DeepXplain Summer School 2025

Francielle Vargas
Ph.D. and M.Sc. in Computer Science and Computational Mathematics
Artificial Intelligence - Natural Language Processing

UNIVERSITY
OF OSLO

Sentiment Analysis Task

Building a Typical NLP Pipeline

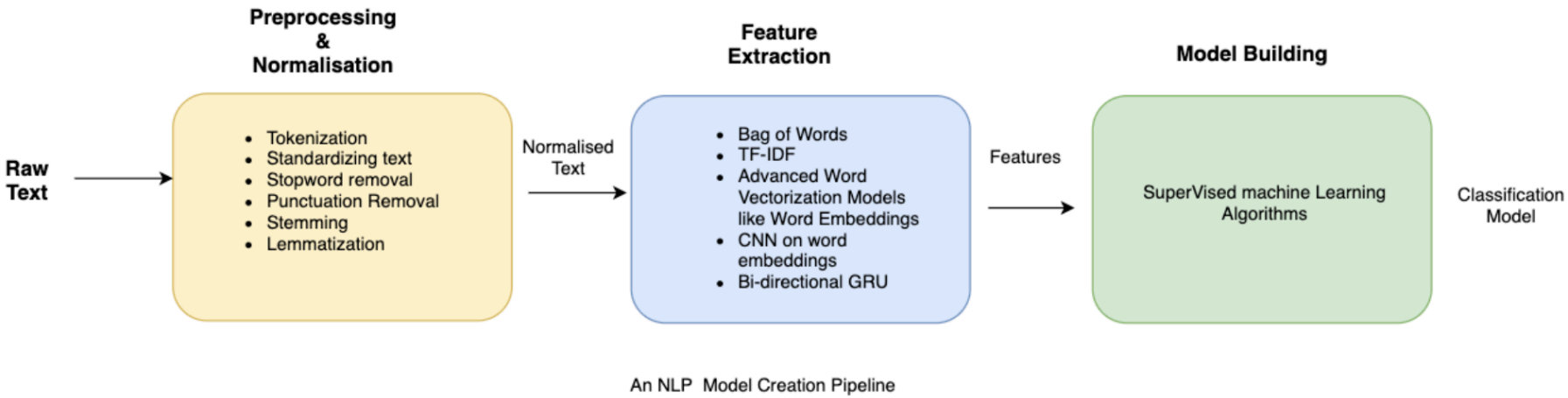

Figure: NLP Pipeline for Sentiment Analysis using Classical Supervised Machine Learning.

Sentiment Analysis Task

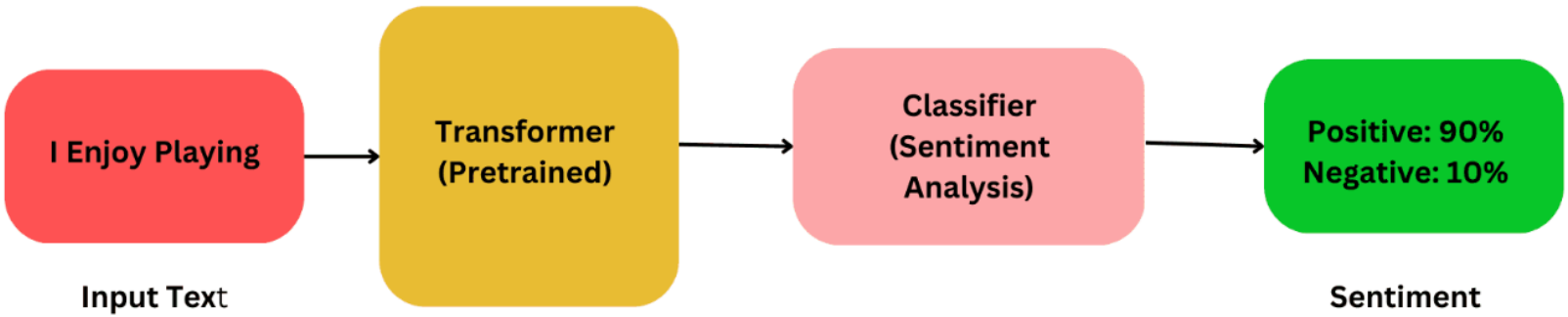

Figure: NLP Pipeline for Sentiment Analysis using Fine-Tuning.

Fine-Tuning

- Think of a pre-trained model like a person who already knows English very well. Fine-tuning is like training that person to be a movie reviewer or a legal document analyst, depending on your task.

