GPT-3

Email 받은 날: 2021.01.16 --- 사용 분야를 자세하게 적은 게 도움이 된 듯 (Korean, poem, Children's song 등)

# OpenAI - 계정

heyhmin@ewhain.net

capstone12

# URL:

<https://beta.openai.com/dashboard?supportSignUp=true&supportForgotPassword=true&email=heyhmin%40ewhain.net&message=Your%20email%20was%20verified.%20You%20can%20continue%20using%20the%20application.&success=true&code=success#>

# Export Code

import os

import openai

openai.api\_key = os.environ["OPENAI\_API\_KEY"]

response = openai.Completion.create(

engine="davinci",

prompt="노래\n\n우리",

temperature=0.5,

max\_tokens=64,

top\_p=1,

frequency\_penalty=0.2

)

# **Your API Key**

This is your **Secret API Key**. Do not share this key with others, or expose it in the browser or other client-side code. \*\*\* 공유 금지!!! 딥공댕에서만 사용!!!

(숨김 - 홈페이지에서 확인 가능) sk-VhCt4227WyTQFU7GkevCkZ4aXCcdoOVEplNHEvHd

## **Python Bindings**

$ pip install openai

import openai

openai.api\_key = "sk-VhCt4227WyTQFU7GkevCkZ4aXCcdoOVEplNHEvHd"

response = openai.Completion.create(engine="davinci", prompt="This is a test", max\_tokens=5)

#OR

$ openai api completions.create -e davinci -p "This is a test" -M 5 --stream

# Introduction

texts in, texts out의 형식

하고자 하는 것에 대한 설명 혹은 몇 가지 예제를 적어서 결과물을 바꿀 수 있다.

**prompt,** **completion**, and **tokens**

prompt: 입력, 최대 2048 tokens (1500 단어)

completion: 생성된 출력, 최대 2048 tokens (1500 단어)

\*\* 확률적이기 때문에 생성할 때마다 결과물이 달라질 수 있음

tokens: 말뭉치 조각 - 단어와 비슷함, 많은 토큰이 공백으로 시작함 “ hello”

# GPT-3 models

[ davinci, curie, babbage, and ada ]

davinci : 가장 유능한 모델, 권장함

ada : 가장 빠른 모델

\*\* 관련 논문 - GPT3 : <https://arxiv.org/abs/2005.14165>

\*\* GPT3 모델 성능 비교 : <https://gpttools.com/comparisontool>

\*\*\*\*\*\* 우선은 gpt3 모델 성능 비교로 간단하게 실행 가능

API가 사용자가 무엇을 원하는지 잘 알 수 있도록 Prompt를 작성해야 함.

# Prompts guidelines

1. 지시사항과 예제를 제시할 것.
2. 패턴을 따르거나 분류하는 것을 원할 경우 양질의 데이터를 제공할 것
3. settings를 확인할 것 - temperature & top\_p

Prompts checklist

1. Is it clear what the intended generation should be?
2. Are there enough examples?
3. Did you check your examples for mistakes? (The API won’t tell you directly)
4. Are you using temp and top\_p correctly?

Generation example

prompt:

Ideas involving education and virtual reality

1. Virtual Mars

Students get to explore Mars via virtual reality and go on missions to collect and catalog what they see.

prompt의 끝이 ‘ ‘ 공백이면 안 됨.

### -> prompt 동요 예시

**아이들이 좋아하는 밝은 분위기의 노래 가사**

**1. 파란 하늘 파란 하늘 꿈이 드리운 푸른 언덕에 아기 염소 여럿이 풀을 뜯고 놀아요 해처럼 밝은 얼굴로**

**2. 우리들 마음에 빛이 있다면 여름엔 여름엔 파랄거예요 산도 들도 나무도 파란잎으로 파랗게 파랗게 덮힌 속에서 파아란 마음으로 자라니까요**

**3.**

<-

: 목록 설명과 하나의 예시를 API에 제공한다.

: 목록의 계속임을 나타내는 숫자 2를 표시한다.

: 적절한 출력을 만나면 유지하고, 아니면 re-generate한다.

# **Engine comparisons**

### **Davinci**

Davinci is the most capable engine and can perform any task the other models can perform and often with less instruction. For applications requiring a lot of understanding of the content, like summarization for a specific audience and content creative generation, Davinci is going to produce the best results. The trade-off with Davinci is that it costs more to use per API call and other engines are faster.

Another area where Davinci shines is in understanding the intent of text. Davinci is quite good at solving many kinds of logic problems and explaining the motives of characters. Davinci has been able to solve some of the most challenging AI problems involving cause and effect.

Good at: **Complex intent, cause and effect, summarization for audience**

### **Curie**

Curie is extremely powerful, yet very fast. While Davinci is stronger when it comes to analyzing complicated text, Curie is quite capable for many nuanced tasks like sentiment classification and summarization. Curie is also quite good at answering questions and performing Q&A and as a general service chatbot.

Good at: **Language translation, complex classification, text sentiment, summarization**

### **Babbage**

Babbage can perform straightforward tasks like simple classification. It’s also quite capable when it comes to Semantic Search ranking how well documents match up with search queries.

Good at: **Moderate classification, semantic search classification**

### **Ada**

Ada is usually the fastest model and can perform tasks like parsing text, address correction and certain kinds of classification tasks that don’t require too much nuance. Ada’s performance can often be improved by providing more context.

Good at: **Parsing text, simple classification, address correction, keywords**

*Note: Any task performed by a faster model like Ada can be performed by a more powerful model like Curie or Davinci.*

# **The Instruct Series**