LargeScaleNLP: An Introduction

There are 3 **prominent** reasons for the revolution that we witnessed in NLP starting 2017:

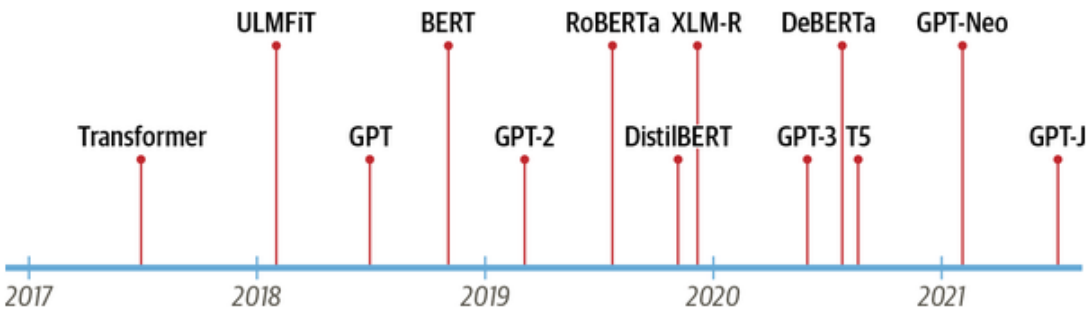
1. The seminal **"Attention is all you need"** paper, which introduced Transformer architecture.
2. Availability of large models, pre-trained on generic texts which can be finetuned (transfer-learning) to specific task at hand.
3. Model Hubs like HuggingFace which host thousands of state-of-the art models which can be downloaded and used.

In short the 3 key ideas that fuelled NLP growth are: **Attention Mechanism**, **Transfer Learning** and **Scaling**.

In [1]:
```
from IPython.display import Image, display
```

Timeline of major developments

In [2]:
```
display(Image(filename='./Pics_01_TransformerTimeline.png', width=600, height=200))
```



ULMFiT (Universal Language Model Fine-tuning for Text Classification: https://arxiv.org/abs/1801.06146) needs a special mention as it catalyzed the concept of **Tranfer-Learning** in NLP. It showed that training long short-term memory (LSTM) networks on a very large and diverse corpus could produce state-of-the-art text classifiers with little labeled data.

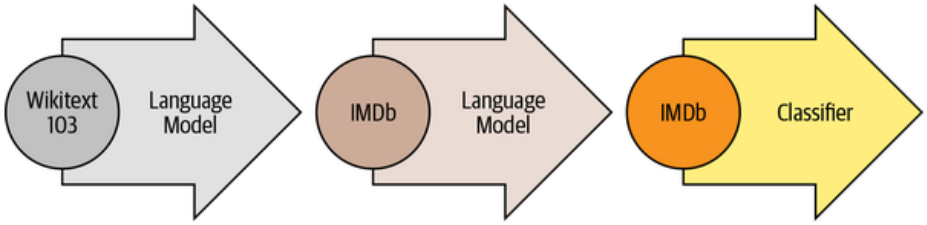
Tranfer-Learning in NLP, as pioneered by **ULMFiT** involves 3 main steps:

1. **Pre-training**: In NLP pre-training is done via **Language Modelling Task** (which states simply is predicting the next token given previous tokens). Pre-training can be done in self-supervised fashion (on a huge text corpus like Wikipedia)
2. **Domain Adaption**: We take the pre-trained model and adapt it to the domain of our final task. Here also the task is **Language Modelling** only, just that the dataset is

Domain specific (like IMDB).

3. **Fine-Tuning**: In this step, the language model is fine-tuned with a classification layer for the target task:

```
In [3]: display(Image(filename='./Pics_02_TransferLearningInNLP.png', width=500, height=100
```



HuggingFace Ecosystem (Hub and Libraries):

The HuggingFace Ecosystem provides 2 main things to NLP practioners:

1. **HuggingFace 🤗 Hub**: For pre-trained **models** (https://huggingface.co/models), **datasets** (https://huggingface.co/datasets), **metrics** (https://huggingface.co/metrics) and **docs** (https://huggingface.co/docs). The hub has a few more things like **Spaces** (to host AI apps) and **Posts** (for posting articles, questions etc. to the community).

