Gaming intent report

Opera Recrutation

Categorization of User Inputs Based on Gaming and Non-Gaming Intent

Author:
M. eng Leonard Feszczuk

January 30, 2025

Contents

1 Introduction **2**
1.1 Objective: . . . 2
1.2 Model Selection and Setup: . . . 2
1.3 Data Collection and Preprocessing: . . . 2
1.4 Fine-tuning the Model: . . . 2

2 Proof of concept **2**
2.1 Scores . . . 3
2.2 Scalability . . . 4

1 Introduction

1.1 Objective:

The primary goal of the project is to set up and utilize machine learning models for text classification. The aim is to detect gaming-related intent in user inputs, which can help identify users' interests in gaming.

1.2 Model Selection and Setup:

We will begin by selecting a suitable pre-trained text classification model from established libraries such as Hugging Face. Models such as BERT, GPT, or DistilBERT will be considered for their capabilities in understanding natural language. The chosen model will be fine-tuned with a custom dataset specifically labeled for gaming-related intents and non-gaming-related intents. This step will involve adapting the model to better understand the context of gaming interactions.

1.3 Data Collection and Preprocessing:

A robust dataset will be collected, consisting of examples of gaming-related content (e.g., game titles, gaming behavior, gaming terminology) and non-gaming-related content (e.g., general conversation, inquiries unrelated to gaming). The data will undergo preprocessing, including text cleaning, tokenization, and appropriate labeling to ensure the model is trained effectively for the task.

1.4 Fine-tuning the Model:

The pre-trained model will be fine-tuned using the prepared dataset, adjusting the model's parameters to improve its ability to classify inputs accurately as gaming-related or non-gaming-related. Performance metrics such as accuracy, precision, recall, and F1 score will be used to evaluate the model's effectiveness and ensure that it meets the required standards for real-time usage.

2 Proof of concept

Code used here can be downloaded from Click here to go to Github.
For the purpose of the task a proof of concept was made containing a pre trained model BERt which is widely accessible, performs well for classification tasks, and has excellent support through libraries like Hugging Face's transformers. The data for training was a kaggle dataset containing of 27000 gaming reddit posts. Given data wasn't labeled for gaming or non gaming intent so the dataset had to be labeld using one of the methods:

- Manual Labeling (Human Annotation):A manual approach where a individual is reviewing and annotating each text sample.
- Semi-Automatic Labeling This method combines human oversight with automatic text classification tools. A pre-trained model can be used to predict labels, and humans review or correct those predictions.

- Rule-Based Labeling (Heuristic Methods) For specific intents related to gaming, rule-based methods can be employed. This involves creating a set of rules or patterns (e.g., keywords or regular expressions) to label data automatically.

To efficiently label the training data, I chose a semi-automatic labeling approach because it was both fast and easy while maintaining a reasonable level of accuracy. A pre-trained model was

```python
file_path = "reddit_gaming_sample.csv"
df = pd.read_csv(file_path)

# Initialize zero-shot classification pipeline
classifier = pipeline("zero-shot-classification", model="facebook/bart-large-mnli")
candidate_labels = ["gaming-related", "not gaming-related"]

def classify_comment(comment):
    result = classifier(comment, candidate_labels)
    return 1 if result["labels"][0] == "gaming-related" else 0


df["gaming_related"] = df["Comment"].apply(classify_comment)
df.to_csv("reddit_gaming_sample_labeled.csv", index=False)
```

Figure 1: Zero shot classification using pre trained model

used to generate initial predictions on a raw dataset. The generated labels were then reviewed and corrected to ensure accuracy. This method significantly reduced the time required for labeling compared to fully manual annotation while maintaining reliable data quality.

2.1 Scores

Altough the initial scores for the model are rather low they'are promissing in terms of fine tuning the model. Some of the things we can do to improve the model are:

	precision	recall	f1-score	support
0	0.83	0.86	0.84	107
1	0.83	0.80	0.81	93
accuracy			0.83	200
macro avg	0.83	0.83	0.83	200
weighted avg	0.83	0.83	0.83	200

Figure 2: Initial scores of BERT model

1. Expanded and Improved Dataset: Increased labeled examples, balanced gaming vs. non-gaming data, and refined label quality using active learning.
2. Hyperparameter Tuning: Adjusted batch size , learning rate , number of epochs

3. Data Augmentation: Applied synonym replacement, back-translation, and paraphrasing to improve model generalization and handle diverse user inputs.

4. Retraining and Evaluation: Continuously fine-tuned the model with improved data, adjusted hyperparameters, and monitored performance to prevent degradation over time.

2.2 Scalability

When dealing with large datasets like 1 million text inputs we need to think about scalability of our program some of the problems include:

- **High Computational Cost**: Fine-tuning and inference require significant resources. .
- **Slow Inference Speed**: Transformer models are inefficient for real-time tasks.
- **Memory and Storage Constraints**: Large models consume excessive memory.
- **Handling Evolving Data**: User patterns shift over time, requiring frequent retraining.
- **Scalability in Distributed Systems**: Load balancing and synchronization issues arise.

To battle this problem we can deploy optimized inference frameworks and batch processing, apply model pruning, knowledge distillation, and caching techniques.