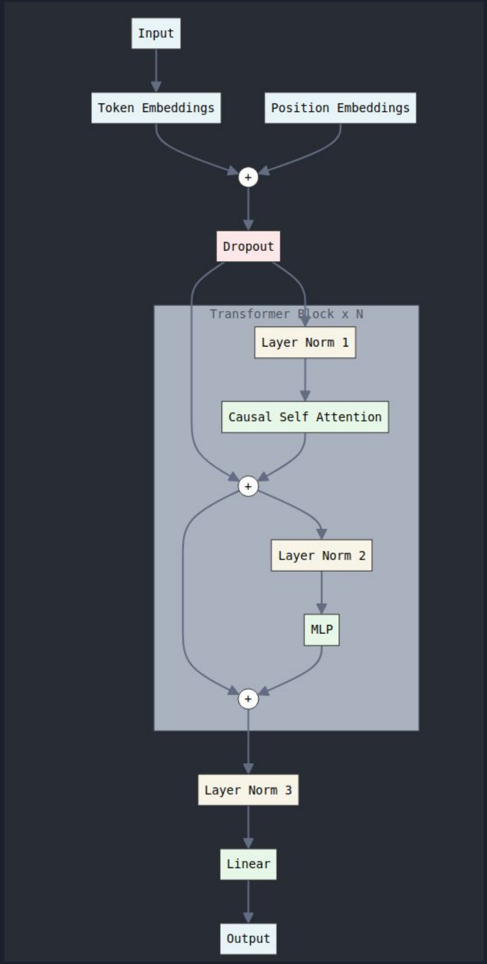
GPT for recruitment

Marwan Akrouch

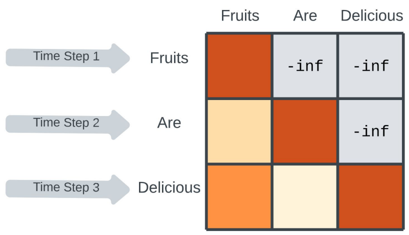
GPT-2 architecture

Parameters	Layers	d_{model}
117M	12	768
345M	24	1024
762M	36	1280
1542M	48	1600

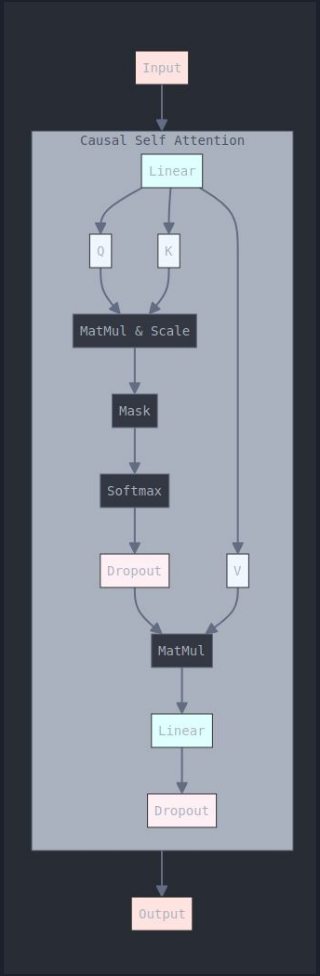


Attention bloc

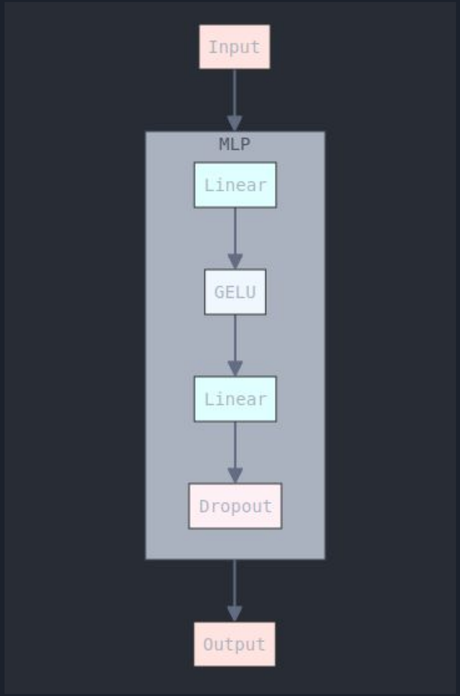
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



