



Министерство науки и высшего образования Российской Федерации
Калужский филиал
федерального государственного бюджетного
образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИУК «Информатика и управление»

КАФЕДРА ИУК4 «Программное обеспечение ЭВМ,

информационные технологии»

Лабораторная работа №4

«Технологии машинного обучения в облачной системе Yandex Cloud»

ДИСЦИПЛИНА: «Облачные технологии»

Выполнил: студент гр. ИУК4-82Б _____ (Сафронов Н.С.)
(подпись) (Ф.И.О.)

Проверил: _____ (Амеличев Г.Э.)
(подпись) (Ф.И.О.)

Дата сдачи (защиты):

Результаты сдачи (защиты):

- Балльная оценка:

- Оценка:

Калуга, 2024

Цель работы: приобретение практических навыков по работе с сервисом Yandex Cloud.

Постановка задачи

Решите задачу согласно варианту с использованием платформы Yandex Cloud.

Вариант 4

С помощью сервиса Yandex Cloud Translate перевести на русский язык машинно-сгенерированный текст на английском языке.

Результаты выполнения работы

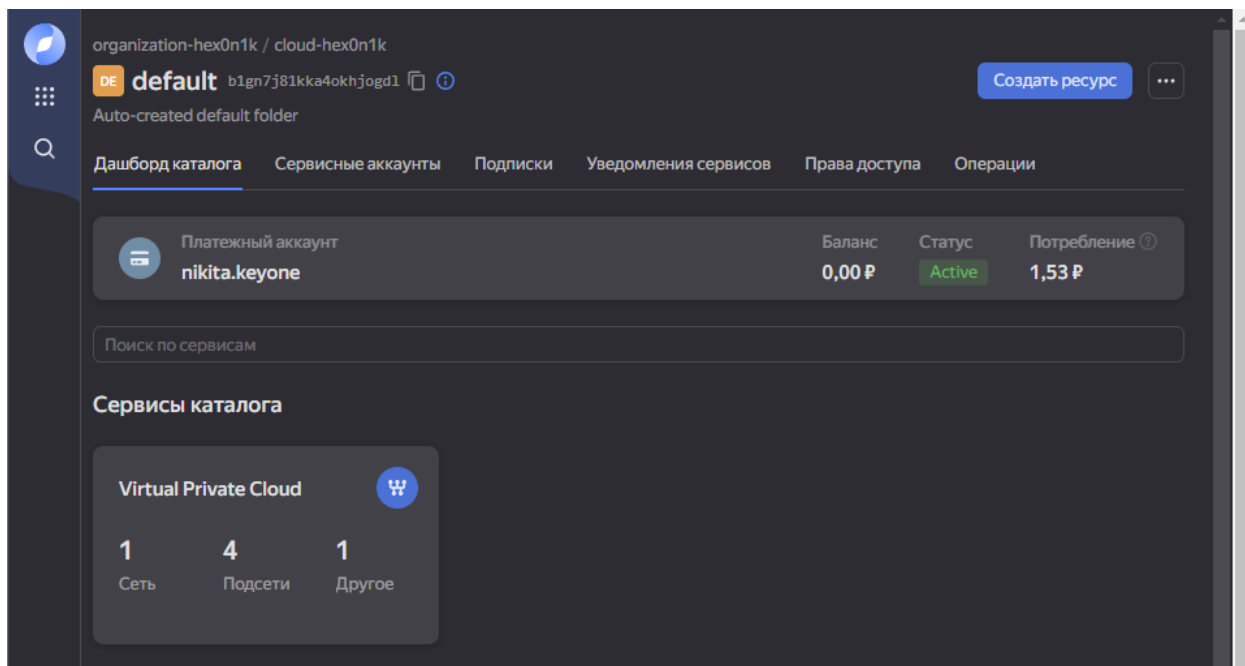


Рисунок 1 – Созданная учётная запись

```

k1@keyone-laptop:~/yandex-cloud$ yc init
Welcome! This command will take you through the configuration process.
Please go to https://oauth.yandex.ru/authorize?response_type=token&client_id=1a6990aa636648e9b2ef855fa7bec2fb in order to obtain OAuth token.

Please enter OAuth token: y0_AgAAAAATSThRAATuwQAAAAECZG5MAABELgUeXtFP3ZJ9sQB8HcdRECE3wg
e 2010-2024 000 «Яндекс»
0 компанийYou have one cloud available: 'cloud-hex0nik' (id = big314qqlhred5lp19v). It is going to be used by default.
Please choose folder to use:
[1] default (id = bign7j8ikka4okhjogdl)
[2] Create a new folder
Please enter your numeric choice: Please enter a value between 1 and 2: 1
Please enter a value between 1 and 2: 1
Your current folder has been set to 'default' (id = bign7j8ikka4okhjogdl).
Do you want to configure a default Compute zone? [Y/n] Y
Which zone do you want to use as a profile default?
[1] ru-central1-a
[2] ru-central1-b
[3] ru-central1-c
[4] ru-central1-d
[5] Don't set default zone
Please enter your numeric choice: 1
Your profile default Compute zone has been set to 'ru-central1-a'.
k1@keyone-laptop:~/yandex-cloud$ yc config list
token: y0_AgAAAAATSThRAATuwQAAAAECZG5MAABELgUeXtFP3ZJ9sQB8HcdRECE3wg
cloud-id: big314qqlhred5lp19v
folder-id: bign7j8ikka4okhjogdl
compute-default-zone: ru-central1-a
k1@keyone-laptop:~/yandex-cloud$

```

Рисунок 2 – Создание профиля

Листинг программы

Файл main.py:

```

from wrappers.cli import CliWrapper
from services.translate import TranslationService
from services.gpt import GptService
import sys

if __name__ == '__main__':
    cli_wrapper = CliWrapper()
    config = cli_wrapper.parse_config(sys.argv[1:])

    gpt_service = GptService(
        token=config['token'],
        folder_id=config['folder_id'],
        api_url=config['gpt_api_url']
    )

    print(f'[GPT] Started generation for prompt
{config["prompt"]}...')
    text = gpt_service.get_generated_text(config['prompt'])
    print('[GPT] Text was successfully generated:', text)

    translations_service = TranslationService(
        token=config['token'],
        target_language=config['target_language'],
        folder_id=config['folder_id'],
        api_url=config['translations_api_url']
    )

    print(
        f'[Translation] Started translation to
