Дипломный проект

Natural language processing.

На тему: Автоматизация проверки адресов.

СОДЕРЖАНИЕ

Обзор

Проект выбран из задачи с работы и направления NLP, которое мне интересно.

Данные для образца взяты с работы, частично и обезличенно по городу Москва и области.

Требуется преобразовывать адреса, приводить к стандартному, понятному виду.

На данный момент, каждый месяц поступают данные в виде различных списков адресов из разных источников. Все источники со своими особенностями.

Обзор

Адреса мы формируем под рассылку. Это около 40 тыс. адресов в месяц. Около 1 -1,5 тыс адресов являются неопределенными. Т е около 4 %.

Это при том, что данные уже обработаны сотрудником вручную в excel от различного мусора в данных.

В денежном выражении:

- по минимальным прикидкам, из 1 тыс адресов, 50 клиентов которых мы не получим. Приблизительно прибыль с 1 клиента в среднем 3 тыс руб. Это приблизительно минимально 150 тыс руб., ежемесячно недополучение прибыли.
- плюс время которое уходит на обработку клерком, около 1 дня.

Анализ проблем

1

Очень тяжело было найти данные по адресам, готовые. Москва и область. Город плюс улицы. Много вроде сайтов, гос. сайтов, но в большинстве случаев либо битые файлы, либо непонятные форматы содержащие кучу данных. Я не смог с этим что то сделать.

2

Парсинг как вариант. Но большинство сайтов обладает определенной иерархией связей. Это отдельная тема которой нужно заниматься.

Анализ проблем

3

Яндекс api в принципе может убирать мусорные слова, догадываться до каких то сокращений, отгадывать что это не Москва а область конкретная, но он работает не совсем. Особенно не приятно когда он начинает замены делать не пойми на что, другие страны порой экзотические. А может и заменить и на что то похожее в другом регионе. Т. е. ему надо давать более менее причесанные данные, и понимать его алгоритм работы.

Библиотека geopy вроде неплохая штука, но она даже до Яндекса не дотягивает. Тут еще более стандартизованные данные ей нужны.

И даже платные стандартизаторы, на тестовом варианте сыпят ошибками.

Анализ проблем

4 Были найдены неплохие данные по городам только Москвы, регионы, области. Но без улиц.

5 Как вариант я взял данные из базы данных с работы. Все что накопилось за все время. Но они тоже не совсем чистые и хорошие. Нуждаются в доработке. Но это получше чем те же данные, которые получаем в мусорном виде.

6 Основные моменты это мусорные слова, сокращения, разное написание, ошибки, неправильные области, регионы, разное написание.

Цель проекта

Увеличить прибыль компании, сократить время на обработку данных.
Стандартизировать свои данные в базе.
Написать решение, которое было бы эффективно в решении проблемы.

Подготовка данных

Стандартным путем для преобразования текста.

Исследование на структуру данных. Частоты появления слов. Стоп слова. Пунктуация, приведение к нижнему регистру.

Различные стемеры, лемитизация.

Архитектура и алгоритмы

Мной была взят механизм self attention, из архитектур глубокого обучения. Так как мне она была изначально интересна своим устройством. Сравнение всего со всем, я посчитал тем, что может помочь. Я пробовал различные вариации.

В каком порядке обрабатывать данные и какими частями.

Что касается архитектуры, тут особо не навешивал. Использовал TfidfVectorizer, с n gram - но они начали хромать с определенного момента и вылетать по памяти. Добавления Sheduler ов особо результаты не меняло. В качестве метрики использовал косинусное сравнение, т к посчитал, что это должно отражать мою задачу.