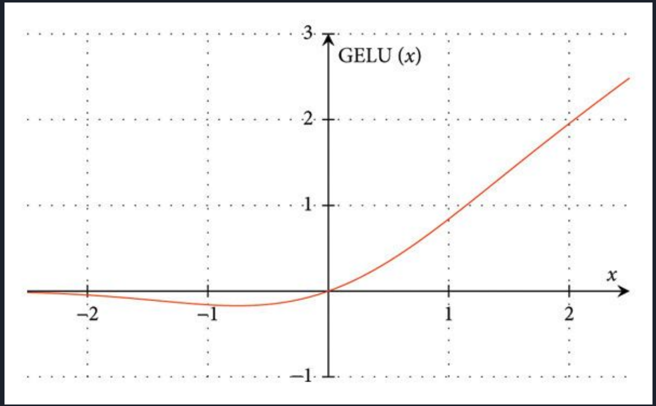
Masked Attention Matrix



MLP bloc



Input
MLP
Linear
GELU
Linear
Dropout
Output



GELU (x)
x
3
2
1
−1
−2
−1
1
2

Tokenization

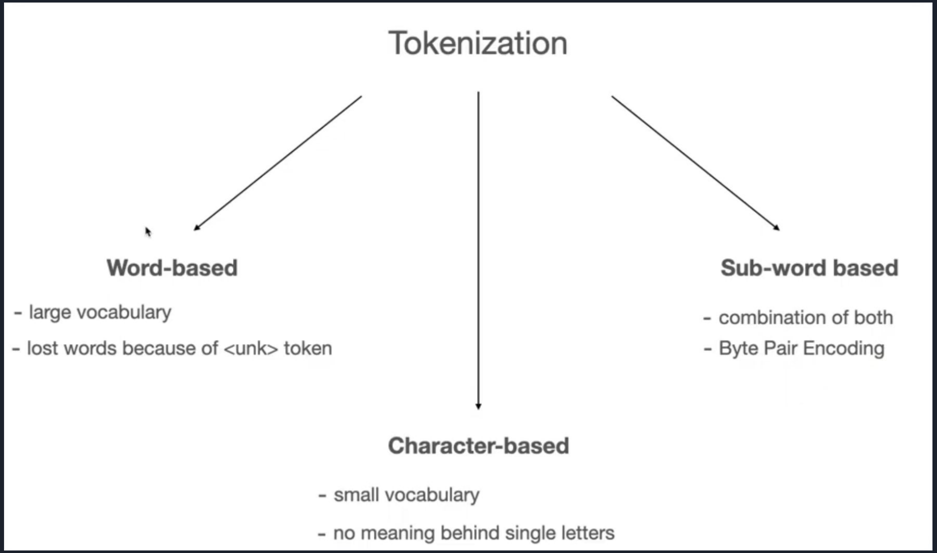
text -> sequence of vectors

Tokenization is the answer to : how to split text into discrete units

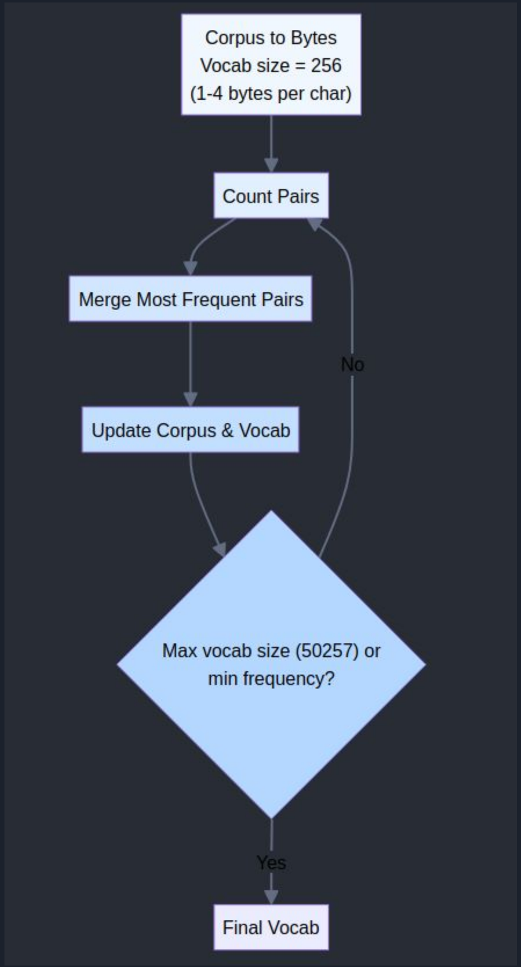
```
AI'll make recruitment fast, easy, efficient and unbia
sed. Go HrFlow! <|endoftext|>
```

