

Product Hierarchical Classification

For the given problem statement, I first explored the dataset shared with me. Based on my understanding of the dataset, which has columns like Title and Text and three levels of categories, where the “Title” column represents the Product name and the “Text” column represents the reviews given by various users.

Objective

To build a hierarchical classifier that will help me predict these three levels of categories: “Cat1,” “Cat2,” and “Cat3” using context from given Text (Reviews) and Title(Product name).

I have divided my solutions into 3 parts (you can find the respective reference notebooks in the “notebooks” folder:

1. Data Visualization and preparation
2. Fine-Tuning/Modeling and performance evaluation on training-validation data
3. Hierarchical Prediction and evaluation of Test data

Notebooks Used

1. **Data_visualization_and_preparation.ipynb → Step1**
2. **product_Cat1_classifier.ipynb → Step2**
3. **product_Cat2_seq2seq_classifier.ipynb → Step3**
4. **product_Cat3_seq2seq_classifier.ipynb → Step4**
5. **Test_result_analysis.ipynb → Step5**

1. Data Visualization and Preparation

Reference Notebook: [data_visualization_and_preparation.ipynb](#)

In the first step, I performed data visualization on the given dataset to understand and analyze the different columns in datasets, types of text or information available in Text/Title columns, number of distinct values in all levels of categories.

- **Check the overall Count, distinct count, and fill rate of columns**

No. of distinct values in 'productId': 6865

No. of distinct values in 'Title': 6513

No. of distinct values in 'userId': 9716

No. of distinct values in 'Time': 210

No. of distinct values in 'Text': 9854

No. of distinct values in 'Cat1': 6

No. of distinct values in 'Cat2': 64

No. of distinct values in 'Cat3': 377

Column	Non-Null	Count	Dtype
productId	10000	non-null	object
Title	9995	non-null	object
userId	10000	non-null	object
Time	10000	non-null	int64
Text	10000	non-null	object
Cat1	10000	non-null	object
Cat2	10000	non-null	object
Cat3	10000	non-null	object

*** 5 records out of 10000 in the “Title” column are missing. For these missing “Title” I tried to fill and map them with available Title in their respective productId. But their respective productId’s [B000GUFFQS, B000PJNRC] don’t have any available Titles.*

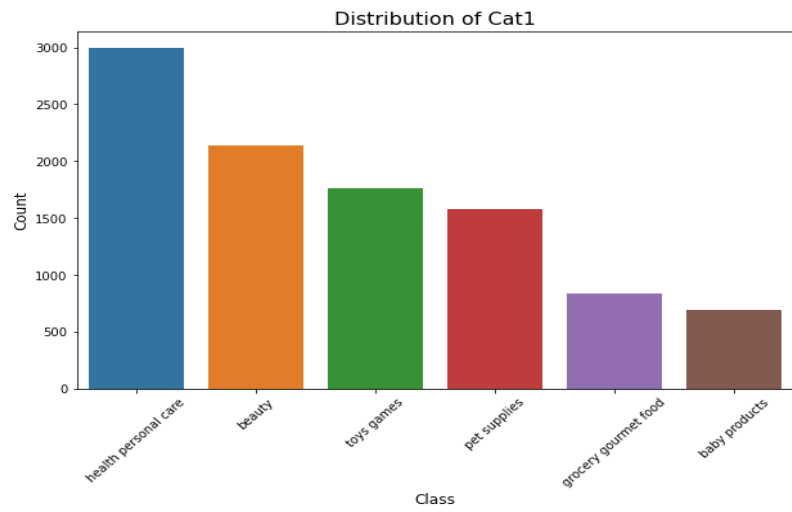
So I have removed these 5 records from 10000 records and did further processes on the remaining 9995 records.

