

Рабочий лист №1

Дата "01" февраля 2025 г.
(заполняется оргкомитетом)

Шифр ПИ-20
(заполняется оргкомитетом)

Оценка работы

(таблица заполняется по итогам проверки работы членами жюри олимпиады)

№ задания	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Итого (итоговый балл, подпись председателя жюри)
Балл																80
№ задания	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	
Балл																15

Магистратура

(название олимпиады, заполняется участником)

Прикладная информатика

(профиль олимпиады, заполняется участником)

Задача 1.

Программная система должна состоять из следующих программных модулей:

1) Сервис ответственный за выгрузку записей телефонных разговоров из БД, генерацию расшифровок и загрузку расшифровок в БД.

2) База данных для генерации расшифровок потребует использование не машинного обучения.

Используемые технологии: Python, Pytorch, FastAPI

Важно это, сервис должен представлять из себя Web API для того, чтобы по запросу отдавать записи телефонных разговоров и тексты расшифровок.

2) База данных где хранение тел. разговоров и их расшифровок. Структура БД.

а) Таблица хранящая номера телефонов, участвующих в каждом звонке ($id_звонка_id$, номер-телефона-звонящего, номер-телефона-отвечающего)

б) Таблица хранящая записи ($id_звонка_id$, путь до файла-записи, путь до файла-расшифровки)

в) ~~Таблица~~
Используемая БД: SQLite

3) Модуль, ответственный за анализ текстовой информации, который ^{собирается с сервера и т.п.} вызывает критическую персональную информацию упомянутую в разговоре и загружает полученный результат в БД. ~~Язык~~ ^{Там же} также представляет собою WebAPI, ^{который предоставляет информацию из БД}

4) База данных для хранения критической информации в зашифрованном виде.
~~Информация~~ База данных MongoDB, информация хранится в виде id ; критическая информация в JSON формате.

5) Сервис, ответственный за взаимодействие с операторами мобильной связи.
Основная задача - получать от операторов мобильной связи информацию о номерах, о частоте и регулярности совершаемых звонков и т.п. для выявления вероятности того, что номер использует диссидентские или террористические звонки; хранение полученных данных в БД. Также является WebAPI, ^{предоставляющий информацию из БД.}

6) БД для хранения данных, полученных от оператора м.с., и $\&$ вычисленной вероятности того, что номер - террористический.
База данных: ~~Postgre~~ SQLite

Дополнительный рабочий лист
(без рабочего листа №1 недействителен)

Дата "01" февраля 2025 г.
(заполняется участником)

Шифр ПЧ-20
(заполняется участником)

7) Сервис, обращающийся к сервисам из пунктов 3 и 5 и на основе полученных данных вызывает звонок является ли разговор мошенническим.

В пунктах, где не указаны технологии, можно использовать любой язык программирования под который существуют библиотеки для & создания WebAPI для работы с REST.

Задача 2

Алгоритм может использовать код из Задачи 3 для получения критической информации, zároveň этот модуль CritInfo

Псевдокод

импорт CritInfo

```
class int Main()
{
    int thresholdScore = int.Parse(input());
    var file Paths = openfile("...");
    Dictionary<string, object Dict(String, int)> critInfo = CritInfo.getInfo(filePaths);
    string[] triggers = ["намеренное удаление", "CVV", "номер карты",
        "фрис", "адрес", "сезонный счет", "номер телефона", "звонки"];
    int[] m = [
        intScore = 0;
        for (i = 0, i < triggers.Length; i++) {
            if (triggers[i].ContainsKey(triggers[i])) {
                if (critInfo[triggers[i]].Item1 != null && !critInfo[triggers[i]].Item2) {
                    score += m[i] * critInfo[triggers[i]].Item1;
                } else {
                    score += m[i] * 0.5;
                }
            }
        }
    }
}
```

```

}
if score > threshold score
    output ("Значок монументальный")
else
    output ("Значок не монументальный")

```

Задача 3

Программа: текст & huge quantity. Set pairs of patterns, using System.Text.RegularExpressions, class Program

```

class Program
{
    public static Main()
    {
        Dictionary<string, (string, int)> result = new();
        var filePath = "...";
        using var fileReader = new FileReader (filePath);
        var text = fileReader file.ReadAllText();
        var words = text.Split(['\n', ' ']);
        List<string> patterns = [ ... ]; // patterns are ranges
        List<string> labels = [ ... ]; // range - code
        int lastJ = -1;
        for (int i = 0; i < words.Length; i++)
        {
            if (words[i] for (var word pattern in patterns)
                for (int j = 0; j < patterns.Length; j++)
                    if (Regex.IsMatch(words[i], patterns[j])))
            {
                if (result.ContainsKey(labels[j]))
                {
                    result[labels[j]] = (result[labels[j]].Item1,
                                         result[labels[j]].Item2 + 1);
                }
                else
                    result.Add((" ", 1));
                lastJ = j;
            }
            if (Regex.IsMatch(words[i], "[0-9]+"))
                result[labels[j]] = (result[labels[j]].Item1,
                                     result[labels[j]].Item2);
        }
        Console.WriteLine(string.Format("foreach (var pair in result)
                                         Console.WriteLine(pair.Key, pair.Value)"));
    }
}

```