Introduction

Monday, May 12, 2025 3:42 PM

Transformer models:

- Large in size
- Have millions to tens of billions of parameters
- Training and deploying them is a complicated process.
- Each model has its own implementation, trying them all out is no easy task.

Transformer library solves these issues.
It provides a single API through which any model can be

- Trained
- Loaded
- Saved

Behind the pipeline

Monday, May 12, 2025 3:42 PM

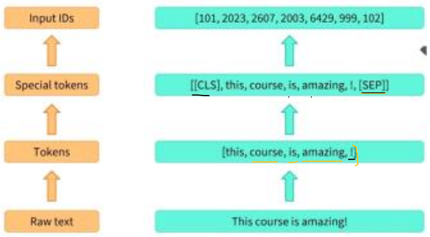
Let's look at what happens when we call pipeline function.

Preprocessing with a tokenizer

Similar to neural networks, transformer models can't process raw text. The first step is to convert text inputs into numbers.

What does a tokenizer do?

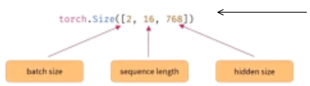
1. Split the text into tokens
2. Assign an integer to each token
3. Add additional tokens as per the requirements of the LLM



Going through the model

- Outputs only hidden state

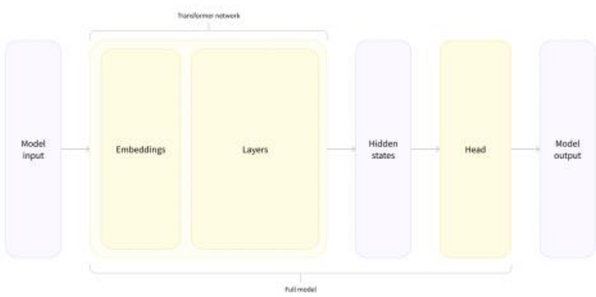
```
DistilBertModel(
  (embeddings): Embeddings(
    (word_embeddings): Embedding(30522, 768, padding_idx=0)
    (position_embeddings): Embedding(512, 768)
    (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
    (dropout): Dropout(p=0.1, inplace=False)
  )
  (transformer): Transformer(
    (layer): ModuleList(
      (0-5): 6 x TransformerBlock(
        (attention): DistilBertSdpaAttention(
          (dropout): Dropout(p=0.1, inplace=False)
          (q_lin): Linear(in_features=768, out_features=768, bias=True)
          (k_lin): Linear(in_features=768, out_features=768, bias=True)
          (v_lin): Linear(in_features=768, out_features=768, bias=True)
          (out_lin): Linear(in_features=768, out_features=768, bias=True)
        )
        (sa_layer_norm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
        (ffn): FFN(
          (dropout): Dropout(p=0.1, inplace=False)
          (lin1): Linear(in_features=768, out_features=3072, bias=True)
          (lin2): Linear(in_features=3072, out_features=768, bias=True)
          (activation): GELUActivation()
        )
        (output_layer_norm): LayerNorm((768,), eps=1e-12,
elementwise_affine=True)
))))
```



← Features/hidden vector (contains contextual understanding)

In simple words
Batch size: no. of sentences
Sequence length: no. of tokens
Hidden size: the vector dimension of each model input.

Model heads: Making sense out of numbers



Other available architectures in Transformer Library

- Model (retrieve the hidden states)
- ForCausalLM
- ForMaskedLM
- ForMultipleChoice
- ForQuestionAnswering
- ForSequenceClassification
- ForTokenClassification

- And others ...

