MAPGPT FINE-TUNE INTENT CLASSIFIER

Problem Statement: build an intent classifier on a small, multilingual dataset.

Using the data parquet file, I did some initial investigation and found that there were six possible labels and six languages:

Label	Count
search_music	50
control_navigation	50
no_action	50
search_pois	40
search_poi	30
drivers_manual	20

Language	Count
English	174
Japanese	30
Korean	25
Spanish	5
French	5
German	1

This meant that a multilingual text embedding model would be needed.

Model

I used DistilBERT multilingual model ‘distilbert-base-multilingual-cased’ because it is free, relatively small, and supports many languages. The pretrained weights do a good job of providing rich contextual embeddings that capture semantic relationships in the input text. These embeddings generalize well, meaning the model can leverage the knowledge learned during pretraining to understand new and varied inputs quickly. This allows the model to adapt effectively to the intent classification task even with limited fine-tuning data, making training faster and improving performance on unseen examples.

Experiments

Given the fact that the fine-tuning data was limited, most of the experiments I explored were around preprocessing and augmenting the data.

Since the data contained chat histories, I chose to combine the entire chat history with the last utterance, separated by special tokens. The previous conversation context could help the model understand the intent better, and even if it wasn’t useful for that purpose, it would at least add some augmentation to the data. I trained a model using just the chat history combined with the last utterance, but the results were mediocre. The validation loss only got down to about 0.82 and F1 to around .7. It also started overfitting quite quickly after a few epochs.

[480/480 10:53, Epoch 20/20]

Epoch	Training Loss	Validation Loss	Accuracy	F1 Macro
5	0.701300	0.827012	0.687500	0.628919
6	0.701300	0.876423	0.687500	0.683406
7	0.443000	0.898963	0.729167	0.707281
8	0.443000	1.112372	0.729167	0.717724
9	0.197600	1.401664	0.666667	0.644710
10	0.197600	1.189561	0.708333	0.685198
11	0.043100	1.380966	0.687500	0.677779
12	0.043100	1.455020	0.666667	0.656615
13	0.011000	1.559875	0.687500	0.649140
14	0.011000	1.549610	0.729167	0.707128
15	0.006800	1.509418	0.708333	0.692642
16	0.006800	1.547874	0.729167	0.707128
17	0.005300	1.583691	0.729167	0.707128
18	0.005300	1.590756	0.729167	0.707128
19	0.004500	1.588816	0.729167	0.707128
20	0.004500	1.587603	0.729167	0.707128

For preprocessing, I removed numbers because they are not meaningful for understanding the intent. I also experimented with removing named entities like people’s names or locations. To do this I downloaded spaCy language-specific models in each

language and used named entity recognition models to replace the named entities with placeholders. The idea was to make the model learn the intent rather than overfit to specific names. Again, I trained a model where all data only had placeholders and no original names and places, but the results were somewhat mixed, the replacement likely tokens took up too much of the text.

[385/960 09:30 < 14:16, 0.67 it/s, Epoch 8/20]

Epoch	Training Loss	Validation Loss	Accuracy	F1 Macro
1	1.168600	1.038334	0.593750	0.534487
2	0.916800	0.840360	0.635417	0.584067
3	0.635800	0.753266	0.750000	0.747151
4	0.189900	0.829356	0.750000	0.724762
5	0.112800	0.694277	0.822917	0.818246
6	0.199400	0.723663	0.833333	0.828856
7	0.009600	0.713601	0.864583	0.858099
8	0.132500	0.805854	0.843750	0.839388

In the end, in order to give the model diverse input, I created four versions of the data: raw combined chat history with last utterance, raw last utterance only, cleaned combined context, and cleaned last utterance only. By combining these I increased the amount of training data and also gave several augmented variations of the same intent, to allow the model to generalize better.

[960/960 12:09, Epoch 10/10]

Epoch	Training Loss	Validation Loss	Accuracy	F1 Weighted
1	0.758200	0.640132	0.786458	0.778551
2	0.221200	0.309669	0.916667	0.914915
3	0.247400	0.229543	0.942708	0.942922
4	0.061100	0.243273	0.947917	0.948607
5	0.018100	0.194383	0.968750	0.968756
6	0.001900	0.211686	0.953125	0.953065
7	0.001300	0.127993	0.963542	0.963540
8	0.001200	0.122727	0.963542	0.963642
9	0.001400	0.113706	0.968750	0.968764
10	0.030900	0.111119	0.973958	0.973914

In the end, I based my evaluation on the weighted F1 score because it takes into consideration the class imbalance in the data, it also takes into consideration both precision and recall. I still stopped training early as it starts to overfit quite quickly. I think the final result showed the model learned to classify well on unseen validation data and still perform with a high precision & recall. To be more confident of the results, more investigation could be done to make sure the validation dataset was sufficiently different from the training set as well as adding more in-the-wild, hard to classify examples to the evaluation data, to make sure it truly generalizes well.

Future Improvements

To improve results further I could have spent more time fine-tuning the learning rate to find the global minimum. I also could have spent time oversampling underrepresented labels and languages or hard to learn examples. In addition, depending on computation requirements, different models could have been considered. If there were latency or size limits, a small, more computationally efficient model such as fastText or Logistic Regression with TF-IDF features may also have given decent baseline results, though creating a good result for multiple languages may require training a separate model for each language. These models are good at quickly classifying text with low resource usage and can perform surprisingly well on intent classification tasks with properly engineered features. On the other hand, if time and resources were not an issue, a more accurate model could be trained using large-scale transformer-based models like RoBERTa, GPT, or T5, which have more parameters and extensive pretraining.

Overall, I saw that adding many different augmentations of the same data greatly helped my results, given the small dataset. In the future, I would have added more augmentations, such as token dropout or replacing words with their synonyms. Other options could be using generative models to create more training text, doing back-translation to translate the sentences into another language and then back again to the original to get more variations. Also, simply translating every entry into every other language, would increase the data set as well.