{config["target_language"]}',
        'for generated text...'
    )
    translation = translations_service.get_translation(text)

```

```
print('[Translation] Text was successfully translated:',
translation)
```

Файл models/base.py:

```
import typing as t
```

```
ModelType = t.TypeVar('ModelType')
```

```
def load_model(cls: t.Type[ModelType], data: t.Mapping) ->
ModelType:
    required = cls.__dict__.get('__required_keys__',
frozenset())
    optional = cls.__dict__.get('__optional_keys__',
frozenset())

    model = cls(
        **{key: data[key] for key in required},
        **{key: data[key] for key in optional if key in data},
    )

    return model
```

Файл models/config.py:

```
"""."""
```

```
import os
```

```
import typing as t
```

```
class Config(t.TypedDict):
```

```
    """Конфигурация для сервиса перевода Yandex Cloud"""
```

```
    token: str
```

```
    folder_id: str
```

```
    target_language: str
```

```
    translations_api_url: str
```

```
    gpt_api_url: str
```

```
    prompt: str
```

```
default_config = Config(
```

```
    token=os.environ.get('YC_TOKEN', ''),
```

```
    folder_id=os.environ.get('YC_FOLDER_ID', ''),
```

```
    target_language=os.environ.get('YC_FOLDER_ID', ''),
```

```
    translations_api_url='https://translate.api.cloud.yandex.net/tra
nslate/v2/'
```

```
        'translate',
```

```
    gpt_api_url='https://llm.api.cloud.yandex.net/foundationModels/v
1/'
```

```
        'completion',
```

```
    prompt=os.environ.get('YC_GPT_PROMPT', ''),
```

```
)
```

Файл models/translation.py:
"""
import typing as t

```
class Translation(t.TypedDict):  
    """  
    text: str  
    detectedLanguageCode: str
```

Файл services/gpt.py:

```
import requests  
import logging  
from models.translation import Translation  
from models.base import load_model
```

```
class GptService:  
    """  
    def __init__(  
        self,  
        token: str,  
        folder_id: str,  
        api_url: str,  
    ):  
        self._token = token  
        self._folder_id = folder_id  
        self._api_url = api_url  
        self._logger = logging.Logger('GPT')  
  
    def _request_generation(self, prompt: str) -> dict:  
        body = {  
            "modelUri": f"gpt://{self._folder_id}/yandexgpt-  
lite/latest",  
            "completionOptions": {  
                "maxTokens": 500,  
            },  
            "messages": [  
                {  
                    "role": "user",  
                    "text": prompt,  
                },  
            ]  
        }  
  
        headers = {  
            'Content-Type': 'application/json',  
            'Authorization': f'Bearer {self._token}',  
        }  
  
        response = requests.post(  
            url=self._api_url,
```

```

        json=body,
        headers=headers,
        timeout=5,
    )

    if not response.ok:
        self._logger.error(
            f'Unable to generate text,
code={response.status_code}'
        )
        raise RuntimeError('Unable to generate')

    return response.json()

def get_generated_text(self, prompt: str) -> dict:
    response = self._request_generation(prompt)
    single_result =
response.get('result').get('alternatives')[0]

    return single_result.get('message').get('text')

