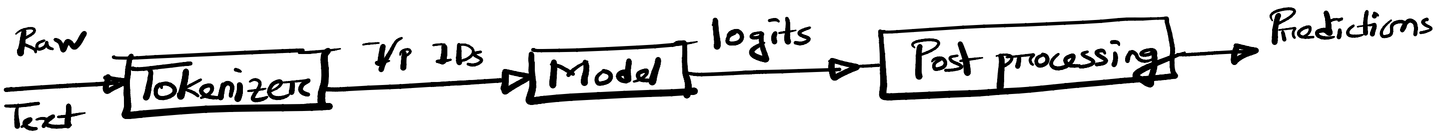
Main features of Huggingface's Xmer library:-

- Ease of use:-
- Flexibility
- Simplicity



Raw text — This course is amazing!

↓

Tokens — [this, course, is, amazing, !]

↓

Special tokens — [[CLS], this, course, is, amazing, !, [SEP]]

↓

i/p IDs — [101, 2023, - - - - - , 102]

Tokenizer

Preprocessing with a tokenizer:-

- Splitting i/p into (sub) words, symbols → tokens
- Mapping each token into integer
- + additional tokens like [CLS] [SEP]

Should be done exactly same way as the model PT.

Xmer models only use tensors as i/p.

For specifying the type of tensors to get back, we use

return_tensors = "pt"

এই argument এ কিছু না দিলে list of lists return হবে।

[illegible] argument ২ কিছু না দিলে list of lists return হবে।

Going through the model:-

downloaded & cached.

O/P:- hidden states/ features.

hidden states কে head এর i/p হিসেবে দেওয়া হয়।

diff. tasks এর জন্য same archi. but diff. head.

Inner এর O/P large হয়। 3D:-

Batch size: No. of seq. processed at a time.

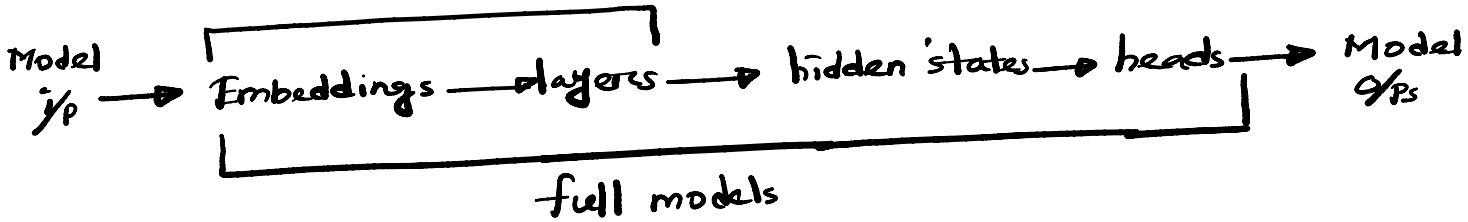
Seq. length: length of the numerical rep. of seq.

hidden size:- Vector rep. of each model i/p.

Model heads:- Making sense out of numbers.

Composed of one/few linear layers.

Transformer Net.



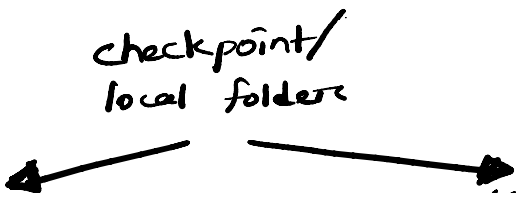
Postprocessing the O/P:-

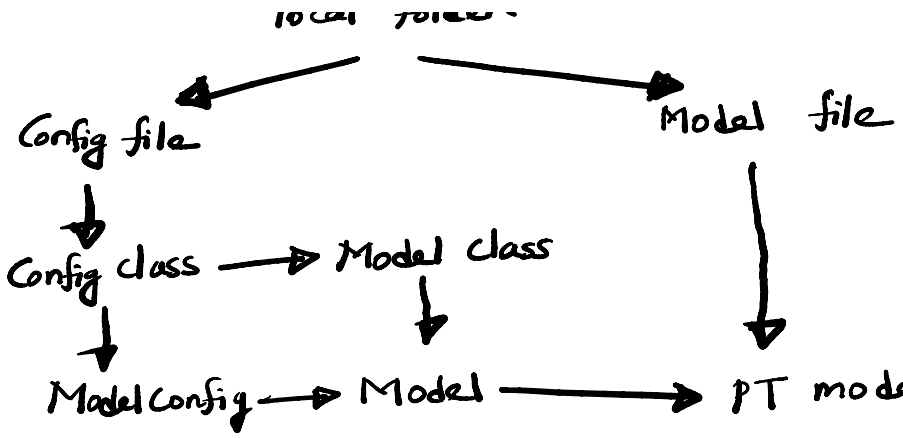
O/P of the model x make sense

Outputs logits

model.config.id2label

Instantiate a Transformers model:-





Wrapper:- prog. components simplifies usage/interaction with other S/W lib./API/system.