Indices in vocab

```
20185, 1183, 787, 19624, 3049, 11, 2562, 11, 6942, 29
0, 46735, 13, 1514, 367, 81, 37535, 0, 220, 50256
```



Tokenization (BPE)



PS: a pre-tokenization step uses regex patterns to split text before applying BPE, this ensures consistent handling of:

- Whitespace
- Contractions (‘ve, ‘s, ‘ll)
- Punctuation
- Numbers
- Special characters

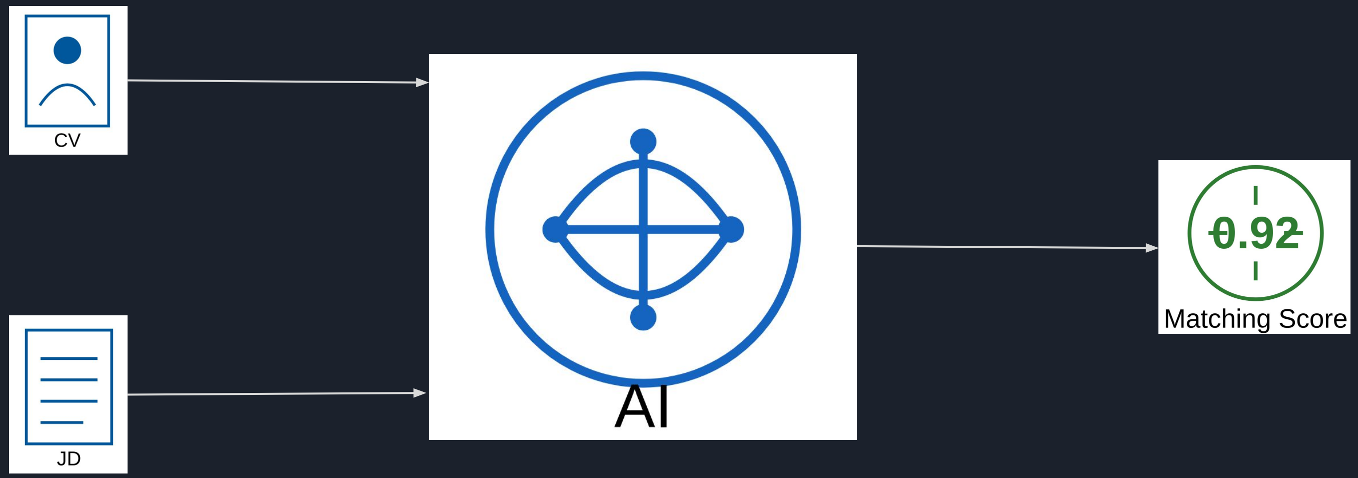
BPE example

- small vocab {a,b,c,d}
- long sequence : 11

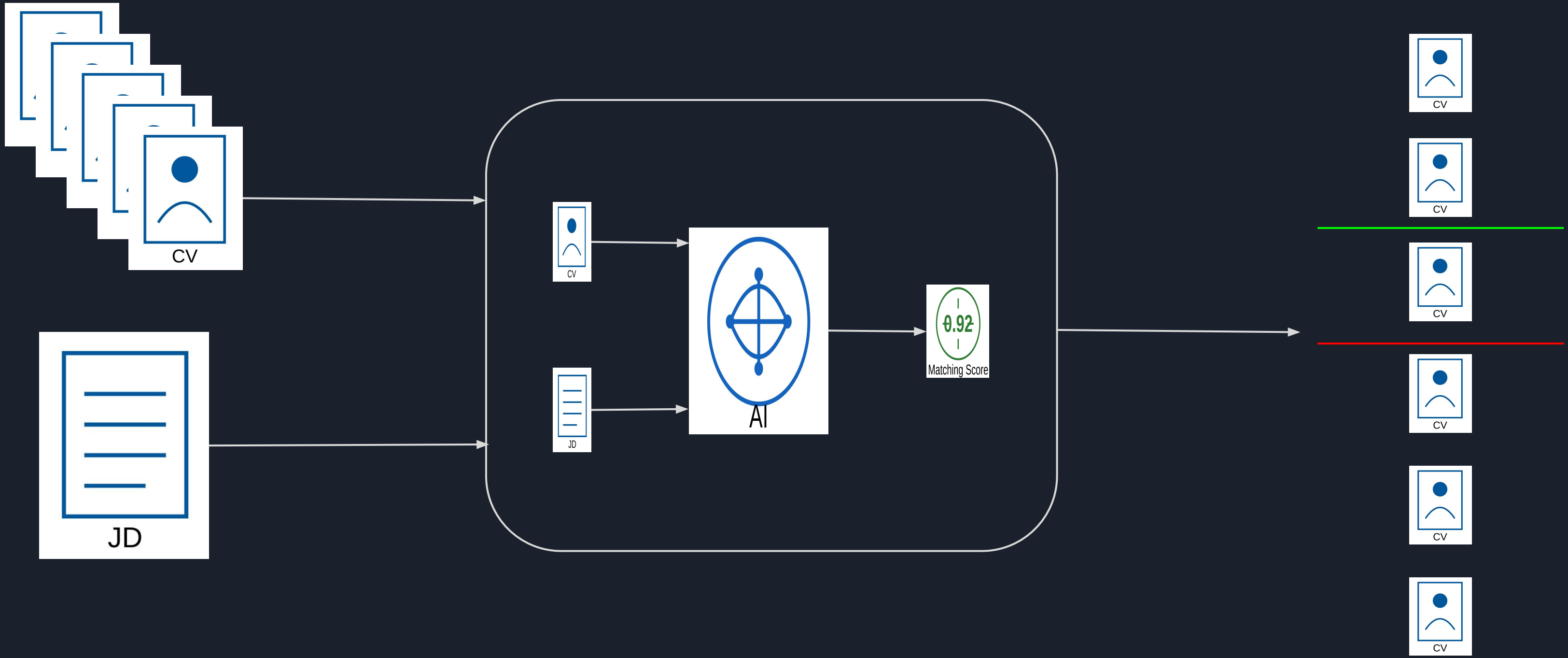
- large vocab {a,b,c,d, X, Y, Z}
- short sequence : 5