```

Файл services/translate.py:

```

import requests
import logging
from models.translation import Translation
from models.base import load_model

class TranslationService:
    """."""
    def __init__(
        self,
        token: str,
        folder_id: str,
        target_language: str,
        api_url: str,
    ):
        self._token = token
        self._folder_id = folder_id
        self._target_language = target_language
        self._api_url = api_url
        self._logger = logging.Logger('Translate')

    def _request_translate(self, text: str) -> dict:
        body = {
            'targetLanguageCode': self._target_language,
            'folderId': self._folder_id,
            'texts': text,
        }

        headers = {
            'Content-Type': 'application/json',
            'Authorization': f'Bearer {self._token}',

```

```

    }

    response = requests.post(
        url=self._api_url,
        json=body,
        headers=headers,
        timeout=5,
    )

    if not response.ok:
        self._logger.error(
            f'Unable to generate text,
code={response.status_code}'
        )
        raise RuntimeError('Unable to translate')

    return response.json()

    def get_translations(self, text: str) -> list[Translation]:
        raw_translations =
self._request_translate(text).get('translations')

        return [load_model(Translation, x) for x in
raw_translations]

    def get_translation(self, text: str) -> Translation:
        return self.get_translations(text)[0].get('text')

```

Файл wrappers/cli.py:

```

import argparse
import copy
from models.config import default_config, Config

class CliWrapper:
    def __init_parser(self) -> argparse.ArgumentParser:
        parser = argparse.ArgumentParser(
            description='Yandex Cloud Translations API'
        )

        parser.add_argument('prompt', type=str)
        parser.add_argument('--token', type=str, required=False)
        parser.add_argument('--folder_id', type=str,
required=False)
        parser.add_argument('--target_language', type=str,
required=False)

        return parser

    def __init__(self):
        self._parser = self._init_parser()

    def parse_config(self, data: str | list[str]) -> Config:

```

```

raw_config = vars(self._parser.parse_args(data))

config = copy.deepcopy(default_config)

if token := raw_config.get('token'):
    config['token'] = token

if folder_id := raw_config.get('folder_id'):
    config['folder_id'] = folder_id

if target_language := raw_config.get('target_language'):
    config['target_language'] = target_language

config['prompt'] = raw_config.get('prompt')

return config

```

Результат выполнения программы

```

k1@keyone-laptop: ~/Documents/studies/8th-term/bmst...
k1@keyone-laptop:~/Documents/studies/8th-term/bmstu-8th-term/cloud-technologies/lab4/src$ python3 main.py --folder_id b1gn7j81kka4okhjogdl --token t1.9euelZrKxs-ezpmJmMubzJaYlZeLne3rnpWax50UkpGZzI6UyZqWy4-cnpLl8_c5T3p0-e9nMFJ7_N3z93l9d07572cwUnv8zef1656VmszHlMaLkJTHi57Pl8j0LJfH7_zF656VmszHlMaLkJTHi57Pl8j0LJfH.NeWdR4REF8z27PKB0VNDQtsXXtjVc3dqmrwfabzvYq-B-aazU2dc1LLVjjeJDJcLipuFJt_0JuhQiiR8mnbqDA --target_language ru "What language model you are?"
[GPT] Started generation for prompt What language model you are?...
[GPT] Text was successfully generated: I am an AI language model trained by Yandex AI. I am able to process natural language text and generate human-like text in response. I can be used for a variety of tasks such as answering questions, summarizing text, generating text for creative writing, and providing information on a wide range of topics.
[Translation] Started translation to ru for generated text...
[Translation] Text was successfully translated: Я - языковая модель искусственного интеллекта, обученная в Yandex AI. Я умею обрабатывать текст на естественном языке и генерировать в ответ текст, похожий на человеческий. Я могу быть использован для решения различных задач, таких как ответы на вопросы, обобщение текста, создание текста для творческого письма и предоставление информации по широкому кругу тем.
k1@keyone-laptop:~/Documents/studies/8th-term/bmstu-8th-term/cloud-technologies/lab4/src$

```

Рисунок 3 – Результат выполнения программы

Вывод: в ходе выполнения лабораторной работы были получены практические навыки по работе с сервисом Yandex Cloud.