Архитектура и алгоритмы

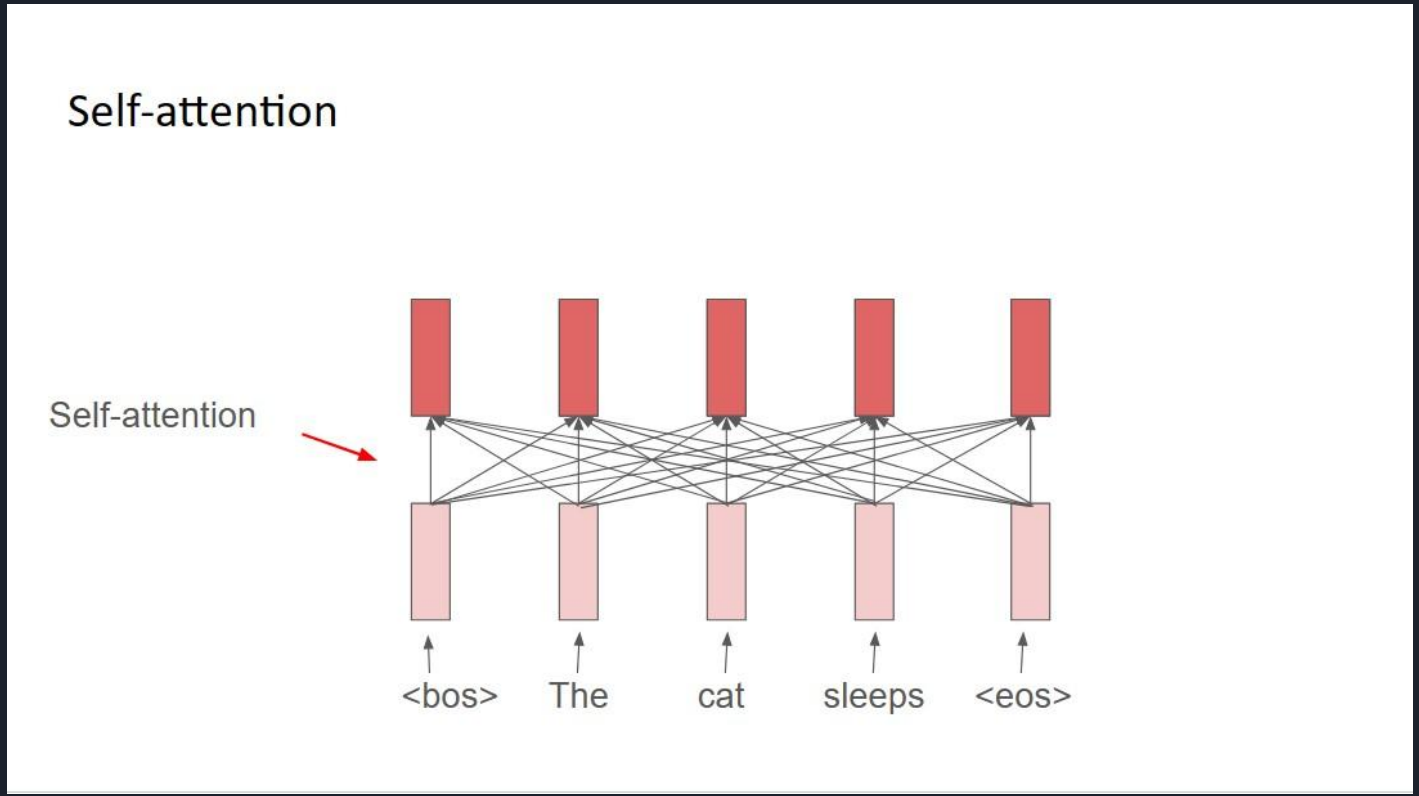
Можно сказать лучшее, что подошло это алгоритм Sequence Matcher. Оказался прост в обращении. Использовал в разных вариациях сравнения, с разными порогами схожести, удавалось повышать и время вычисления и качество. Например сравнение по Левенштайну показало лучшие результаты как по времени так и по качеству.

Список источников, что помог.

Нечеткое сравнение строк с помощью rapidfuzz / Хабр (amazingsoftworks.com)

difflib — Helpers for computing deltas — Python 3.11.4 documentation

Пример работы self attention из лекции.



Что пробовал.

Использование NER не дало результаты. Библиотеки Natasha, word 2 vec, seq2seq, NLP библиотека Spacy, понижение размерности с TruncatedSVD для того, чтобы использовать n gram.

