Кизилова М.В., Резниченко О.С., Асадуллаев Р.Г., Хорольская Е.Н. Краткий анализ методов и программных инструментов дедубликации данных в АИС «Навигатор дополнительного образования» в профиле Белгородского муниципального округа

[1,4] НИУ «БелГУ», Институт фармации, химии и биологии, г. Белгород;
[2,3] НИУ «БелГУ», Институт инженерных и цифровых технологий, г. Белгород;
[1] МАУ ДО «ЦДО «Успех» Белгородского р-на Белгородской обл.», пгт. Разумное

Одной из проблем АИС «Навигатор дополнительного образования» является дублирование данных детей при регистрации на портале. Применение методов и инструментов ИИ позволяет существенно упростить решение данной задачи. Анализ описанных в общедоступных источниках методов по дедубликации текстовых массивов позволил выделить четыре: хэширование, правила эвристики, Word Embedding (word2vec), Библиотека Python Dedupe. Критерием эффективности метода является получение на выходе списка на удаление данных и списка для анализа экспертом. В полной мере этому требованию соответствует эвристический алгоритм.

***Ключевые слова**: дополнительное образование, дедубликация данных, методы ИИ*

АИС «Навигатор дополнительного образования» является информационно-коммуникативной средой, разработанной в рамках реализации национального проекта «Образование», федерального проекта «Успех каждого ребенка» (с 1 января 2019 года). С 1 января 2023 года одна из задач функционирования АИС «Навигатор» состоит во внедрении социального заказа на оказание муниципальных услуг в социальной сфере посредством социального сертификата с номиналом, величину которого определяет муниципалитет. В рамках портала функционируют отдельные профили муниципалитетов в каждом регионе, включающие в себя профили образовательных организаций, имеющих лицензию на деятельность в сфере дополнительного образования. Работу с модулями АИС «Навигатор» в профиле муниципалитета осуществляет муниципльный администратор.

Одной из задач муниципального администратора АИС «Навигатор» является удаление дублирующихся учетных записей пользователей на портале. Причин появления дубликатов, как правило, несколько: утеря пользователем пароля к личному кабинету в АИС «Навигатор» или к почте, зарегистрированной в системе в качестве логина. В связи с этим пользователь не может восстановить пароль и регистрирует новую учетную запись с новой электронной

почтой. Так же дубликаты создают дети в возрасте от 14 до 17 лет, регистрируя новые личные кабинеты, где в качестве представителя часто указывают себя. Третьей причиной появления дубликатов является сопряжение АИС «Навигатор» и Госуслуг, когда родитель заходит в свой профиль, используя кнопку «Зайти через Госуслуги». В этом случае, если почты на этих двух ресурсах не совпадают, автоматически в АИС «Навигатор» создается новый кабинет, в который подтягиваются данные из личного кабинета на Госуслугах.

Во всех этих случаях необходимо удалять дубликаты, чтобы не перегружать профиль муниципалитета и не замедлять портал. Кроме того, наличие дубликатов создает ложные представления о количестве детей в муниципалитете, что отражается на статистике в процессе мониторинга выполнения федерального целевого показателя по охвату дополнительным образованием детей в возрасте от 5 до 17 лет.

Возникает необходимость поиска и использования методов и программных инструментов для решения задачи дедубликации внесенных в АИС «Навигатор» данных о детях. Метод должен обеспечить поиск дублирующихся записей обучающихся Белгородского муниципального округа в возрасте от 5 до 17 лет включительно, зарегистрированных в АИС «Навигатор». В настоящее время число дубликатов составляет порядка 12 тыс. записей.

Источник данных для дедубликации в АИС «Навигатор» – Модуль «Дети». Выгрузка Модуля осуществляется в муниципальном доступе в профиле Белгородского муниципального округа и представляет собой таблицу в файле .csv. Обзор предлагаемых различными источниками программных инструметов позволил выделить четыре наиболее подходящих для решения поставленной задачи метода: хэширование, правила эвристики, Word Embedding (word2vec), Библиотека Python Dedupe. Рассмотрим их краткую характеристику применительно к задаче дедубликации записей детей, зарегистрированных в личных кабинетах родителей (законных представителей) в АИС «Навигатор» в профиле Белгородского муниципального округа.

