

Rapport TP5

NLP avec 'Attentions Models'

Ghandouz Amina G2

1. About the dataset

The Amazon Products Sales dataset 2023 from Kaggle was scraped from the Amazon site, it has 140 categories in csv format such as *Gaming Consoles*, *Car Electronics*, *All Electronics*, *All Books*, *Make-up*...etc

Each category has 9 columns which are:

- *name*: the product title.
- *main category & sub category*: hierarchical classification for the product.
- *image*: contains visual reference urls.
- *link*: offers direct product page access.
- *discount price*: pricing details (current offer).
- *actual price*: original price.
- *ratings*: customers feedback (average score).
- *no of rating*: review volume.

This dataset has on total 1,103,170 values with 551,585 missed values, after replacing all the missing values with the mean we got more than 551,000 duplicated rows, and due the executing time it took on tokenization step (64min) we decided to use only one category (*Home Dcor*) from the *Home_Dcor.csv* file, it contains 1,272 samples with 9 columns.

	name	main_category	sub_category	image	link	ratings	no_of_ratings	discount_price	actual_price
0	RAG28 Wooden Wall Decor Set of 11 (SPW1) Multi...	home & kitchen	Home Décor	https://m.media-amazon.com/images/I/81GQf14cn...	https://www.amazon.in/RAG28-Wooden-Multicolour...	3.9	2,657	₹284.05	₹799
1	ArtX Paper Motivational Wall Frames, Inspirati...	home & kitchen	Home Décor	https://m.media-amazon.com/images/I/61VDPBbPgH...	https://www.amazon.in/ArtX-Motivational-Quotes...	4.4	572	₹593	₹1,399
2	Mahogany Life Vintage Glass Hanging Photo Fram...	home & kitchen	Home Décor	https://m.media-amazon.com/images/I/81dRcIG8gB...	https://www.amazon.in/Mahogany-Life-Vintage-Ha...	4.2	210	₹469	₹901
3	Indianara Religious Painting - Synthetic Wood, ...	home & kitchen	Home Décor	https://m.media-amazon.com/images/I/91aG2syqs9...	https://www.amazon.in/Indianara-Panchmukhi-Syn...	4.5	589	₹182	₹499
4	Lexton 20 Piece Photo Clips String Light Batt...	home & kitchen	Home Décor	https://m.media-amazon.com/images/I/51emgTV2Mf...	https://www.amazon.in/Lexton-Included-Christma...	4.0	1,539	₹244.99	₹499

2. Pre-processing step:

We got 129 total missing values, we deleted them to avoid getting duplicate values, we had 1143 values and for the discount/actual price columns we did remove ₹ char to get only numerical values. We did collect all the necessary information on the *input_text* column for the tokenization step.

```
df['input_text'] = f"Product name: {df['name']}, Category: {df['main_category']}, Sub-category: {df['sub_category']}, Rating: {df['ratings'].fillna('0').astype(str)}, Price: {df['actual_price'].fillna('0').astype(str)}"
df['target_text'] = "This is a great product for " + df['main_category'] + " users."
df[['input_text', 'target_text']].head()
```

	input_text	target_text
0	Product name: 0 RAG28 Wooden Wall Decor ...	This is a great product for home & kitchen users.
1	Product name: 0 RAG28 Wooden Wall Decor ...	This is a great product for home & kitchen users.
2	Product name: 0 RAG28 Wooden Wall Decor ...	This is a great product for home & kitchen users.
3	Product name: 0 RAG28 Wooden Wall Decor ...	This is a great product for home & kitchen users.
4	Product name: 0 RAG28 Wooden Wall Decor ...	This is a great product for home & kitchen users.

3. Tokenization step:

We worked with GPT-2 tokenize model

```
# load the GPT-2 tokenizer
tokenizer = GPT2Tokenizer.from_pretrained('gpt2')
tokenizer.pad_token = tokenizer.eos_token
```

Where we first prepared our dataset for training a transformer model by converting text into numerical tokens that the model can process, we took a row of data and its index, we tokenized both input text and target label, we padded shorter text to ensure all the rows has the same length and then stored the tokenized target.

```
def tokenize_sample(row, idx):
    if (idx + 1) % 100 == 0:
        print(f"sample {idx + 1}")
    input_enc = tokenizer(
        row['input_text'],
        padding='max_length',
        truncation=True,
        max_length=512
    )

    target_enc = tokenizer(
        row['target_text'],
        padding='max_length',
        truncation=True,
        max_length=512
    )

    input_enc['labels'] = target_enc['input_ids']
    return input_enc

train_dataset = df[['input_text', 'target_text']].apply(lambda x: tokenize_sample(x, x.name), axis=1).tolist()
train_dataset = Dataset.from_list(train_dataset)
```