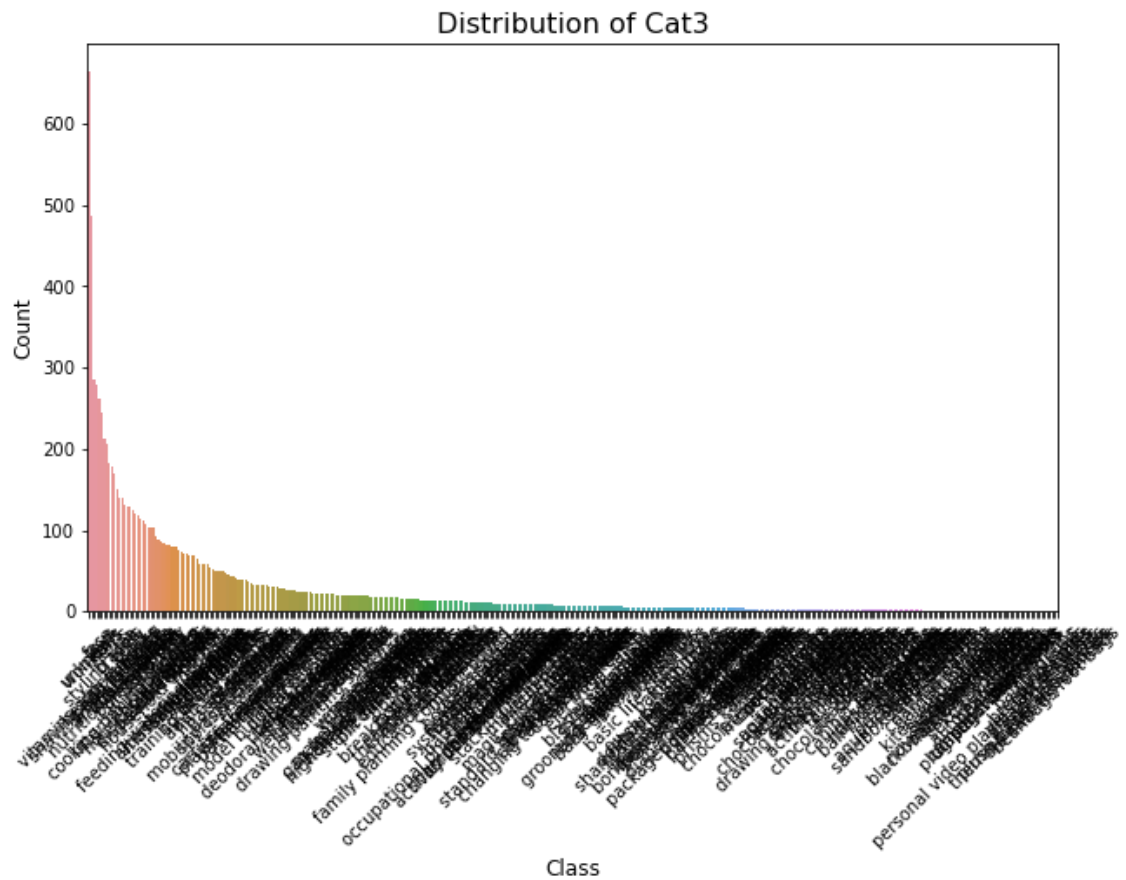
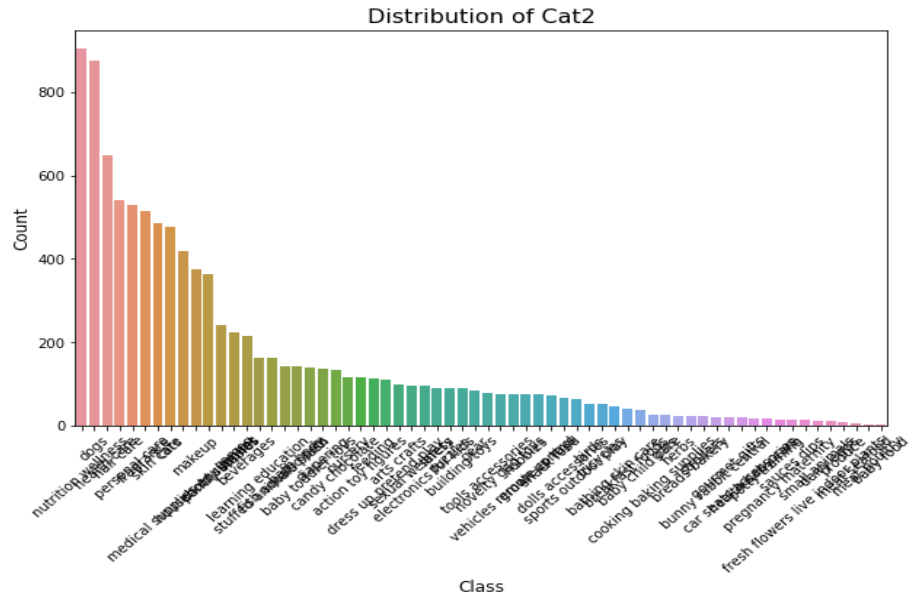
- **Removing Noise and Cleaning:** I did basic cleaning on the “Text” and “Title” columns like removing symbols, emails, URLs, numbers, etc, and renaming the respective columns as “clean_Text” and “clean_Title”.
- **Distribution of Cat1, Cat2, and Cat3 columns**