Хэширование – сравнение хеш сумм записей для быстрого поиска дубликатов. Хеш-функция (англ. Hash function от hash – «превращать в фарш», «мешанина»), или функция свертки – функция, преобразующая массив входных данных произвольного размера в выходную битовую строку определенного (установленного) размера в соответствии с определенным алгоритмом [2]. В идеале хэш-функция должна работать без коллизий, иными словами, ни одна пара различных входных значений не должна генерировать одно и то же значение хэш-функции. Существует множество алгоритмов хеширования, различающихся свойствами [4]. Примеры свойств: разрядность, вычислительная сложность, криптостойкость. Существует несколько широко используемых хэш-функций. Все они были разработаны математиками и программистами. В процессе их дальнейшего изучения было выявлено, что некоторые из них имеют недостатки, однако все они считаются приемлемыми для не криптографических приложений: MD5 (генерирует 128-битное хэш-значение), SHA-1 (Secure Hash Algorithm. SHA-1 – это первая версия алгоритма, за которой в дальнейшем последовала SHA-2, создает 160-битный (20 байт), SHA-2 (вторая версия

алгоритма, имеет множество разновидностей, наиболее часто используемая – SHA-256, которая возвращает 256-битное хэш-значение), SHA-3 (алгоритм хэширования был разработан в конце 2015 года и до сих пор еще не получил широкого применения), SHA3-256 (алгоритм с эквивалентной применимостью более раннего алгоритма SHA-256) [7]. Как правило, хэш-функции используются для проверки правильности передачи данных. Одним из таких применений является проверка сжатых коллекций файлов, таких как архивные файлы (*zip* или *rar*). Имея архив и его ожидаемое хэш-значение (обычно называемое контрольной суммой), можно выполнить собственное вычисление хэш-функции, чтобы убедиться в целостности полученного вами архива [5].

Эвристический алгоритм – это алгоритм решения задачи, правильность которого для всех возможных случаев не доказана, но про который известно, что он дает достаточно хорошее решение в большинстве случаев. В действительности может быть даже известно (то есть доказано), что эвристический алгоритм формально неверен. Его все равно можно применять, если при этом он дает неверный результат только в отдельных, достаточно редких и хорошо выделяемых случаях или же дает неточный, но все же приемлемый результат. Проще говоря, эвристика – это не полностью математически обоснованный (или даже «не совсем корректный»), но при этом практически полезный алгоритм [6]. Эвристика, в отличие от корректного алгоритма решения задачи, обладает следующими особенностями:

– она не гарантирует нахождение лучшего решения;

– она не гарантирует нахождение решения, даже если оно заведомо существует (возможен «пропуск цели»);

– она может дать неверное решение в некоторых случаях.

Эвристические алгоритмы широко применяются для решения задач высокой вычислительной сложности, то есть вместо полного перебора вариантов, занимающего существенное время, а иногда технически невозможного, применяется значительно более быстрый, но недостаточно теоретически обоснованный алгоритм. В области искусственного интеллекта, таких как распознавание образов, эвристические алгоритмы широко применяются также и по причине отсутствия общего решения поставленной задачи. Различные эвристические подходы применяются в антивирусных программах, компьютерных играх и т. д. Например, программы, играющие в шахматы, проводят середину игры, основываясь, преимущественно, на эвристических алгоритмах. Возможность (допустимость) использования эвристик для решения каждой конкретной задачи определяется соотношением затрат на решение задачи точным и эвристическим методами, ценой ошибки и статистическими параметрами эвристики. Кроме того, важным является наличие или отсутствие на выходе «фильтра здравого смысла» – оценки результата человеком.