Выводы

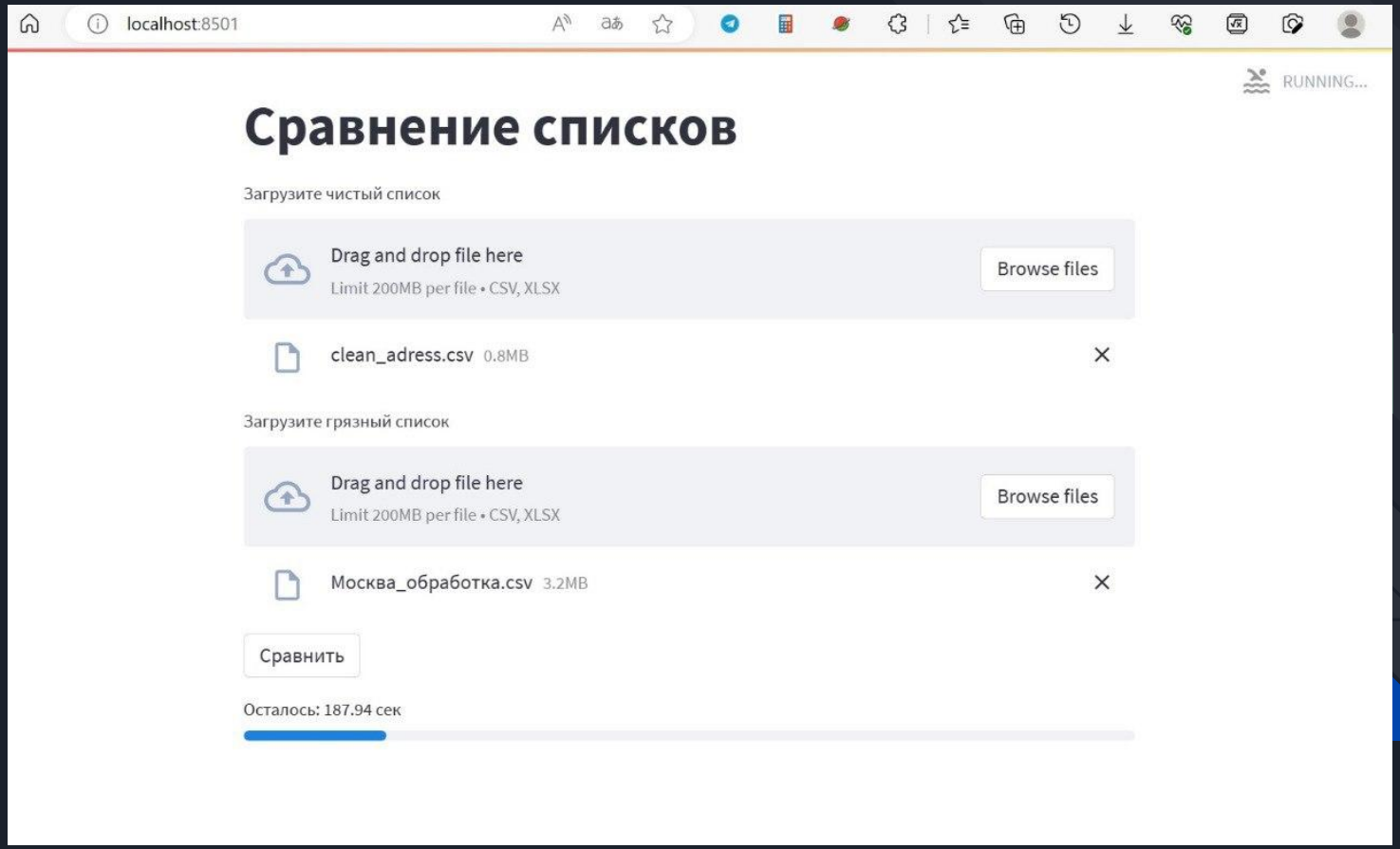
Лучшее, что сработало Sequence Matcher обработка по Левенштайну, сравнение 2 списков. Сначала города, потом отдельно улицы в пороге схожести 0,8 %,, все это конкатенируем и прогоняем через Яндекс API.

Как вариант таким способом стандартизировать базу которая есть по адресам. И так как база по сути уже устоявшаяся , новых адресов появляется не так много, то можно этим же алгоритмом все стандартизировать по сравнению.

Самое интересное, chat GPT если ему правильно расписать промты , то выдает отличные результаты по адресам. Понятно что там супер умные нейронки одна на другой.

Как вариант копать в эту сторону с Нейронками.

Обертка в стримлит



Спасибо за внимание и за курс !