Deployment with AWS Sagemaker:

Deploying from Huggingface’s Model Garden:

Deploying models in SageMaker provides you with production-ready endpoints that scale easily within your AWS environment. In order to deploy we simply create an instance of the HuggingFaceModel`(...)` class and call the deploy method off it.

We use the new HuggingFace Inference Toolkit for Amazon SageMaker. This new Inference Toolkit leverages the pipelines from the transformers library to allow zero-code deployments of models without writing any code for pre- or post-processing,

Deploying PyTorch Models:

All the models that were trained with pytorch and the best weights are restored as .pth files.

For deploying the models with aws sagemaker we create a notebook instance of sagemaker and upload the weights file, the vocabulary to encode raw sentences to id’s and an inference.py file for defining the model creation, input preprocessing, prediction generation and output preprocessing.

Next create a notebook(deploy.ipynb), tarball the weights file and upload it to a S3 bucket.
Next create a PyTorchModel object with paths to the weights file and the entry point file i.e, inference.py in S3 bucket.
Now the model can be deployed using the deploy method of the PyTorchModel object.
Once successfully deployed the model will live in an endpoint in the cloud and can be invoked to receive requests to make predictions

These requests will be handled as defined by the functions in the inference.py file.

The inference.py file has following functions:

1) To tell the inference image how to load the model checkpoint,we implement a function called `model_fn`. This function takes one positional argument
    - `model_dir`: the directory of the static model checkpoints in the inference image.

Next, you need to tell the hosting service how to handle the incoming data. This includes:

- How to parse the incoming request
- How to use the trained model to make inference
- How to return the prediction to the caller of the service

2) **`input_fn`** function

The SageMaker PyTorch model server will invoke an `input_fn` function in your inference entry point. This function handles data decoding. The `input_fn` have the following signature:

```
def input_fn(request_body, request_content_type)
```

The two positional arguments are: - `request_body`: the payload of the incoming request - `request_content_type`: the content type of the incoming request

3) **`predict_fn`**

After the inference request has been deserialized by `input_fn`, the SageMaker PyTorch model server invokes `predict_fn` on the return value of `input_fn`.

The `predict_fn` function has the following signature:

```
def predict_fn(input_object, model)
```

The two positional arguments are: - `input_object`: the return value from `input_fn` - `model`: the return value from `model_fn`

The return of `predict_fn` is the first argument to be passed to `output_fn`

`4)` **`output_fn`**

After invoking `predict_fn`, the model server invokes `output_fn` for data post-process. The `output_fn` has the following signature:

```
def output_fn(prediction, content_type)
```

The two positional arguments are: - `prediction`: the return value from `predict_fn` - `content_type`: the content type of the response

The return of `output_fn` should be a byte array of data serialized to `content_type`.