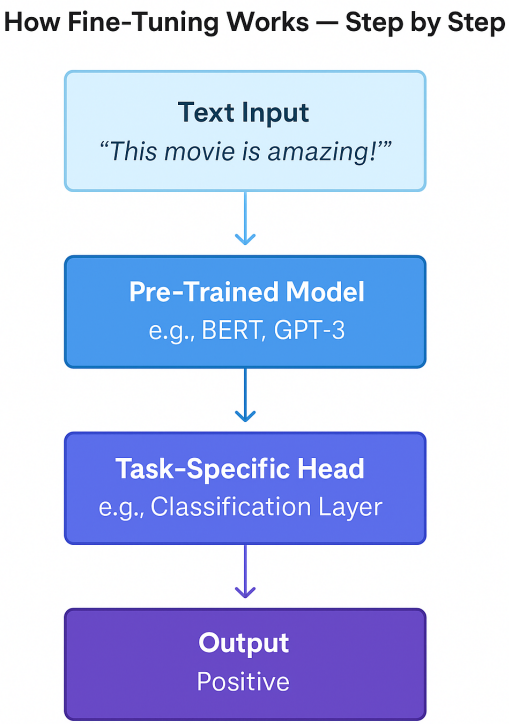

Figure: Fine-tuning Step-By-Step.

Transformers: BERT (Bidirectional Encoder Representations from Transformers)

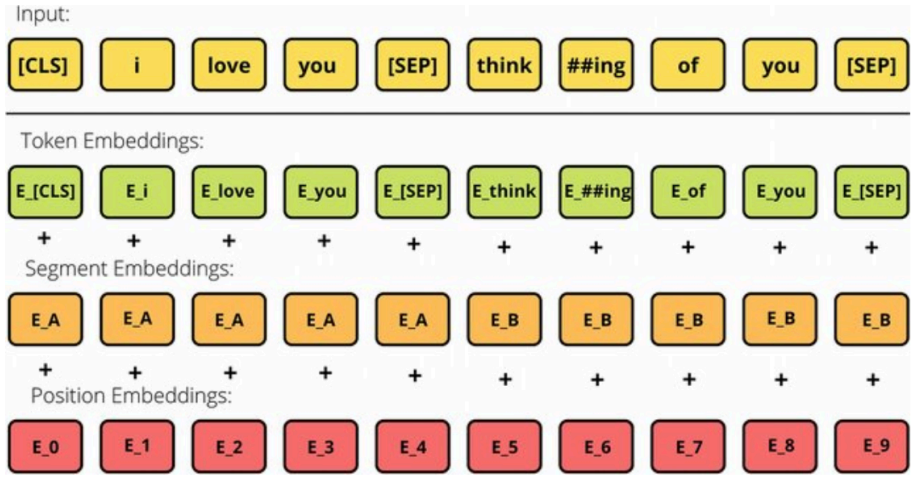

Figure: Input representation in BERT.

Task 1: To empirically and conceptually evaluate the main advancement introduced by BERT, namely, the use of bidirectional attention and pre-training with Masked Language Modeling, in comparison to earlier architectures such as RNNs (LSTM/GRU) and static embeddings (Word2Vec, GloVe), Bag-of-Words (BoW).

Task 2: Understanding the function of each BERT input.

```
from transformers import BertTokenizer

tokenizer = BertTokenizer.from_pretrained("bert-base-uncased")

encoded = tokenizer(
    "This is sentence A.",
    "And this is sentence B.",
    padding="max_length",
    truncation=True,
    max_length=10,
    return_tensors="pt"
)

print(encoded.keys())
#dict_keys(['input_ids', 'token_type_ids', 'attention_mask'])
```

```
Be aware, overflowing tokens are not returned for the setting you have chosen, i.e. sequence pairs with the 'longest_first' truncation strategy. So the returned list will always be empty even if some tokens have been removed.
KeysView({'input_ids': tensor([[ 101, 2023, 2003, 6251, 1037,  102, 1998, 2023, 2003,  102]]), 'token_type_ids': tensor([[0, 0, 0, 0, 0, 0, 1, 1, 1, 1]]), 'attention_mask': tensor([[1, 1, 1, 1, 1, 1, 1, 1, 1, 1]])})
```

Figure: BERT (Bidirectional Encoder Representations from Transformers) takes three main inputs for each text example it processes. These inputs are vectors that represent different types of information about the input text and all have the same length (usually max_length, such as 512 tokens).