No. of distinct categories in 'Cat1': 6

No. of distinct categories in 'Cat2': 64

No. of distinct categories in 'Cat3': 377





** I have analyzed the distribution of Cat1, Cat2, and Cat3 using the plots and count, and also based on that we have further decided on our modeling strategy:

1. **At the first level “Cat1”:** There are 6 distinct classes with a decent count in each class --> Can use transformers-based encoder models like BERT for Sequence classification.
2. **At the second level “Cat2”:** There are around 64 distinct classes in “Cat2” and not all classes have decent counts. Few classes like "dairy eggs", "produce", "meat seafood", "meat poultry", and "baby food" have less than 10 in number at “Cat2 -->
 - Very complex and not feasible to develop any classification model on top of Cat2.
 - Not all classes would split properly into training and validation data
3. **At the third level “Cat3”:** There are around 377 classes in “Cat3”, and a few classes in “Cat3” have one record.

- **Splitting Data into train, validation, and test set**

Training set: 7496 (75% of data)

Validation set: 1499 (15% of data)

Testing set: 1000 (10% of data)

***Also created a **master_category_data**, which have the master list of all the classes in Cat1, Cat2 and Cat3*

- **Combined clean_Text and clean_Title:** After analyzing the words in the Title columns, I believe, it could be very useful while training to capture important information for predicting Hierarchical categories. So I have combined the **clean_Text** and **clean_Title** to create new “**combined_text**” column for all train, validation and test set like this:

>> df_train[combined_text]='the name of the product is ' + df_train['clean_Title'] + ' and the review of the product given by the user after purchasing it is ' + df_train['clean_Text']

>> the name of the the product is singer knitting machine and the review of the product given by the user after purchasing it is you should just save your time and money no matter how much time you use it skips knits and causes holes in your work

2. Hierarchical Model Building and Evaluation of Train-Validation Data.

After preparing the data and analyzing the distribution and count of classes available in Cat1, Cat2, and Cat3, I concluded that It would not be feasible to build a deterministic model to perform hierarchical classification and prediction of classes at each level Cat1, Cat2, Cat3.

Approach Used

I came up with a unique approach for this problem statement by using a **mixture of BERT (for Cat1) and Sequence2Sequence Flan T5 (for Cat2, Cat3) models** to perform hierarchical classification.

Level 1 Stage: Product Cat1 Classifier using BERT

As we have enough records for each class in Cat1, I have used the BERT model at level 1 to perform sequence Classification on available text data (“combined_text”) to predict 6 classes in Cat1.

Reference Notebooks: [product_Cat1_classifier.ipynb](#)

Model and Tokenizer used: BertForSequenceClassification, BertTokenizer

Input Sequence: combined_text

“the **name** of the product is **numberm scotchgard fabric protector** and the **review** of the product given by the user after purchasing it is i ordered three cans in order to treat cushions we were having recovered with seven cats in the house its important to be able to easily remove cat hair and an occasional hairball the instructions on the can strongly suggest doing a colorfast test on a hidden area of the material unfortunately all of our fabric failed the colorfast test so weve not been able to use the product although i expect its fine with compatible material just be aware that it cant be used on everything ive heard its great on canvas shoes so ill be giving that a try”


Output: probability score of 6 classes: 'pet supplies' 'health personal care' 'grocery gourmet food' 'toys games', 'beauty', and 'baby products'.

-Selected class with max prob_score as predicted_category at level1 (pred_Cat1) for given text.

Training and Model evaluation on validation set:

We have fine-tuned the BERT model on the custom training dataset and evaluated the model performance on the validation set.

Achieved a very good performance of 92% on the validation set



Epoch	Training Loss	Validation Loss	Accuracy	Precision	Recall	F1
1	No log	0.329897	0.901268	0.901461	0.901268	0.901256
2	0.452700	0.356400	0.899933	0.903293	0.899933	0.900029
3	0.186300	0.355803	0.917278	0.918109	0.917278	0.917252

**Please go through the reference notebook for more details on model configuration, tokenization, modeling, and evaluation steps.