Section 2.3 Models:-

AutoModel & sim. wrappers Automatically guess appropriate model archi.

Loading a PT Xmer:-

BertModel. from_pretrained ("bert_base-uncased")

Weights downloaded & cached into
~/.cache/huggingface/transformers.

Saving Methods:-

model. save_pretrained("dir....")

dir.... — ২টা file save করবে

Config.json
- attributes necessary to build model archi.
- Contains metadata
- where checkpoints originated, Xmer version
- when last saved the checkpoint

} model Archi.

pytorch_model.bin
- State dir.
- Contains all weights

Using a Xmer for inference:-

Xmer model only process numbers that tokenizer generates.

- Xmer model only process numbers that tokenizer generates.
  Tokenizer casts i/p to appropriate framework tensors.

i/p seqs. → tokenizer → Vocab. indices (input IDs), list of lists

Tensors accepts rectangular shapes (Matrices)

input IDs —torch.tensor( )→ tensors, i/p to Xmer models.

Section 2.1 Tokenizers:—

Core components of NLP
translate text into numbers (data that Xmer model can understand).
goal:- find most meaningful & smallest rep.

Word-based tokenizer:—

easy to set up
use only with few rules
decent performance often
different ways to split texts.
by whitespace using Python's split() fn.

Disadv:- may results very large vocab.
inflections are identified as diff. words:—
dog, dogs
run, running
unknown/[UNK]
↑ [UNK], ↑ losses of info.

Character based tokenizers:—

Vocab. smaller (256)
fewer OOV tokens
charac. x convey meaning
large no. tokens (1 word → 10+ tokens)
whole english words (1,70,000) with just 256 chars.

Whole english words (1,70,000) with just 256 chars.

Subword tokenization:-

frequently used words ✗ Split
rare " ✓ "
good coverage + small vocab.
~ no unknown tokens
helpful in Turkish lang. where long complex words can be formed

Loading & saving of tokenizers:-

from_pretrained()
save_pretrained()

Encoding:- 2 steps

tokenization
Conversion to i/p IDs

Instantiate same tokenizer as the model → same rules used to PT.
" vocab.

Tokenization:- Auto/BertTokenizer.from_pretrained( )
Tokens to i/p IDs:- tokenizer.convert_tokens_to_ids( )
Decoding:- tokenizer.decode( )

Section 2.5 Handling multiple seq.

How to handle multiple seq. of variable length?
Vocab. only i/p to allow model to work well?
Too long seq.?

Model expects a batch of i/ps:-

seq. → list of no. → tensor → i/p to the model.
✗ mere expects multiple sentences by default

Batching is the act of sending multiple sentences through the model.

1 sentence এলেও, you can batch with a single sentence.

= [ids, ids]

= [ids, ids]

issue 1:- i/p হিসেবে কমের multiple sentence expect করে

" 2:- 2 sentences having diff. length.
tensors need to rectangular shape.
pad the i/ps.

issue 3:- padding করার কারণে model এর logit same আসে না each token কে contextualize করার জন্য।
Soln:- Attn mask use করে padding কে ignore করতে বলা।

Attn Masks:- 1s indicating ⟶ attended into
0s " ⟶ ignore.

Longer seq:-
limit to the lengths of seq. ⟶ 512/1024.
seq. > 512/1024
Crash
Soln:- use a model with a longer supported seq. length, Longformer LED
truncate your seq.

Section 2.6:- Putting it all together:-

tokenizer( ) fn directly use করা
can handle everything related to tokenization.
multiple seq.
padding : longest, max_length
Conversion to specific framework tensors
pt = Pytorch tensors
tf = Tensorflow "
np = Numpy arrays.

2.7

Basic usage completed:-

2.8 Quiz:-