JD - CV scoring

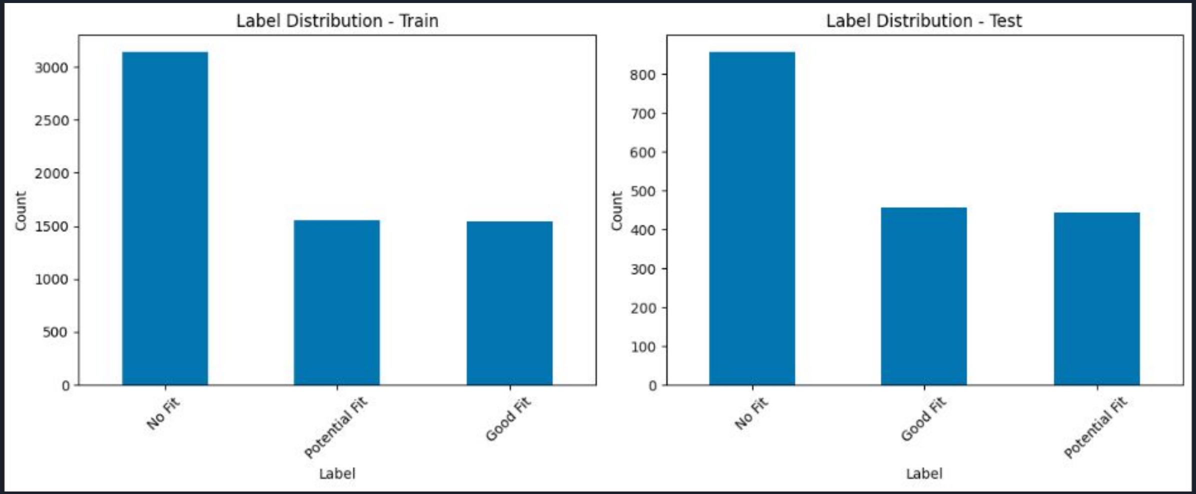
recruitment AI scoring



recruitment AI ranking

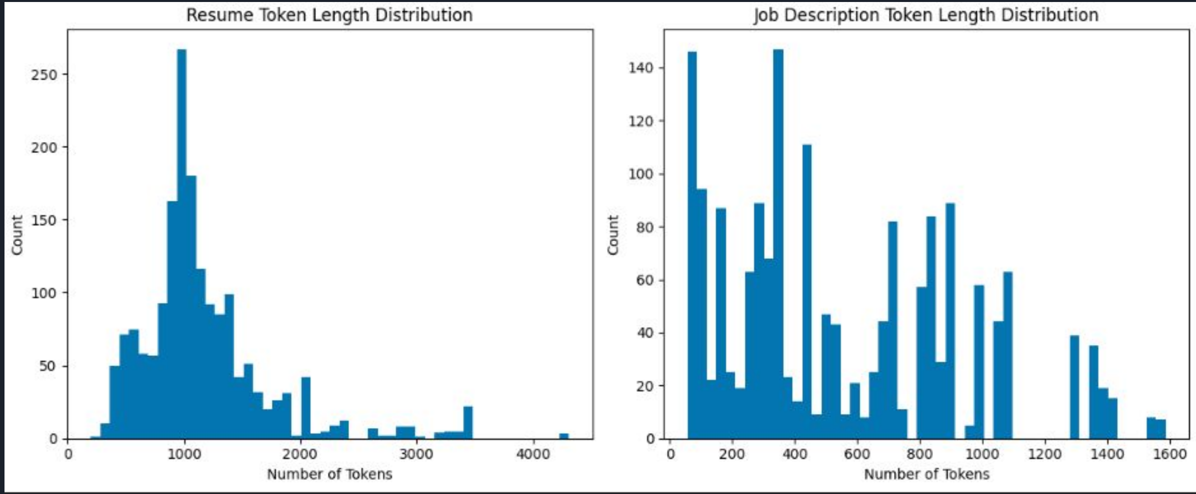


Dataset



Label distribution in dataset (cnamuangtoun/resume-job-description-fit)

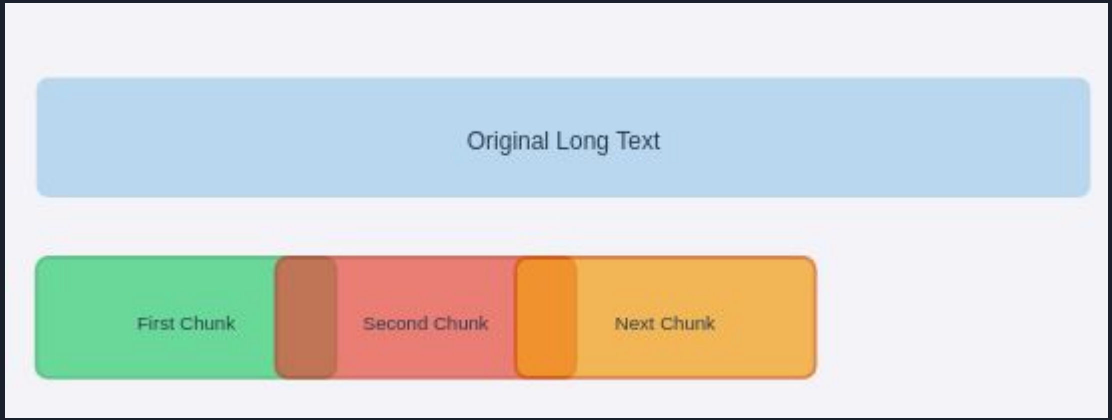
Number of tokens in inputs



Token count in train samples (CV left, JD right)

