusion_matrix(true_labels_simplified, predicted_labels_simplified, labels=['Positive', 'Negative'])  
  
# Create the heatmap  
plt.figure(figsize=(8, 6))  
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",  
            xticklabels=['Positive', 'Negative'], yticklabels=['Positive', 'Negative'])  
plt.xlabel("Predicted Label")  
plt.ylabel("True Label")  
plt.title("Confusion Matrix")  
plt.show()
```

1. Simplification of Labels:

- true_labels_simplified and predicted_labels_simplified categorize labels as "Positive" if they contain the substring "dog" (case-insensitive); otherwise, they are categorized as "Negative."

2. Confusion Matrix Calculation:

- confusion_matrix computes the confusion matrix using the simplified labels and specifies the order of labels as ['Positive', 'Negative'].

HeatMap

Evaluation Metrics

1. BLEU:

BLEU (Bilingual Evaluation Understudy) is an NLP metric for tasks like machine translation. It Shows how much the generated text matches the reference text word-for-word.

Scores range from 0 to 1, with higher scores showing better similarity. However, it prioritizes precision over recall, so a high score doesn't always mean perfect quality.

2. CIDEr:

CIDEr (Consensus-based Image Description Evaluation) is a metric for image captioning that measures how similar a generated caption is to human-written references. It focuses on matching words that are rare and important, comparing them to human-written captions.

Scores range from 0 to 1, with higher scores indicating better alignment and descriptiveness compared to human references.

3. SPICE:

SPICE (Semantic Propositional Image Caption Evaluation) measures the quality of image captions by comparing their semantic content to human references. Shows how well the meaning or relationships in the generated caption match human-written captions.

Scores range from 0 to 1, with higher scores indicating better semantic alignment with human-written captions.

Evaluation Metrics (Scores)

1. BLEU:

2. CIDEr:

3. SPICE:

Conclusion