Design a Machine Learning system to check if @someone at Twitter should be allowed or not.

1. **Text - nlp**
2. **Action: tweet block people from seeing this tweet (abuse and spam content); Labels tag(abuse) scratch; report do not have old solution**
3. **Social network**
4. **Scope binary data abuse not abuse**
5. **Quick solution first iteration complex limited resources: 1-2 annotators**

Objective: machinel learning system to detect abuse tweet to prevent people from seeing it. Customer retention

- **Data collection**
  Reports: 10k reports regarding abuse @ negative class
  1. **Reports + annotators (true negative class 10k + 50% 10k postive class) => 20k => traditional ml such logisitc regression/ svm/ tree-based model (simple scalable interpreting; probabilities) => 6 months => 600 k**
  2. **Reports + crowdsourcing + annotators => true positive class true negative class**
- **Feature Selection:**
  User profile based features: has this account been reported in the past? Number of followers; number of followings; number of favorites; account age at Twitter; previous retweet to tweet ratio; number of comments …..
  Tweet content based features: (text) the number of words, number of digits; number of special characters/illegal characters; the content of Tweet(normalization: lowercase; remove special characters; digit => word => word embedding)
  Cross tweet- user features: tweet the user has retweeted
  Feature Selection on non-text features tree models to generate feature importance; stepwise to select important features
  1000 most common word in tweet “This is a good movie” [9,34,55,199,40] [0,0,0,1,0] 500 oov buckets
- **Algorithm: logistic regression**
  Cross entropy loss function
  Macro F1 Score top 20 models
  Precision of the negative class to select the best model
  Training and testing 70% 30%
  category

- **600k data points labeled**
  CNN/ RNN/ TRANSFORMER
  CNN: kernel kernel size of 2 140 280
  RNN: cannot be run in paralell sequence; long dependency: LSTM/ GRU
  Transformer: no sequential; postional embedding + attention long dependency
  Load a pretraining BERT + cls pad tokens => tokens from the bert model => concatenation layer to add non-text features => Relu layers => sigmoid function output layer

Model Evaluation:

- **Offline Metrics: Macro F1 score; Precision of negative class**
- **Online Metrics: Customer retention rate; report rate; average session time …**

Model Deployment:

Model: optimization (quantization); parllel on GPU; Multi-stage models
Batching/Online learning: find "global" solution/ better solution;
Streaming update our model complex
Test: data test to detect data drift; A/B online metrics

===
Feedback notes:

1. 总分总答法;先说最好的**model if we have time/interested, dive deep**
2. **Session**中间问下**direction dive deep into** 哪个部分？
3. **Data collection + algorithm + feature selection**

1. **High level diagram**
2. **Online process(e.g. Feature store) offline process**
3. **Loop process**
4. **Single Failure Point in Machine Learning feature leaky**
5. **Backup Plan**
6. **Privacy**
7. **Centric/ devices**