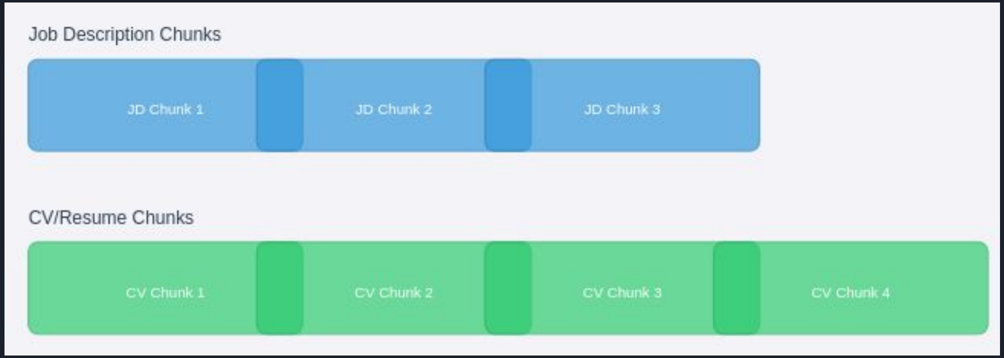
context size = 1024 not enough

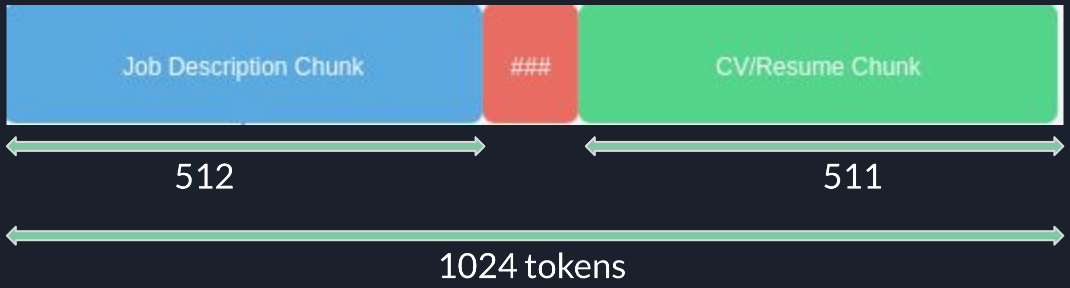
Chunking



dividing long text into 1024 token chunks

Train GPT on chunks





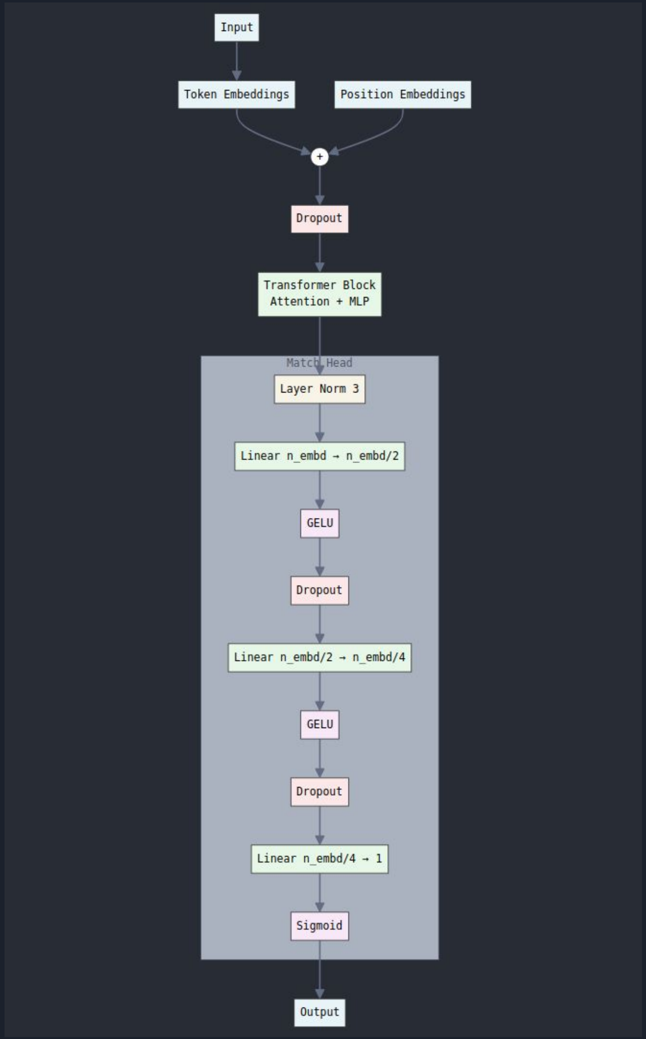
Fine-tuning GPT

Classification with three classes :
No Fit - Potential - Good Fit

Should take into account ordinality

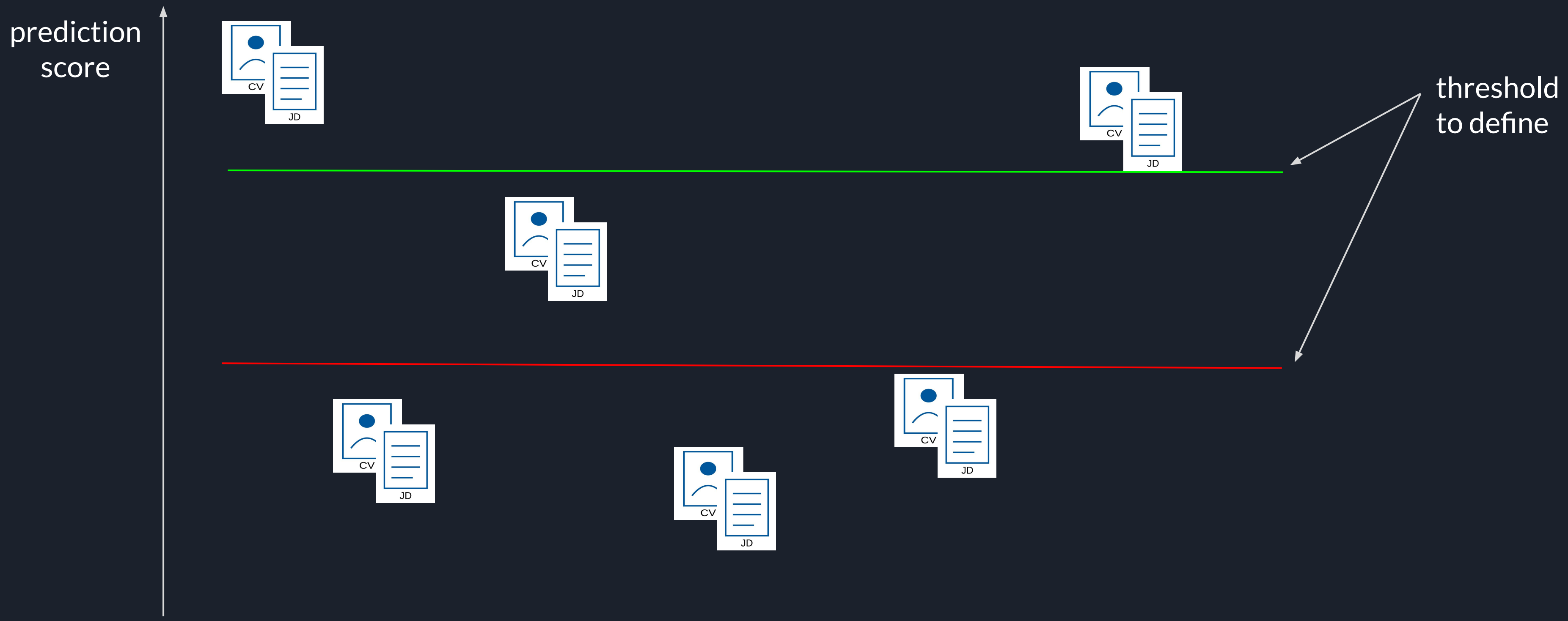
Solution : regression ! sigmoid + MSE

```
label_map={
            'No Fit': 0.0,
            'Potential Fit': 0.6,
            'Good Fit': 1.0
        }
```



Postprocessing

score(JD,CV) = avg(score(JD_chunk_i,CV_chunk_j))



Metrics

Precision, Recall, F1 per class

Weighted F1 Score : average F1 based on frequency - > used on val to keep best model