We are going to work with `AutoModelForsequenceClassification`

AutoModel

```
DistilBertModel(
  (embeddings): Embeddings(
    (word_embeddings): Embedding(30522, 768, padding_idx=0)
    (position_embeddings): Embedding(512, 768)
    (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
    (dropout): Dropout(p=0.1, inplace=False)
  )
  (transformer): Transformer(
    (layer): ModuleList(
      (0-5): 6 x TransformerBlock(
        (attention): DistilBertSdpaAttention(
          (dropout): Dropout(p=0.1, inplace=False)
          (q_lin): Linear(in_features=768, out_features=768, bias=True)
          (k_lin): Linear(in_features=768, out_features=768, bias=True)
          (v_lin): Linear(in_features=768, out_features=768, bias=True)
          (out_lin): Linear(in_features=768, out_features=768, bias=True)
        )
        (sa_layer_norm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
        (ffn): FFN(
          (dropout): Dropout(p=0.1, inplace=False)
          (lin1): Linear(in_features=768, out_features=3072, bias=True)
          (lin2): Linear(in_features=3072, out_features=768, bias=True)
          (activation): GELUActivation()
        )
        (output_layer_norm): LayerNorm((768,), eps=1e-12,
elementwise_affine=True)
))))
```

AutoModelForSequenceClassification

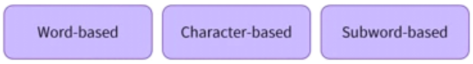
```
DistilBertForSequenceClassification(
  (distilbert): DistilBertModel(
    (embeddings): Embeddings(
      (word_embeddings): Embedding(30522, 768, padding_idx=0)
      (position_embeddings): Embedding(512, 768)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False))
    (transformer): Transformer(
      (layer): ModuleList(
        (0-5): 6 x TransformerBlock(
          (attention): DistilBertSdpaAttention(
            (dropout): Dropout(p=0.1, inplace=False)
            (q_lin): Linear(in_features=768, out_features=768, bias=True)
            (k_lin): Linear(in_features=768, out_features=768, bias=True)
            (v_lin): Linear(in_features=768, out_features=768, bias=True)
            (out_lin): Linear(in_features=768, out_features=768, bias=True)
          )
          (sa_layer_norm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
          (ffn): FFN(
            (dropout): Dropout(p=0.1, inplace=False)
            (lin1): Linear(in_features=768, out_features=3072, bias=True)
            (lin2): Linear(in_features=3072, out_features=768, bias=True)
            (activation): GELUActivation()
          )
          (output_layer_norm): LayerNorm((768,), eps=1e-12,
elementwise_affine=True)))))
  (pre_classifier): Linear(in_features=768, out_features=768, bias=True)
  (classifier): Linear(in_features=768, out_features=2, bias=True)
  (dropout): Dropout(p=0.2, inplace=False)
)
```

Tokenizers

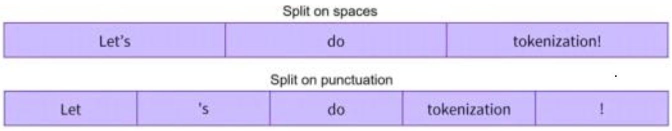
Friday, May 16, 2025 7:06 AM

Purpose: Translate text into data that can be processed by the model.
- Models only process numbers.
- Need a way to convert raw text into numbers that can be fed into models.

Types of Tokenization

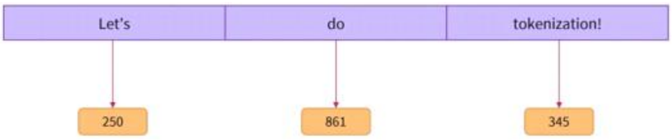


Word Based
- Simply split on spaces/punctuation. Other rules can also be added.

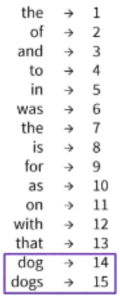


Let's do tokenization!

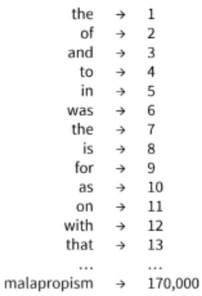
- Each token has a specific ID.



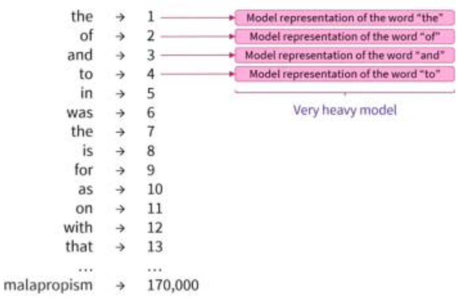
- Very similar words have entirely different meanings.



- The vocabulary can end up very large.



- Large vocabulary results in heavy models.

