Бриф от UNIGOTO

# Задание

## 1. Краткое описание проекта/краткая вводная к проекту

Мы создаём сервис для упрощения процесса выбора места для обучения будущих студентов. Начнём с ВУЗов.

Перед вами стоит задача реализовать ML-решение, способное сформировать персональные ранжированные списки вузов для каждого уникального запроса, состоящего из таких данных, как музыка, фильмы, книги, игры или хобби.

## 2. Формулировка задания

​На вводе — информация о своих интересах, включая фильмы, музыку, книги, игры, хобби и т. д.

Модель выдаёт ранжированный список ВУЗов, встречающихся у наибольшего количества профилей таблицы, у которых пересекаются значения по введенным интересам.

Следует предусмотреть возможность сделать ранжирование по факультетам или специальностям.

## 3. Описать, как действовать при решении задачи

Как это реализовали раньше:

1. ​Входные интересы пользователя очищаются от лишних бессмысленных слов, знаков препинания и т. д.
2. Далее эти интересы преобразуются в вектор, состоящий из 1 и 0 фиксированной длины, которая указывается в коде вручную.

Длина вектора соответствует количеству интересов, рассматриваемых в данном случае. Например, 0-ой элемент вектора — это интерес «Музыка». Если в входном списке интересов, который передал нам пользователь, есть слово «музыка», то 0-ой элемент нашего вектора будет равен 1. Если слово «музыка» отсутствует, то 0-ой элемент в нашем векторе будет равен 0.

1. Далее методом knn находятся ближайшие соседи к сформированному вектору.
2. Все интересы, вузы и факультеты имеют свой id, который был создан на этапе создания модели. После нахождения рекомендаций происходит перевод id в имена вузов и факультетов.

Помимо прочего имеется таблица с разными вариантами названий одних и тех же ВУЗов (аббревиатуры, старые названия, и т. д.) для того, чтобы можно было объединять результаты по повторяющимся вузам. К таблице CSV будет предоставлен доступ для скачивания.

## 4. Все примечания и подсказки

Пример:

Запрос — [{Программирование на фортан},{djaz,soul},{терминатор},{вебер, страустрап, тоненбаум},{косынка},{футбол},{спб},{3}]

Ответ — [{32,9,2}, {12,8,3}, {2,7,3}]

Входные данные для обучения, таблица уникальный id записи, интересы (разделённые, как выше кино и т. д.), институт, кафедра, факультет. Объем данных большой, требуется сортировка, отброс мусорных записей.

Задача со звездочкой – указание строки ввода города в начале и предоставление отдельного ранжированного списка с ограничением мест обучения по городу или региону.

# Требования

## 1. Функциональные требования

В проекте должно быть реализовано:

1. Ввод — несколько строк, вывод — ранжированный список.
2. Вывод топ-20 результатов вузов, факультетов и специальностей с наибольшим количеством анкет с аналогичными значениями в формате списка.
3. Предусмотрена возможность выдачи результатов по вузам, факультетам и специальностям по отдельности, а также с ограничением по городу или региону. Должно указываться в начале запроса до начала сравнения.

## 2. Технические требования

На основе больших данных, переданных заказчиком в виде API, произведите обработку данных (убрать пустые строки, шумы), а также разработайте самостоятельную рекомендательную модель и документацию. Документация должна содержать информацию по обработке первоначальных данных, обучению, сборке, использованию и переобучению.

Плюсом будет реализация в виде Docker-контейнера и реализация взаимодействия в виде API, предпочтительнее в виде GRPC с описанием схемы взаимодействия.

Для более подробного описания смотрите техническое задание.

# Техническое задание на разработку рекомендательной системы, для проекта UniGoTo

**Цели рекомендательной системы**

Выдача персонального ранжированного списка направлений обучения в высших учебных заведениях на основе данных, вводимых пользователем рекомендательной системы.

**Условия выполнения**

На основе больших данных, переданных заказчиком в виде API, произведите обработку данных (убрать пустые строки, шумы), а также разработайте самостоятельную рекомендательную модель и документацию. Документация должна содержать информацию по обработке первоначальных данных, обучению, сборке, использованию и переобучению.

Плюсом будет реализация в виде Docker-контейнера и реализация взаимодействия в виде API, предпочтительнее в виде GRPC с описанием схемы взаимодействия.

**Предположительные этапы работ**

Сборка данных, обработка данных для реализации системы, создание рекомендательной системы, тестирование, написание документации.

**Описание API сервиса для получения входных данных**

API будет предоставляться по адресу unigoto.ru/api/input для доступа будет использоваться входные POST-параметры:

* token (обязательный) — данный параметр содержит токен авторизации, выдаётся индивидуально, строка ненулевой длины.
* page (обязательный) — содержит номер страницы выходных данных, строка не нулевой длины, состоящая из номера в диапазоне (1-700000), данные отсортированы от старых к новым. Также рекомендуем учесть, что объём данных достаточно большой и использовать его весь не имеет смысла — важен подход к решению задачи, его финальная реализация выходит за рамки Хакатона.

**Ограничения**

Лимит на количество запросов и количество этих запросов зависит от количества участников и будет рассчитан после подтверждения участников.

**Описание выходных данных**

Выходные данные отдаются после успешного запроса и имеют формат json.

Массив из 999 элементов имеет значения:

**about** (string) — содержимое поля «О себе»

**activities** (string) — содержимое поля «Деятельность»

**books** (string) — содержимое поля «Любимые книги»

**games** (string) — содержимое поля «Любимые игры»

**interests** (string) — содержимое поля «Интересы»

**country** (object) — информация о стране, указанной на странице пользователя в разделе «Контакты». Возвращаются следующие поля:

**id** (integer) — идентификатор страны

**title** (string) — название страны

**city** (object) — информация о городе, указанном на странице пользователя в разделе «Контакты». Возвращаются следующие поля:

**id** (integer) — идентификатор города

**title** (string) — название города

**education** (object) — информация о высшем учебном заведении пользователя. Возвращаются поля:

**university** (integer) — идентификатор университета

**university\_name** (string) — название университета

**faculty** (integer) — идентификатор факультета

**faculty\_name** (string)— название факультета

**graduation** (integer) — год окончания