Prediction on unseen Test data:

Once our BERT model is fine-tuned and trained on our custom data, I used the trained model to perform prediction on unseen test data.

And save the test data having predicted category 1 with column “pred_Cat1”.

—> *Next, We will use the predicted category “pred_Cat1” of test data to predict their level 2 categories in the next step.*

Level 2 stage : Product Cat2 Classifier using Sequence2Sequence Flan-T5 model

Reference Notebooks: [product_Cat2_seq2seq_classifier.ipynb](#)

At Level 2, As we don't have enough records in classes to build a classification model, I have come up with an approach to build the **Seq2Seq** model to perform **Question-answer** tasks. So that model can learn while training and helps to answer the correct category from the given context.

To train and fine-tune the model on this task, I have constructed a question-answer prompt with options formatting where it uses `combined_text` as the context and `Cat1` as a hint and gives options to predict from the unique `Cat2` classes associated with the `Cat1` of a `productId`.

Example prompt:

Context: the **name** of the product is **singer knitting machine** and the **review** of the product given by the user after purchasing it is you should just save your time and money no matter how much time you use it skips knits and causes holes in your work

(—> Using `combined_text` as context)

Hint: This is a **toys games** product. —> (using `Cat1` category : **toys games** as Hint)

Question: What is the most appropriate sub-category for this product?

(—> Framed the question to predict `Cat2`)

Options: (---> suggested options from `Cat2` classes associated with `Cat1` of a `productId`)

- A) hobbies
- B) tricycles
- C) action toy figures
- D) electronics for kids
- E) arts crafts
- F) vehicles remote control
- G) dolls accessories
- H) novelty gag toys
- I) dress up pretend play
- J) games
- K) baby toddler toys
- L) building toys
- M) learning education
- N) grown up toys
- O) stuffed animals plush

Notes:

- We have used actual Cat1 to give Hint and suggest options w.r.t their Cat1 while fine-tuning the model using train data and evaluating the performance of validation data.
- But we have used predicted Category 1 (pred_Cat1) to give Hint and suggest options w.r.t their pred_Cat1 while constructing the prompt for Testing data to predict Hierarchically.

```
# Generate prompts for each row (for training/validation/testing)
train_df['question'] = train_df.apply(lambda row: create_Cat1_prompt(row, "Cat1",
master_Cat_df), axis=1)
val_df['question'] = val_df.apply(lambda row: create_Cat1_prompt(row, "Cat1",
master_Cat_df), axis=1)

#Used predicted category 1 generated using BertClassifier at Level1 to perform
hierarchical classification
test_df['question'] = test_df.apply(lambda row: create_Cat1_prompt(row, "pred_Cat1",
master_Cat_df), axis=
```

Model name: "google/flan-t5-base"

Model and Tokenizer used: T5ForConditionalGeneration, T5Tokenizer


Input Sequence: column "question"

Output Sequence: predict the most appropriate level 2 category from the given options.

Compute metrics: used rouge score as computing metrics while fine-tuning

Model Performance Evaluation on Validation Set:

After fine-tuning the Flan-T5 model for this specific Q/A use case, we have achieved the following performance on the validation set at Level 2.

✓  [2811/2811 1:10:49, Epoch 3/3]

Epoch	Training Loss	Validation Loss	Rouge1	Rouge2	RougeL	RougeLsum
1	0.442000	0.148002	0.797698	0.486324	0.797343	0.797621
2	0.153500	0.137243	0.816311	0.500334	0.815132	0.816055
3	0.123900	0.140406	0.825801	0.507005	0.824486	0.826024

Precision: 0.8110921383169505

Recall: 0.818545697131421

F1-Score: 0.8099489270078442

Accuracy: 0.818545697131421

Prediction on unseen Test data:

After fine-tuning and evaluating the model performance, I made predictions on unseen test data.

And saved the test data having predicted category 2 with column “pred_Cat2”.

—> *Next, We will use the both predicted categories “pred_Cat1” and “pred_Cat2” of test data to predict their level 3 categories in the next step.*

Level 3 Stage: Product Cat3 Classifier using Sequence2Sequence Flan-T5 model

Reference Notebooks: product_Cat3_seq2seq_classifier.ipynb

Similarly to the Level 2 strategy, In this stage as well I have constructed a prompt in a similar way but here I am using both the Level 1 category (Cat1) and the Level 2 category (Cat2) in the prompt.