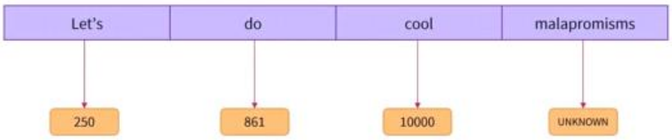


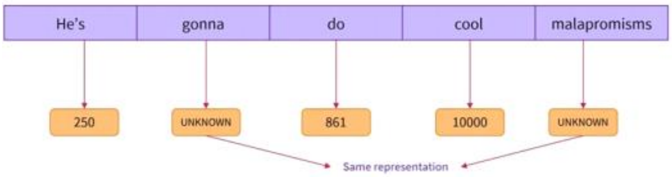
500,000 words in the English language.

- We can limit the amount of words we add to the vocabulary.

the	→	1
of	→	2
and	→	3
to	→	4
in	→	5
was	→	6
the	→	7
is	→	8
for	→	9
as	→	10
on	→	11
with	→	12
that	→	13
...		...
hug	→	10,000

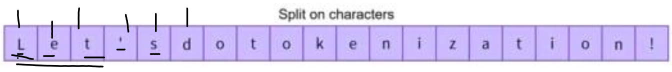
- Limiting words lead to the problem of unknown words.



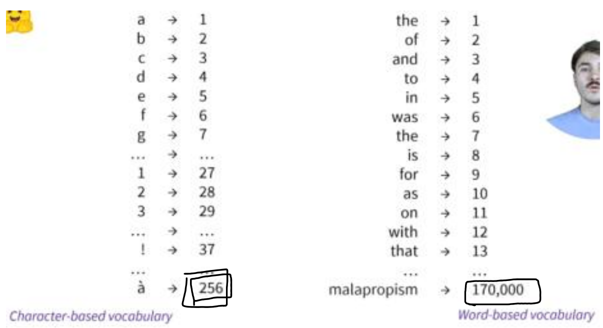


One way to reduce amount of unknown words is to use character level tokenization.

Character Level Tokenization



Provides two primary benefits
- Smaller vocabulary
- Fewer out of vocabulary tokens



768

I love you

I	Love	You

768 → 3 x 768

I	L	O	V	E	Y	O	U

9 x 768

1024 x 768

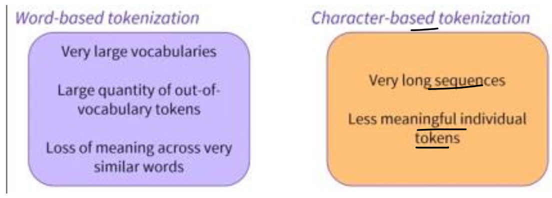
- However, this approach is not perfect either.
- When text is split based on characters, it loses the meaning. A character doesn't carry a meaning on its own.
- Remember that this is also language specific. In Chinese language, a character carries more meaning than a word.
- Another aspect to consider is that, model has to pre-process a very large amount of tokens every time. It wasn't a case with word level tokenization.

Why not the best of both worlds?

Subword tokenization

- Splitting words into sub-words.

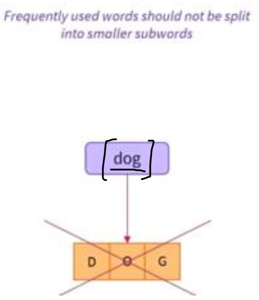
- Splitting words into sub-words.

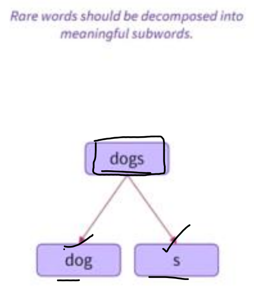


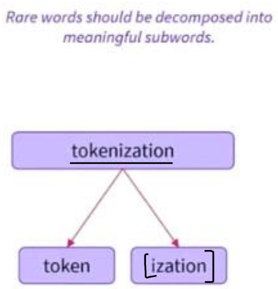
Find a middle ground between word level and character level tokenization.

