Initial task during internship with a lot of data:

Given a user query (e.g. “funny face filters”) predict the category of the Huawei App Gallery (Disclaimer: the task was a bit ridiculous… but it is what it is)

Available labeled data: App named and their descriptions from the Huawei App Gallery

Testing data: Annotated user queries myself (which can be HIGHLY ambiguous)

Note: no sub-categories for Games. So 1 classifier to distinguish APP from GAME and second one to classify subcategory of APP but not for GAME

Modified Task:

Given a user query (e.g. “funny face filters”) predict the category of the Apple Store (openly available dataset - much less data)

Available labeled data: App names and their descriptions from the Apple store

Testing data: Annotated user queries myself

Note: same as above, no subcategories for Games given

Obviously it would have been useful to have the data of what people search for and then download, but I did not get this one until the last month of my internship at Huawei, after which I quit.

Solution: Given that the training and testing data is different, I used a linear generative model instead of using deep-learning approaches. However, I compared my approach to using the [CLS] token in a classification task using BERT

Data:
There are two category 1 types - Games and Apps. Games were not further sub classified, but apps were. So I trained two classifiers, one to distinguish Apps from Games and a second one to classify amongst the App Categories (Navigation, Entertainment etc.).

Results
For Category 1 (App vs Game) linear models slightly outperformed BERT. Time complexity was also significantly better (a couple of seconds on CPU vs 20 minutes on GPU).

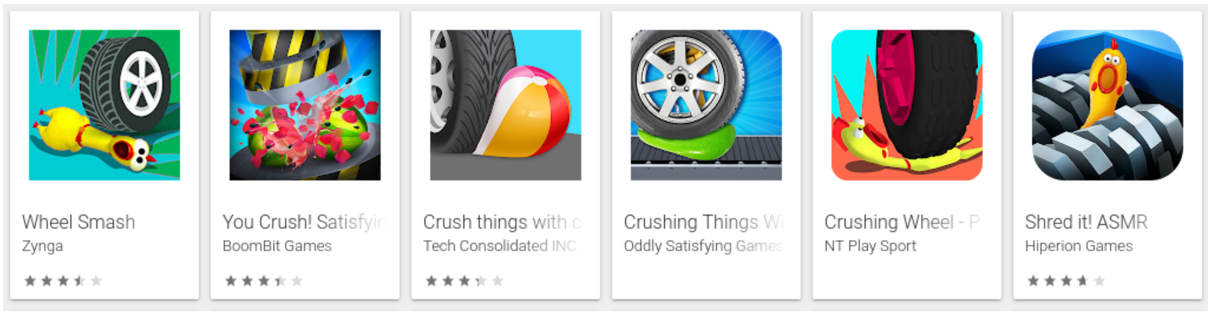
For Category 2 (categories within the APP category 1) Linear models significantly outperformed BERT: only 32% accuracy and an F1 score of 0.28 compared to Naive Bayes of 68% accuracy and 0.65 F1 score. This is not surprising given the difference of training and testing data and the ambiguity in the category 2 data

Additional notes:

During my internship I (obviously) conducted many more experiments and tried many more methods to push accuracy and precision up. One of them being trying to construct a corpus with short phrases using chunking and high coefficients. This reduced accuracy and recall, but increased precision.

Example where current approach fails:

User query: “running over things”



Obviously a language model and actual user data is needed in order to be able to tel that “running over things” is the same as “smashing” or “crushing” things in the app universe