Assignment 1: Submit a write-up on the following:

- Hugging face agents

An Agent is an LLM assistant that performs VERY SPECIFIC actions. It could be image generation, it could be sentence completion, or code generation (and execution of that code
Agents
In this case, we are using an agent to call upon an image generation based on our request

two types of agents, based on the main Agent class:

- CodeAgent acts in one shot, generating code to solve the task, then executes it at once.
- ReactAgent acts step by step, each step consisting of one thought, then one tool call and execution. It has two classes:
  - ReactJsonAgent writes its tool calls in JSON.
  - ReactCodeAgent writes its tool calls in Python code.

https://huggingface.co/docs/transformers/main_classes/agent#transformers.Agent

https://huggingface.co/docs/transformers/main_classes/agent#transformers.Agent.execute_tool_call

- Hugging face pipeline for text generation

  The pipeline performs the following operations, all in two line:
  1. Model download
  2. Parameter setting
  3. Query tokenization
  4. Query answering

  In other words, you can answer questions very simply. This is the closest 'Chat GPT'-esque behaviour that is present in this code.
  In other words, you can answer questions very simply. This is the closest 'Chat GPT'-esque behaviour that is present in this code

  Generation
  Each framework has a generate method for text generation implemented in their respective GenerationMixin class:

- PyTorch generate() is implemented in GenerationMixin.
- TensorFlow generate() is implemented in TFGenerationMixin.
- Flax/JAX generate() is implemented in FlaxGenerationMixin.

  Text generation strategies

Text generation is essential to many NLP tasks, such as open-ended text generation, summarization, translation, and more. It also plays a role in a variety of mixed-modality applications that have text as an output like speech-to-text and vision-to-text. Some of the models that can generate text include GPT2, XLNet, OpenAI GPT, CTRL, TransformerXL, XLM, Bart, T5, GIT, Whisper.

few examples that use generate() method to produce text outputs for different tasks:

- Text summarization
- Image captioning
- Audio transcription

TextGenerationPipeline
class transformers.**TextGenerationPipeline**
(*args**kwargs)
Parameters

- **model** (PreTrainedModel or TFPreTrainedModel) — The model that will be used by the pipeline to make predictions. This needs to be a model inheriting from PreTrainedModel for PyTorch and TFPreTrainedModel for TensorFlow.
- **tokenizer** (PreTrainedTokenizer) — The tokenizer that will be used by the pipeline to encode data for the model. This object inherits from PreTrainedTokenizer.
- **modelcard** (str or ModelCard, *optional*) — Model card attributed to the model for this pipeline.
- **framework** (str, *optional*) — The framework to use, either "pt" for PyTorch or "tf" for TensorFlow. The specified framework must be installed.
  If no framework is specified, will default to the one currently installed. If no framework is specified and both frameworks are installed, will default to the framework of the model, or to PyTorch if no model is provided.
- **task** (str, defaults to "") — A task-identifier for the pipeline.
- **num_workers** (int, *optional*, defaults to 8) — When the pipeline will use *DataLoader* (when passing a dataset, on GPU for a Pytorch model), the number of workers to be used.
- **batch_size** (int, *optional*, defaults to 1) — When the pipeline will use *DataLoader* (when passing a dataset, on GPU for a Pytorch model), the size of the batch to use, for inference this is not always beneficial, please read Batching with pipelines .
- **args_parser** (ArgumentHandler, *optional*) — Reference to the object in charge of parsing supplied pipeline parameters.
- **device** (int, *optional*, defaults to -1) — Device ordinal for CPU/GPU supports. Setting this to -1 will leverage CPU, a positive will run the model on the associated CUDA device id. You can pass native torch.device or a str too

- **torch_dtype** (str or torch.dtype, *optional*) — Sent directly as model_kwargs (just a simpler shortcut) to use the available precision for this model (torch.float16, torch.bfloat16, … or "auto")
- **binary_output** (bool, *optional*, defaults to False) — Flag indicating if the output the pipeline should happen in a serialized format (i.e., pickle) or as the raw output data e.g. text.

  Language generation pipeline using any ModelWithLMHead. This pipeline predicts the words that will follow a specified text prompt. When the underlying model is a conversational model, it can also accept one or more chats, in which case the pipeline will operate in chat mode and will continue the chat(s) by adding its response(s). Each chat takes the form of a list of dicts, where each dict contains “role” and “content” keys.

- HF inference endpoints
- Inference Endpoints
- Inference Endpoints offers a secure production solution to easily deploy any Transformers, Sentence-Transformers and Diffusers models from the Hub on dedicated and autoscaling infrastructure managed by Hugging Face.
- A Hugging Face Endpoint is built from a Hugging Face Model Repository. When an Endpoint is created, the service creates image artifacts that are either built from the model you select or a custom-provided container image. The image artifacts are completely decoupled from the Hugging Face Hub source repositories to ensure the highest security and reliability levels.
- Inference Endpoints support all of the Transformers, Sentence-Transformers and Diffusers tasks as well as custom tasks not supported by Transformers yet like speaker diarization and diffusion.
- In addition, Inference Endpoints gives you the option to use a custom container image managed on an external service, for instance, Docker Hub, AWS ECR, Azure ACR, or Google GCR

Documentation and Examples

- Security & Compliance
- Supported Transformers Task

- API Reference
- Autoscaling
- FAQ
- Help & Support

Guides

- Access the solution (UI)
- Create your first Endpoint

- Give feedback on the image generation and explore different models available on the Hugging Face website

impressive and provide a powerful set of tools for creating a wide variety of visual content from text prompts. The platform offers access to some of the leading text-to-image models, such as Stable Diffusion and DALLE-2, which have demonstrated remarkable abilities to generate high-quality, photorealistic images.

Raghad alqirnas