2. **4 NLP Libraries** for using/fine-tuning the models and datasets hosted in the Hub:

   A. HuggingFace 🤗 **Transformer**:

   - It provides a **standardized interface** to a wide range of transformer models on the hub as well as code and tools to adapt these models to new use cases.
   - It aslo provides task-specific **heads** to easily fine-tune transformers on downstream tasks such as text classification, named entity recognition, and question answering.

   B. HuggingFace 🤗 **Dataset**:

   - It offers similar functionality for data processing as Pandas but is designed from the ground up for tackling large datasets.
   - It provides a **standard interface** for datasets available on the Hub.
   - It also provides smart caching and avoids RAM limitations by leveraging a special mechanism called **memory mapping** that stores the contents of a file in virtual memory and enables multiple processes to modify a file more efficiently.

   C. HuggingFace 🤗 **Tokenizers**:

   - It provides many tokenization strategies.

- It also takes care of all the pre- and postprocessing steps, such as normalizing the inputs and transforming the model outputs to the required format.
- The interface for loading tokenizers is same as loading pretrained model weights with Transformers.

D. HuggingFace 🤗 **Accelerate**:

- It gives us full control over the training loop and allows us to train large-scale transformers entirely from scratch.

Exploring the Transformer Library of HuggingFace

Transformer Library allows to interact with the library at various levels of abstraction. The most basic object is ***pipeline()* function**, which abstracts away all the steps needed to convert raw text into a set of predictions from a **fine-tuned model**.

In [4]:
```
from transformers import pipeline
import pandas as pd
```

NER (Named Entity Recognition)

In [5]:
```
text = """Dear Amazon, last week I ordered an Optimus Prime action figure
from your online store in Germany. Unfortunately, when I opened the package,
I discovered to my horror that I had been sent an action figure of Megatron
instead! As a lifelong enemy of the Decepticons, I hope you can understand my
dilemma. To resolve the issue, I demand an exchange of Megatron for the
Optimus Prime figure I ordered. Enclosed are copies of my records concerning
this purchase. I expect to hear from you soon. Sincerely, Bumblebee."""

ner_tagger = pipeline("ner", aggregation_strategy="simple")
outputs = ner_tagger(text)
pd.DataFrame(outputs)
```

```
No model was supplied, defaulted to dbmdz/bert-large-cased-finetuned-conll03-english
and revision 4c53496 (https://huggingface.co/dbmdz/bert-large-cased-finetuned-conll0
3-english).
Using a pipeline without specifying a model name and revision in production is not r
ecommended.
Some weights of the model checkpoint at dbmdz/bert-large-cased-finetuned-conll03-eng
lish were not used when initializing BertForTokenClassification: ['bert.pooler.dens
e.bias', 'bert.pooler.dense.weight']
- This IS expected if you are initializing BertForTokenClassification from the check
point of a model trained on another task or with another architecture (e.g. initiali
zing a BertForSequenceClassification model from a BertForPreTraining model).
- This IS NOT expected if you are initializing BertForTokenClassification from the c
heckpoint of a model that you expect to be exactly identical (initializing a BertFor
SequenceClassification model from a BertForSequenceClassification model).
Device set to use cuda:0
```

Out[5]:

	entity_group	score	word	start	end
0	ORG	0.879010	Amazon	5	11
1	MISC	0.990859	Optimus Prime	36	49
2	LOC	0.999755	Germany	90	97
3	MISC	0.556569	Mega	208	212
4	PER	0.590257	##tron	212	216
5	ORG	0.669692	Decept	253	259
6	MISC	0.498350	##icons	259	264
7	MISC	0.775361	Megatron	350	358
8	MISC	0.987854	Optimus Prime	367	380
9	PER	0.812096	Bumblebee	502	511

By default, this pipeline() selects a particular pretrained model that has been fine-tuned for "**ner**" task. We can also choose a **particular** model from the Hub to use in a pipeline for a specific task — say, **Question Answering**.

Question Answering

In [6]:
```
reader = pipeline("question-answering", model = "deepset/roberta-base-squad2")
question = "What does the customer want?"
outputs = reader(question=question, context=text)
pd.DataFrame([outputs])
```

```
Device set to use cuda:0
```

Out[6]:

	score	start	end	answer
0	0.074738	335	358	an exchange of Megatron

Some of the currently available pipelines are:

- feature-extraction (get the vector representation of a text)
- fill-mask
- ner (named entity recognition)
- question-answering
- sentiment-analysis
- summarization
- text-generation
- translation
- zero-shot-classification