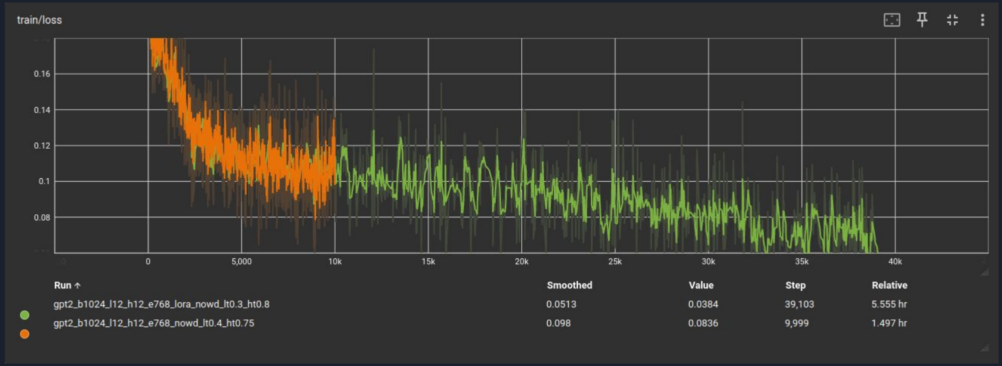
Confusion Matrix : for quick looks

Training

- lr = 5e-6 (with warm up)   x10 for match_head
- ADAM optimizer, reduced weight decay param later
- Batch size = 8   (best could do for gpt2 124M on rtx 4090)
- grad accum = 8   (64 batch size without compute gain)
- Implemented LoRA
- balanced train a little bit   (50-25-25)   - > (40,30,30)

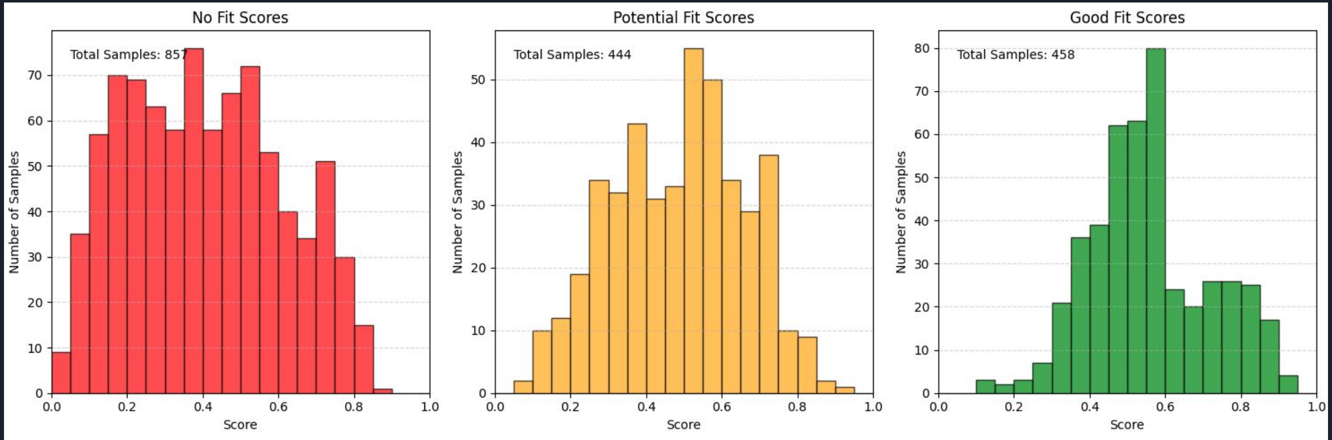
Train val Loss

val weighted F1 (top) , val loss (bottom)



train loss

Performance on test set



There’s room for improvement

Demo

Streamlit based POC demo :

[Local URL]

[Network URL]

Proof-of-concept

Model is learning patterns

Room for improvement

Prototyping pipeline ready

Research directions

Scaling, data quality, compute

Larger Context - Mask Attention

Contrastive learning

$$-\log \frac{\exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)}$$

Better chunking

More robust chunking strategies : experiences

$$W(E_{ij}) = f(\text{duration}(E_{ij}), \text{recentness}(E_{ij})) \cdot P(E_{ij}, J_k)$$

model

experience i

Interaction between chunks: “Chunk attention”

Self chunk attention (CV-CV, JD-JD),   Cross chunk attention (CV-JD, JD-CV)

Thank you for your time !

Let’s discuss