# Introduction

* **!pip install transformers**
* ***Virtual environments*** which are self-contained directory trees that each contain a Python installation with a particular Python version alongside all the packages the application needs
* **pip install "transformers[sentencepiece]" 🡪** development version of using pip package manager.

# Natural Language Processing

* NLP is a field of linguistics and machine learning focused on understanding everything related to human language.
* The aim of NLP tasks is not only to understand single words individually, but to be able to understand the context of those words.
* NLP isn’t limited to written text though.
* It also tackles complex challenges in speech recognition and computer vision, such as generating a transcript of an audio sample or a description of an image.

The following is a list of common NLP tasks, with some examples of each:

1. **Classifying whole sentences**: Getting the sentiment of a review, detecting if an email is spam, determining if a sentence is grammatically correct or whether two sentences are logically related or not.
2. **Classifying each word in a sentence:** Identifying the grammatical components of a sentence (noun, verb, adjective), or the named entities (person, location, organization).
3. **Generating text context:** Completing a prompt with auto-generated text, filling in the blanks in a text with masked words.
4. **Extracting and answer from text:** Given a question and a context, extracting the answer to the question based on the information provided in the context.
5. **Generating a new sentence from an input text:** Translating a text into another language, summarizing a text.

# Why NLP is challenging?

* Computers don’t process information in the same way as humans.
* For example, when we read the sentence “I am hungry,” we can easily understand its meaning. Similarly, given two sentences such as “I am hungry” and “I am sad,” we’re able to easily determine how similar they are.
* For machine learning (ML) models, such tasks are more difficult.
* The text needs to be processed in a way that enables the model to learn from it.
* As language is complex, we need to think carefully about how this processing must be done.

# Transformers

* Transformer models are used to solve all kinds of NLP tasks.
* Facebook AI, google AI, Microsoft, Grammarly, are some of the companies and organizations using Hugging Face and Transformer models, who also contribute back to the community by sharing their models.
* The Transformers library  provides the functionality to create and use those shared models.
* The Model Hub contains thousands of pretrained models that anyone can download and use.
* One can also upload your own models to the Hub.

# Working with Pipelines

* The most basic object in the 🤗 Transformers library is the pipeline()  function.
* It connects a model with its necessary preprocessing and postprocessing steps, allowing us to directly input any text and get an intelligible answer.
* We can even pass several sentences!
* There are three main steps involved when you pass some text to a pipeline:

1. The text is preprocessed into a format the model can understand.
2. The preprocessed inputs are passed to the model.
3. The predictions of the model are post-processed, so you can make sense of them.

* Some of the currently available pipelines are:

1. Feature-extraction (get the vector representation of a text)
2. fill-mask
3. ner (named entity recognition
4. question-answering
5. sentiment-analysis
6. summarization
7. text-generation
8. translation
9. zero-shot-classification

# Zero-shot Classification

* A more challenging task where we need to classify texts that haven’t been labelled.
* This is a common scenario in real-world projects because annotating text is usually time-consuming and requires domain expertise.
* For this use case, the zero-shot-classification pipeline is very powerful: it allows you to specify which labels to use for the classification, so you don’t have to rely on the labels of the pretrained model.
* You’ve already seen how the model can classify a sentence as positive or negative using those two labels — but it can also classify the text using any other set of labels you like.
* This pipeline is called zero-shot because you don’t need to fine-tune the model on your data to use it.
* It can directly return probability scores for any list of labels you want.

# Text-Generation

* The main idea here is that you provide a prompt and the model will auto-complete it by generating the remaining text.
* This is similar to the predictive text feature that is found on many phones.
* Text generation involves randomness, so it’s normal if you don’t get the same results .
* ou can control how many different sequences are generated with the argument  num\_return\_sequences and the total length of the output text with the argument  max\_length.
* On Models Hub, you can refine your search for a model by clicking on the language tags, and pick a model that will generate text in another language.
* The Model Hub even contains checkpoints for multilingual models that support several languages
* Once you select a model by clicking on it, you’ll see that there is a widget enabling you to try it directly online. This way you can quickly test the model’s capabilities before downloading it.
* All the models can be tested directly through your browser using the Inference API, which is available on the Hugging Face website. You can play with the model directly on this page by inputting custom text and watching the model process the input data.
* The Inference API that powers the widget is also available as a paid product, which comes in handy if you need it for your workflows

# Mask-Filling

* The idea of this task is to fill in the blanks in a given text
* The top\_k  argument controls how many possibilities you want to be displayed.
* Note that here the model fills in the special <mask> word, which is often referred to as a mask token.
* Other mask-filling models might have different mask tokens, so it’s always good to verify the proper mask word when exploring other models.
* One way to check it is by looking at the mask word used in the widget.

# Named-Entity-Recognition

* Named entity recognition (NER) is a task where the model has to find which parts of the input text correspond to entities such as persons, locations, or organizations.
* We pass the option  grouped\_entitities = True in the pipeline creation function to tell the pipeline to regroup together the parts of the sentence that correspond to the same entity: here the model correctly grouped “Hugging” and “Face” as a single organization, even though the name consists of multiple words. In fact, the preprocessing even splits some words into smaller parts. For instance, Sylvain is split into four pieces: s, ##yl, ##va and ##in. In the post-processing step, the pipeline successfully regrouped those pieces.

# Question-Answering

* The question-answering  pipeline answers questions using information from a given context.
* Note that this pipeline works by extracting information from the provided context; it does not generate the answer,

# Summarization

* Summarization is the task of reducing a text into a shorter text while keeping all (or most) of the important aspects referenced in the text

# Translation

* For translation, you can use a default model if you provide a language pair in the task name (such as “translation\_en\_to\_fr”), but the easiest way is to pick the model you want to use on the Model Hub.