Problem 1

Let's say you are a Data Scientist working in a company that analyzes social media contents. Business team approached you and told you to build an Agent that will understand the context of a social media post that they will use to segment the content,find popularity,trends etc.

Task:
● Write down your approach to make the dataset, preprocess and train ML model to build such an Intelligent Agent.
● Please note that a social media post could contain Texts, Images,Videos etc. And you have to take all of this kind of data to a single Agent.
● You don't have to write any code. Just give us a detailed step by step description about the process.

Response

Designing any Data Science project starts with getting a clear idea of business priorities. Engineering design decisions are swayed based upon the short, medium and long term business goals. Conversely, business goals might also change when faced with engineering constraints and realities (you can't safely design a very efficient airplane within a short few years and go to market).

We'll first list down some design parameters and some assumptions for our business goals.

Some Design Parameters:

- *Model Type*
- *Transfer Learning or Train from Scratch*
- *Use Ensembles?*
- *Low Latency Inference Required?*
- *Batch Inference or Real Time Inference?*
- *Batch Learning or Online Learning? Training Frequency?*

Assumptions:
Do we want a robust solution or do we want to go to market ASAP with the first iteration?
Ans: Come up with the first implementation ASAP to aid the Business team in their decision making process.

How many languages We'll support:
Ans: As many major languages as possible.

How do we define context?
Ans: We define context as a **list of tags** tied to each post. Each post is defined as a tuple of all available channels (Video,Image,Text,..) . Essentially narrowing down our model design to a ***multi-input, multi-label classification problem.***

What will be our Success Metric?
Ans: Since topics are likely to follow a power distribution, we can't settle on an average metric. Instead we'll consider the **weighted F0.5 score** as our metric. F0.5 penalizes false positives more and the weightage on class representation will give **niche topics/tags** enough importance.

Quality and Quantity of labeled data.
Ans: Collaborating with the business team, we've gathered a reliable dataset of a few thousand correctly labeled posts across a diverse set of topics.

Infrastructure Budget
Ans: For iteration 1 we have access to a cluster of medium level ec2 gpu instances with a few hundred Gigabytes of RAM.

Making the Dataset

We are given access to a REST API from multiple social media platforms. We write scripts that pull data from each site over the last 24 months. This time window will cover seasonality and yearly one-off events(Religious/Cultural Festivals, Black Friday etc.). We'll store the *text data in a No-SQL database, the images and video in blob storage like s3 and any accompanying metadata(user information, time of post, device type) in a RDBMS like PostgreSQL. The associated labels will also be stored in the RDBMS.*

We'll maintain an anonymized identification key across all storages.

Preprocessing Data for Training

Since we have multiple types of data. We'll need at least 4 separate pre-processing pipelines.
For Images:
1. Convert to uniform format and resolution
2. Convert to uniform dimensions(2d crop and number of channels)
3. Normalize using imagenet parameters.(For transfer learning)

For videos:
1. Convert to uniform format,resolution and dimensions
2. Sample fixed number of frames from random positions of video timeline(Most frames are repetitive and not likely to contain rich information)

For Texts:
1. Clip to a fixed number of words with padding(for feeding into a transformer based model).

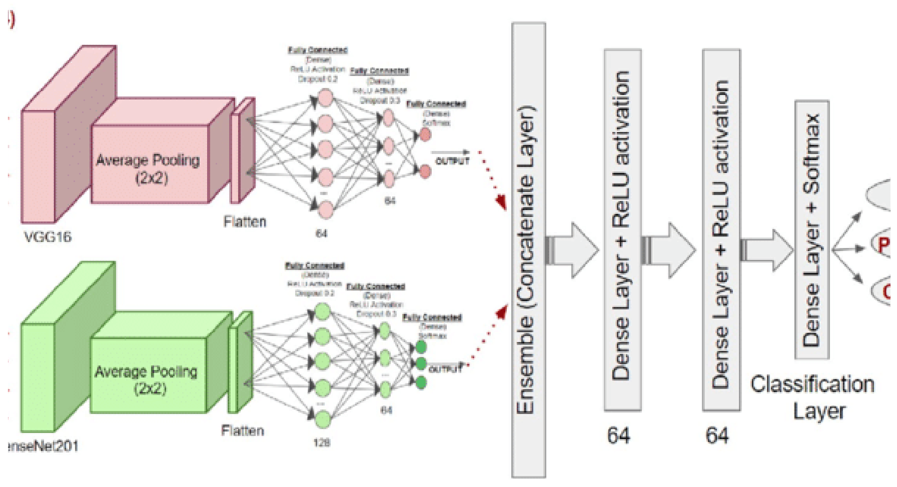
For Metadata:
1. Fill-in missing information with imputation from the full dataset.
2. Convert text labels to categorical values and perform one-hot encoding.

3. Scale numerical variables with min-max or normal scaler.

Model Design

Since We're dealing with multiple types of unstructured data, we'll leverage transfer learning with State of the art pre-trained models. We'll use separate pre-trained models for each type of data and concatenate their final layer tensor representations. The final stage will be multiple dense layers with a final softmax softmax layer with K out-put nodes. Where K is the number of distinct tags.

The idea can be visualized with an example from the medical image analysis domain(source)-



Average Pooling (2x2)
VGG16
Flatten
Fully Connected (Dense) ReLU Activation Dropout 0.2
Fully Connected (Dense) ReLU Activation Dropout 0.3
Fully Connected (Dense) Softmax
OUTPUT
64
64
Average Pooling (2x2)
enseNet201
Flatten
128
64
OUTPUT
Ensemble (Concatenate Layer)
Dense Layer + ReLU activation
64
Dense Layer + ReLU activation
64
Dense Layer + Softmax
Classification Layer

But instead of having two pre-trained image models, we'll have-

1. one pre-trained model for text(Bert Multilingual for example)
2. one pre-trained model for image (Any performant model)
3. one pre-trained model for video frames
4. one independent Deep Neural Network for Structured meta-data that we'll train from scratch.

We'll concatenate the final latent layer of all these models and feed the combined feature vector to a final classification stage.This separation of subtasks will allow us to debug our combined "Agent" better.

Pipeline Orchestration

1. **Training**
    a. We'll have a moving window of 24 months to sample training data from.
    b. To compensate for data-drift, we'll run scheduled training and update the model on a monthly basis or bi-weekly if data drift is later observed.

c. We'll use Apache Airflow to create two separate DAGs for training and preprocessing. The training DAG will be triggered upon completion of preprocessing DAG.

d. We'll use MLFlow to keep track of our model's training and performance metrics on a MLFlow web dash-board.
e. We'll use model and data versioning for experiment reproducibility. Updated model will replace the previous model in a designated folder.

2. **Inference:**
    a. We'll also have a separate inference pipeline. Here, preprocessing and inference can be combined into a single airflow DAG.
    b. This DAG will be scheduled to run every 6/12/24 hours according to business needs.
    c. Since the volume of data is huge and we don't need to be comprehensive to get an idea about the trends, we'll randomly sample a few hundred thousand posts every inference cycle.
    d. For each post in the current sample, we'll output a list of tags that the model predicts(above cut-off probability) it to belong to and store into postgreSQL. We can have an identifier column and a column for each tag with a binary indicator(0/1). We can also choose to store the probability with a float between 0.0 to 1.0.

3. **Delivery:**
    a. The tag distribution and other metrics like increase in popularity of each tag, tag clusters etc. can be visualized on a dashboard for business team's consumption.
    b. We can write a separate layer that computes these metrics from model outputs stored in the postgreSQL database.
    c. Finally, we can expose a REST API endpoint with FastAPI for consumption by the web dashboard.