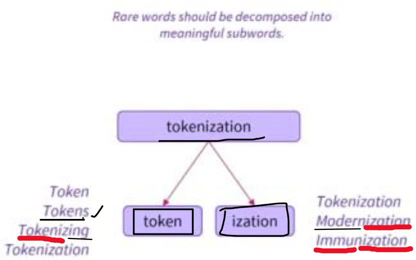
Principles of Sub-word tokenization

- Frequently used words shouldn't be split into smaller sub-words.
- Rare words should be split into meaningful sub-words.

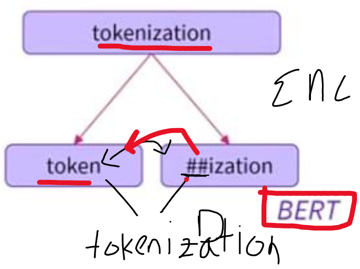








- Subword tokenization algorithms can identify start of word tokens



- Keeps semantic meaning
- useful in agglutinative languages such as Turkish, where you can form (almost) arbitrarily long complex words by stringing together subwords.

Word	Breakdown	Meaning
1 ev	ev 1	house
2 evler	ev + ler 2	houses
3 evimiz	ev + imiz 3	our house
4 evlerimiz	ev + ler + imiz	our houses
5 evlerimizde	ev + ler + imiz + de 4	in our houses
6 evlerimizden	ev + ler + imiz + den 5	from our houses
7 evlerimizdenmiş	ev + ler + imiz + den + miş 6	(apparently) from our houses

Examples:

Encoding:
Process of converting raw text into token IDs.
It is done in two steps.
1. *Raw text to tokens*
2. *Tokens to token IDs.*

Decoding:
Process of converting token IDs into text.

Models

Friday, May 16, 2025 8:28 AM

Look closely at creating and using a model.

- Use *AutoModel* class: allows to download any model from a checkpoint.
- *AutoModel* class and all of its relatives are simply wrappers over the wide variety of models available in the library.

However, if you know the type of model, you can use the class that defines its architecture.

AutoConfig Class

- Allows you to instantiate the configuration of a pretrained model from any checkpoint.

Creating a Transformer

- Initiate a BERT model.

Configuration	Architecture	Weights

1. AutoConfig (input = "any model's name")
2. Model Specific Class. (BertConfig)

Architecture + Weights
AutoModel (any model)
BertModel / GPTModel ()

Handling multiple sequences

Friday, May 16, 2025 7:06 PM

We will answer following questions
- How do we handle multiple sequences?
- How do we handle multiple sequences of different lengths?
- Are vocabulary indices the only inputs that allow a model to work well?
- Is there such a thing as too long a sequence?

What were we doing previously?

"I've been waiting for a HuggingFace course my whole life."

i	'	ve	been	waiting	for	a	hugging	##face	course	my	whole	life	.

← tokens

1045	1005	2310	2042	3403	2005	1037	17662	12172	....	1012

← ids

[1045, 1005, 2310, 2042, 3403, 2005, 1037, 17662, 12172, 2607, 2026, 2878, 2166, 1012]

Passing these ids will produce an error.

All models expect a 2D input by default and these IDs are 1D.
[[1045, 1005, 2310, 2042, 3403, 2005, 1037, 17662, 12172, 2607, 2026, 2878, 2166, 1012]]

Batching Inputs

"I am alive.", --> x1
"I am doing very well." --> x2

single_input = [x1] = ["I", "am", "alive", "."]
single_input = [x2]

batched_inputs = [x1, x2] = [["I", "am", "alive", "."],
[["I", "am", "doing", "very", "well", "."]

In general, batched_inputs = [x1, x2, ..., xn]

Padding inputs

inputs = [[x1, x2, x3], ---> sequence 1, three ids
[x4, x5]] ---> sequence 2, two ids

Feeding this "inputs" tensor will produce an error. All input
sequences must have same size.

inputs = [[x1, x2, x3], ---> sequence 1, three ids
[x4, x5, 0]] ---> sequence 2, three ids

Attention Masks
- It tells the attention layers where to put focus and where to not.

Let's understand with an example

I	Love	You	.	I	Hate	You	pad
1	2	3	4	1	5	3	0
1	1	1	1	1	1	1	0

Longer Sequences
Models can process a limited amount of tokens at a time.
This amount is known as context length/sequence length.