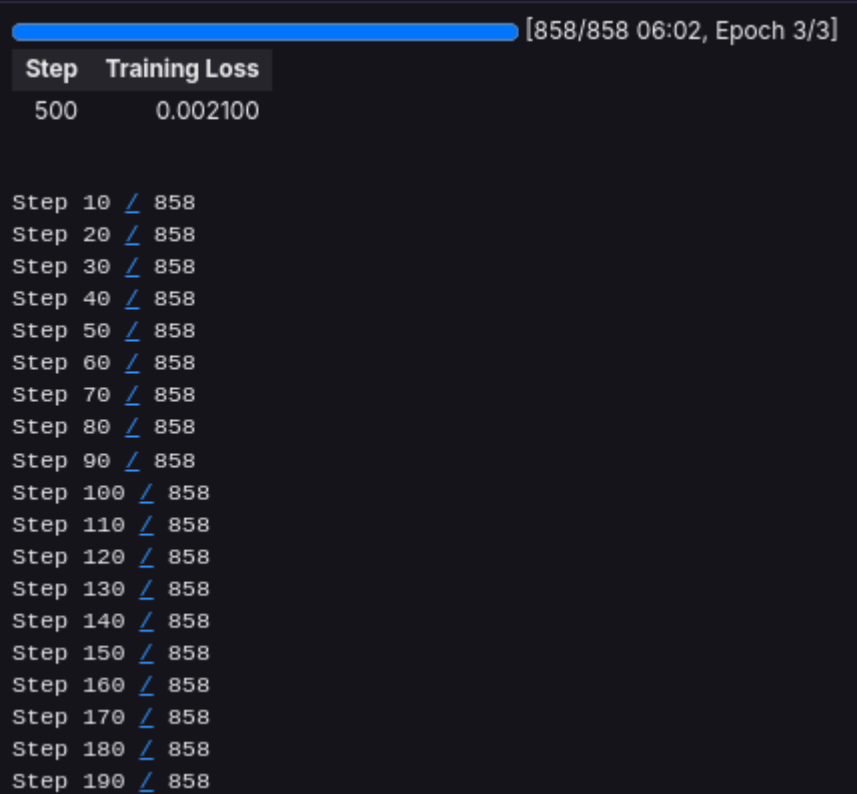
4. training model step:

We fine-tuned the pre-trained GPT-2 model to generate products description with the hyper-parameters:

- `learning_rate` = `2e-5`
- `batch_size` = `4`

- `epochs = 3`
- `save_steps = 100` (where we save the model every 100 steps).

We did custom `LoggingCallback` for debugging purpose where we print training progress each 10 steps, on the start/end of the epoch and print total steps completed.



```

[858/858 06:02, Epoch 3/3]
Step  Training Loss
500    0.002100

Step 10 / 858
Step 20 / 858
Step 30 / 858
Step 40 / 858
Step 50 / 858
Step 60 / 858
Step 70 / 858
Step 80 / 858
Step 90 / 858
Step 100 / 858
Step 110 / 858
Step 120 / 858
Step 130 / 858
Step 140 / 858
Step 150 / 858
Step 160 / 858
Step 170 / 858
Step 180 / 858
Step 190 / 858

```

We generated product descriptions automatically using our trained model (GPT-2), where it took structured product data (names, categories and prices) which were previously converted into numeric tokens using an attention mask to focus only on relevant parts of the input, the model created a human-like description (up to 50 words) by predicting the most suitable words one by one. As output, our numerical data is converted back into readable text and each generated description is added as a new column named `generated_description` alongside the original data.

5. Evaluation step:

We used *ROUGE* evaluation which compares AI-generated text to human-written references by checking:

- Word overlap (`rouge1`): Matching single words.
- Phrase overlap (`rouge2`): Matching pairs of consecutive words.
- Longest matching sequence (`rougeL`): Matching the longest identical stretch of words.

Each score has three parts:

- Precision: % of AI-generated words that appear in the reference.
- Recall: % of reference words captured by the AI.
- F-measure: Balanced average of both.

We got

```
{'rouge1': Score(precision=0.7272, recall=0.6818, fmeasure=0.7038),  
'rouge2': Score(precision=0.5, recall=0.4444, fmeasure=0.4705),  
'rougeL': Score(precision=0.6818, recall=0.6363, fmeasure=0.6581)}
```

- ROUGE-1 (F=0.70): ~70% of single words matched the reference, this high value means our model can capture key product terms well.
- ROUGE-2 (F=0.47): only ~47% of two-word phrases matched and recall < precision (0.44<0.5) means that AI omits some references details.
- ROUGE-L (F=0.65): ~65% of the longest matching word sequence aligned which suggests some coherent phrasing matches human writing.