**Slide 1: Title Slide**

"Hi everyone, Welcome to Part 4 of our course “Generative AI with the Hugging Face's ecosystem.

In this part, I will guide you through the powerful tools and resources offered by Hugging Face. Let's dive into how we can leverage these tools to build and fine-tune LLMs effectively"

**Slide 2: What We Will Learn**

"In this part of the course, we will cover several key aspects:

First, an introduction to Hugging Face and its ecosystem.

Then, we will explore Hugging Face Hubs including models, datasets, and spaces.

Next, understanding and utilizing Hugging Face libraries such as Transformers and Datasets, Evaluate

Finally, I will introduce hands-on on fine-tuning pre-trained language models.

By the end of this module, you will have a comprehensive understanding of Hugging Face's ecosystem and its practical use-cases."

**Slide 3: Introduction to Hugging Face x NLP**

"To start, we will look at the NLP landscape since the arrival of Transformer models, then understand why Hugging Face stands out, and explore the various components of Hugging Face's ecosystem."

**Slide 4: NLP Since Transformer's Arrival**

Now we talk about the NLP landscape since the arrival of transformer model.

" The Transformer model has significantly impacted NLP, enabling more efficient and effective processing of language data.

Key components of transformers include encoders and decoders. The encoder processes the input text, and the decoder generates the output. This architecture is the foundation of many powerful LLM that we use today.

Since the arrival of the transformer in 2017, many large language models have been built upon the structure of the original transformer. Basically, these models can be classified into three groups:

* LLMs with only the Encoder part, such as BERT, RoBERTa, and ELECTRA.
* LLMs with only the Decoder part, such as GPT, Llama 2, and Mistral.
* LLMs with both the Encoder-Decoder parts, such as T5 and BART.

These models excel in tasks such as text classification, text generation, translation, question answering, named entity recognition, and summarization."

**Slide 5: Why Hugging Face?**

Why Hugging Face ? To answer this question, let’s see how was the NLP development community before Hugging Face.

"Before Hugging Face, the development of transformer models involved non-standardized code and extensive engineering efforts for new use cases.

With Hugging Face, it introduced a standardized interface for a wide range of transformer models, it provided code and tools to easily adapt the models to new use-cases, simplifying the adaptation process and accelerating development."

**Slide 6: Hugging Face's Ecosystem**

Now let’s see what does Hugging Face offer in its ecosystem

Firstly, Hugging Face is a collaboration platform for the machine learning community.

"Hugging Face offers an extensive ecosystem including:

* Libraries like Transformers, Datasets, Tokenizers, Accelerate, etc
* A Hub is developed to facilitate sharing machine learning models, checkpoints and artifacts. At the time of recoding this video, Hugging Face has over 600,000 models, 150,000 datasets, and 50,000 app demos.
* The libraries like Transformers or Datasets can communicate easily with the Hug via APIs that allows to load or share a model or a dataset easily.

**Slide 7: Hugging Face Hubs**

Let’s take a look at Hugging Face Hub;

"The Hugging Face Hub is divided into three main sections:

1. Models: Download and upload open-source models.
2. Datasets: Access and contribute datasets.
3. Spaces: Explore applications built on models and datasets using tools like Streamlit or Gradio."

**Slide 8: Hugging Face Hubs | Models**

Now take a closer look at Models on Hugging Face Hubs

"On the Models Hub, you can find a vast collection of open-source models ready for use or fine-tuning. This resource is invaluable for quickly implementing state-of-the-art NLP solutions.

You can click on one specific NLP task to filter out only models that serve this task.

"

**Slide 9: Hugging Face Hubs | Datasets**

What’s about the Datasets Hubs;

"The Datasets Hub provides access to a wide range of datasets, allowing you to download or contribute datasets for various NLP tasks. This helps standardize data preprocessing and sharing.

Similarly to Models Hub, you can click on one specific NLP task to filter out only datasets that serve this task.

"

**Slide 10: Hugging Face Hubs | Spaces**

Now let’s talk about Spaces Hub.

"The Spaces Hub showcases applications and demos built with models and datasets from Hugging Face or other sources. It provides a platform for sharing and inspiring innovative uses of NLP and LLM models in real-world applications."

**Slide 11: Hugging Face Libraries** "Hugging Face offers many libraries, including:

* Transformers for model implementations.
* Datasets for standardized data handling.
* Evaluate for consistent model evaluation.
* And many more like Accelerate, PEFT, etc

These libraries streamline the process of developing, fine-tuning, and evaluating NLP models."