Если же на выходе результат решения критически оценивается человеком, то ситуация становится ещё лучше: когда ошибка, выданная эвристикой, оказывается слишком мала, чтобы человек её заметил, цена этой ошибки обычно гораздо ниже, а серьёзные ошибки будут отсеяны «фильтром здравого смысла», следовательно, не нанесут существенного вреда.

Word Embedding (word2vec) – преобразует текст в векторы, затем сравнивают косинусную близость. Векторное представление слов (англ. Word embedding) – общее название для различных подходов к моделированию языка и обучению представлений в обработке естественного языка, направленных на сопоставление словам из некоторого словаря векторов небольшой размерности.

Word2vec – способ построения сжатого пространства векторов слов, использующий нейронные сети. Принимает на вход большой текстовый корпус и сопоставляет каждому слову вектор. В word2vec существуют две основных модели обучения: Skip-gram и CBOW (англ. Continuous Bag of Words). В модели Skip-gram по слову предсказываются слова из его контекста, а в модели CBOW по контексту подбирается наиболее вероятное слово. На выходном слое используется функция softmax или его вариация, чтобы получить на выходе распределение вероятности каждого слова. Для ускорения обучения моделей Skip-gram и CBOW используются модификации softmax, такие как иерархический softmax и negative sampling, позволяющие вычислять распределение вероятностей быстрее, чем за линейное время от размера словаря [1].

Библиотека Python Dedupe – библиотека для дедубликации с активным обучением. В нашем случае «повторы» – дубликаты – это структурированные записи, описывающие одну и ту же сущность (entity). Поскольку степень неточности на самом деле не ограничена, для поиска нечетких дубликатов, особенно в больших данных, требуются различные сложные алгоритмы и наборы инструментов. Эти задачи объединяются под термином entity resolution (букв, разрешение сущностей, ER).

Indexing (Blocking) – шаг, на котором происходит быстрое отбрасывание как можно большего количества ненужных сравнений пар без потери дубликатов. На вход алгоритм, реализующий данный шаг, получает сам датасет, а на выходе – набор блоков.

Block Processing – шаг, на котором блоки дополнительно очищаются от шума. Набор блоков, полученный на предыдущем шаге, трансформируется в новый набор блоков с примерно таким же показателем recall, но с более высоким precision. Как вариант, можно перебрасывать записи между блоками.

Matching – шаг, который оценивает похожесть отобранных записей. Чаще всего это некоторая функция похожести. Его выход – это взвешенный граф похожести, где узлы – это записи, а ребра содержат оценку их похожести. При этом граф не связный: оценки похожести получаются только для пар, которые находятся в одном блоке.

Entity Clustering – шаг, который собирает похожие пары в кластеры; таким образом создается набор уникальных сущностей. В основном как классические, так и SOTA-решения опираются на эту концепцию [3].

Таким образом, проанализировав методы и программные инструменты дедубликации данных, представляющих собой текстовый массив, можно с уверенностью говорить об эвристическом алгоритме, как наиболее эффективном пути решения поставленной задачи. В конечном счете при использовании данного метода должны формироваться два списка: список записей для удаления

из АИС «Навигатор» и список для анализа экспертом. Список для анализа экспертом может состоять из записей с ошибками в фамилии, имени или отчестве ребенка, дате рождения.

...

1. URL: https://colab.research.google.com/github/Combo-Breaker/NLP_DPO_2020/blob/master/sem_02/Word_embeddings.ipynb
2. URL: https://habr.com/ru/companies/ruvds/articles/747084/
3. URL: https://habr.com/ru/companies/unidata/articles/698268/9
4. URL: https://patents.google.com/patent/RU2825549C1/ru
5. URL: https://skillbox.ru/media/code/kheshfunktsiya-chto-eto-dlya-chego-nuzhna-i-kak-rabotaet/
6. URL: https://sky.pro/wiki/analytics/evristicheskij-poisk-opredelenie-printsipy-raboty-i-primenenie/
7. URL: https://skyeng.ru/it-industry/it/chto-takoye-khesh-i-kak-on-rabotayet-prostymi-slovami/