**** Notes**

- Use Cat1, and Cat2 while fine-tuning training data and evaluating model performance
- Use pred_Cat1, and pred_Cat2 while testing on unseen test data.

Example Prompt:

Context: the name of the the product is singer knitting machine and the review of the product given by the user after purchasing it is you should just save your time and money no matter how much time you use it skips knits and causes holes in your work

Hint: This is a toys games product in the arts crafts sub-category.

—> (using Cat1 category: toys games and Cat2 category: arts crafts as Hint)


Question: What is the most appropriate specific category for this product?

Options:

- A) drawing sketching tablets
- B) stickers
- C) drawing painting supplies
- D) easels
- E) unknown
- F) craft kits
- G) blackboards whiteboards
- H) aprons smocks
- I) clay dough

```
train_df['question'] = train_df.apply(lambda row: create_prompt_for_Cat3(row, "Cat1",  
"Cat2" ,master_Cat_df), axis=1)  
val_df['question'] = val_df.apply(lambda row: create_prompt_for_Cat3(row, "Cat1",  
"Cat2" ,master_Cat_df), axis=1)  
test_df['question'] = test_df.apply(lambda row: create_prompt_for_Cat3(row,  
"pred_Cat1","pred_Cat2", master_Cat_df), axis=1)
```

Model Evaluation on Validation set:



Epoch	Training Loss	Validation Loss	Rouge1	Rouge2	RougeL	RougeLsum
1	No log	0.121744	0.782284	0.508117	0.781873	0.781743
2	0.818000	0.100112	0.813491	0.534134	0.813655	0.812765
3	0.116600	0.100169	0.820572	0.539137	0.820066	0.819432

Precision: 0.7945041816601798
Recall: 0.7918612408272181
f1-Score: 0.7822668503286893
Accuracy: 0.7918612408272181

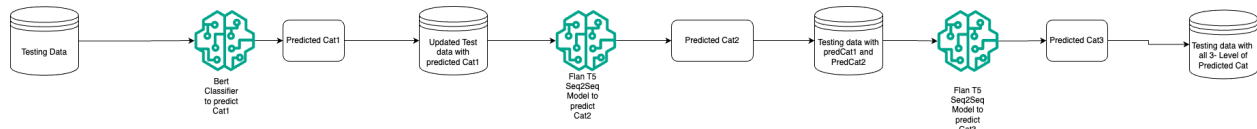
Prediction on unseen Test data:

After fine-tuning and evaluating the model performance, I made predictions on unseen test data.

And saved the test data having predicted category 3 with column "pred_Cat3".

→ Now we will evaluate the performance of the model on combined Hierarchical prediction of test data at Level 1, 2, and 3

3. Hierarchical Prediction and evaluation of Test data



Test Result Analysis

Metrics	Cat1	Cat2	Cat3	Average
Precision	0.9283	0.7627	0.6215	0.7708
Recall	0.928	0.765	0.646	0.7796
F1 Score	0.928	0.7561	0.619	0.7677
Accuracy	0.928	0.765	0.646	0.7796

- Cat1: The model performs excellently with high precision, recall, F1 score, and accuracy, all above **92%**. This indicates that the model consistently predicts the correct labels in this category with very few mistakes.
- Cat2: The performance is slightly lower compared to Cat1, with precision, recall, F1 score, and accuracy all in the **75-77%** range.
- Cat3: This category shows the lowest performance. Precision, recall, F1 score, and accuracy are all significantly lower (around **62-65%**), indicating that the model is struggling more with this category, either due to more variability in the data, or more class imbalance.
- Average Result: The average precision, recall, and F1 score are **0.7813**, suggesting that overall, the model performs well across all categories, with around **77%** of predictions being correct. The accuracy of **0.7796** indicates that about **78%** of all predictions across all categories were correct, which is a reasonable performance

No of correct prediction out of 1000 (Cat1 only): **928 (92.8%)**

No of correct prediction out of 1000 (Cat1, Cat2 combined): **765 (76.5%)**

No of correct prediction out of 1000 (Cat1, Cat2, Cat3 combined): **635 (63.5%)**

No of correct prediction out of 1000 (Cat1, Cat3 combined): **646 (64.6%)**

No of correct prediction out of 1000 (Cat2, Cat3 combined): **635 (63.5%)**

I have also shared the prediction on testing at all three levels of the category for your reference.[Named: [testing_data_pred_Cat_1_2_3.csv](#)]