**Slide 12: Hugging Face Docs** "

To leverage Hugging Face libraries for NLP tasks, the best way is to explore the Hugging Face documentation. When you click on the Docs page on the Hugging Face website, it shows you all the Hugging Face libraries and provides access links to the documentation of each library, including examples, use cases, notebook files, etc. This is the best source to master Hugging Face libraries.

In the scope of this course, I will just show you several major libraries that allow you to build, fine-tune, and evaluate LLMs for text processing.

**Slide 13: Hugging Face Libraries | Transformers**

In the documentation page of Hugging Face Transformers library, you are guided to install the library, how to use this library for various tasks in NLP like text classification, question answering, translation and so on. For each task and use case, we will have example code and explanation how to apply it in the real-world case.

**Slide 14: Hugging Face Libraries | Transformers | Example code-snippet** "

Now let's highlight some key benefits of the Transformers library:

First, the Transformers library provides a simple and standardized interface for a wide range of transformer models.

Second, it offers great adaptability by implementing code and tools to adapt transformer models to new use cases, such as translation or question-answering.

Furthermore, it supports multiple deep learning frameworks like PyTorch, TensorFlow, and JAX, making it highly adaptable.

Last but not least, it is designed to adapt task-specific neural network heads, allowing for easy fine-tuning of transformers on downstream tasks such as text classification, summarization, named entity recognition, or question answering.

Here's a code example showing how to set up a sentiment analysis pipeline using a pre-trained BERT model in just a few lines of code:

First, we load the necessary modules from the Transformers library.

Then, we define the model name, which is available in the Hugging Face Models hub.

After that, using the provided modules, we can load the pre-trained model and its corresponding tokenizer from the Model hub based on the model name.

Next, we set up a classifier with the pipeline module, specifying the task as ‘sentiment-analysis,’ the model, and the tokenizer.

Finally, we can test it by adding a sentence as the argument to the classifier, and we will obtain the label and its probability score.

"

**Slide 15: Datasets** "

Now let’s talk about the Datasets library.

From the documentation, you will learn:

* How to load a dataset from the Hub, local files, a specific slice of a split, etc.
* How to process or stream the dataset.
* How to use it with TensorFlow, PyTorch, JAX, or Spark.
* How to deal with different types of datasets like CSV, JSON, Parquet, Arrow, SQL, and WebDataset.
* And so on.

For example, in just two lines of code, we can load a dataset from the Hub.

So, I highly recommend you check out the documentation page to learn more about this library.

"

**Slide 16: Datasets** "

Next, let's look at some major benefits of the Datasets library:

* The Datasets library offers a standardized interface for thousands of datasets.
* Features like:
  + Smart caching, which eliminates the need to redo the reprocessing each time you run the code.
  + Memory mapping, which stores the contents of a file in virtual memory and enables multiple processes to modify a file more efficiently.
  + Compatibility with popular frameworks like Pandas and NumPy, making data handling efficient and seamless.

Here's a code example to load and explore the 'emotion' dataset:

First, we need to load the function load\_dataset from the datasets library.

Then, it is easy to load a specific dataset by specifying the dataset name, for example, 'emotion,' as the argument of this function.

The function will automatically download the dataset from the Hub.

We can then print out some results to check the dataset.

"

**Slide 17: Evaluate** "

Another important library in the Hugging Face ecosystem is Evaluate.

First, as always, I highly recommend checking out the documentation page of this library. The documentation page will guide you on:

* How to choose the right metric, including generic metrics, task-specific metrics, and dataset-specific metrics.
* How to add new evaluations, using the evaluator, etc.
* How to use the Hugging Face Evaluate library with other ML frameworks like Transformers, Keras and TensorFlow, scikit-learn, etc.

"

**Slide 18: Evaluate** "

Now let’s highlight the major conveniences of the Evaluate library.

First, the Evaluate library allows for consistent and reproducible model evaluation, whether on a local machine or in a distributed training setup.

Next, it provides access to various metrics and tools for comparison and measurement.

Here's a code example showing how to compute metrics using the SQuAD metric:

First, we initiate the metric function by calling the load function from the evaluate library and specifying the metric name, in this case, "squad."

Then, imagine we have a prediction list and a reference list.

Finally, we just need to call the compute method on this metric object, adding predictions and references as arguments. As a result, we obtain the exact\_match score and the F1 score.

**Slide 19: Fine-tune a Pre-trained LM with HF** "

Now let’s talk about one of the coolest capabilities of Hugging Face libraries: fine-tuning pre-trained language models. This process with Hugging Face involves several steps:

First:

* Loading and processing datasets using the Datasets library.
* Loading a pre-trained model from the Transformer Hub and tokenizing input texts using the Transformers library.

Next, we can set up the trainer and evaluation during training using the Transformers library.

Then, we use the Evaluate library to evaluate the trained model. If the evaluation score is not satisfactory, we should return to the training step to tune the training parameters. If the score is satisfactory, we have obtained the final fine-tuned transformer model.

Depending on how we choose the dataset and set up the task-specific neural network head during training, the obtained fine-tuned models can handle different downstream tasks in NLP, such as text classification, translation, question-answering, etc.

In summary, this process enables you to customize models for specific NLP tasks efficiently.

."

**Slide 20: Fine-tune a Pre-trained LM with HF** "

Now let’s not reinvent the wheel; we will look at the documentation of the Transformers library to see how it guides us through the task of fine-tuning a pre-trained model.

You will learn:

* How to prepare a dataset.
* How to train with PyTorch Trainer, TensorFlow with Keras, or in native PyTorch, etc.

Let’s check it out.

**Slide 23: End-to-End Fine-tuned Example** "

Now, let's look at a practical example.

Our use-case is that

* We'll fine-tune a pre-trained model for spam classification in emails using the 'distilbert-base-uncased' model and the 'tanquangduong/spam-detection-dataset-splits' dataset.
* After fine-tuning, we'll push the model to the Hub for others to use."

Now, let’s get our hands dirty with the implementation of fine-tuning a BERT model for text classification on spam detection.

First, we import the necessary libraries and modules, such as transformers, datasets, and evaluate.

Then, we use the datasets library to load and process the dataset. This involves loading the spam detection dataset into a suitable format for training. We can print out some results of the loaded dataset to verify its contents.

Next, we apply several EDA techniques to verify the consistency of the dataset. First, we need to transform the dataset into a DataFrame object. Then, we write a function to convert binary labels to label names and add a label\_name feature to the DataFrame. After that, we can plot the number of label names for each class to evaluate the distribution of spam and not\_spam classes. We can see that we have a balanced dataset.

Moreover, we can also calculate the text length of the emails and compare the statistical information regarding email length for spam and not\_spam emails. We can see that the median length for spam and not\_spam emails is quite similar, around 45 words.

After the EDA step, we need to reset the dataset format to be compatible with the transformers framework for training.

Next, using the pretrained model name, AutoTokenizer, and AutoModelForSequenceClassification class, we load the tokenizer and the model from the Model Hub.

Then, we write a function to tokenize the text batch, encoding text to numerical representations.

After that, we set up the Trainer by defining the training arguments, compute\_metrics function, and the Trainer class.

Afterwards, we train the model using the train method of the Trainer. During the training process, we can see the log for each epoch, including training loss, validation loss, accuracy score, and F1 score. We can see that we obtained a very good score.

Let’s move on to the inference step. In this step, we will run the trained model over the test subset of the dataset to obtain the predictions.

The next step is evaluation. We will compare the predictions and the reference over the test subset using the confusion matrix. Great, we can see that our fine-tuned model obtains very good performance in the test phase.

Well, the last step should be “push to the hub,” which allows us to push the trained model to the Hugging Face Model Hub for community use.

In summary, this notebook walks through the entire process of fine-tuning a BERT model for a specific NLP task, from data preparation to model evaluation and sharing.

**Slide 21: Share Your Model** "

One of the great initiatives of Hugging Face is open-source sharing to accelerate AI community development. Hugging Face makes it easy to share your fine-tuned models with the community. With a single line of code, you can push your models to the Hub, enabling others to use and build upon your work, fostering collaboration and innovation.."

**Slide 24: In Summary** "

To wrap up, we covered the Hugging Face ecosystem, including Hubs and Libraries, and learned how to get the most from Hugging Face’s documentation.

We learned how to fine-tune pre-trained models and explored an end-to-end example, and learned how to share your model with the community.

Leveraging Hugging Face's ecosystem empowers you to efficiently build and deploy advanced NLP solutions.