Отчёт о проделанной работе Коряушкин Кирилл Русланович студент 3 курса Программной инженерии ДВФУ

Изначально я планировал обучить модель на определение ошибочного написания слова и вывода его корректной формы. Для реализации задуманного я выбрал модель **Transformer** библиотеки PyTorch. Спустя несколько дней обучения на датасете из 35 тысяч слов русского алфавита ‘**test\_word\_edit.txt’**, я пришёл к выводу, что ничего хорошего из-этого не выйдет, так как насколько бы сильно я не изменял гиперпараметры, результат не менялся, да и к тому же концептуально неправильно подошёл к решение задачи, пытаясь предугадать ошибку, а затем предугадать букву которая должна быть на месте ошибки, как мне показалось эта задача невозможна, так что я изменил подход и попытался с помощью модели **spelling correction**предугадать само слово в случае его неправильного написания. Но увы и тут я потерпел ряд неудач при обучении, так что я переключился на другой способ.

В дальнейшем я планировал использовать уже обученную модель, для вычисления embedding слова и сравнения его с embedding других слов в словаре посредством функции косинусного сходства. Изначально мой выбор пал на модель **Word2vec** и обученные векторы из файла ‘**GoogleNews-vectors-negative300.bin.gz**’, который уже содержит embedding огромного множества слов, собственно говоря, само слово которое необходимо сравнить со словарём я прогонял через модель **Word2vec** и сравнивал со значением из ‘**GoogleNews-vectors-negative300.bin.gz’** с помощью косинусного сходства. Но и тут ничего не получилось, так как функция косинусого сходства всегда возвращала одинаковые значения равные единице, от этого получалось, что исходный вектор ортогонален ко всем остальным и на выход подавался самый первый вектор, потому что я неправильно посмотрел на сам алгоритм и у меня получалось, что вектор исходного слова в любом случае не содержится в словаре и получается что он заполняется автоматически нулям, оттуда и ортогональность.   
И тут я решил всё пересмотреть использование модели Word2vec, и использовать модель **fasttext** и на удивление мне удалось найти результаты её обучения на основе русских слов ‘**cc.ru.300.bin.gz’**, здесь я уже не допустил подобной оплошности и правильно всё рассмотрел, как итог получился рабочий код, единственным минусом которого является его крайне долгая загрузка, так как исходный файл модели очень тяжеловесный.

Так же в случае неудачи у меня была запасная программа, основанная на алгоритме Левенштейна, хоть она и не относится к машинному обучению, но всё в задании тоже не говорится об их непременном использовании.

Результат работы

В конечном счёте получилась программа, которая возвращает правильное написание одного из этих слов: банан, апельсин, виноград, аптека, муравей, фасад, огонь, зайчик, ежеминутно, винт, антарктика, процент, слух, судак

Пример работы программы: на вход подаём слово “оптека”, на выходе получаем  и тоже самое со всеми вышеперечисленными словами.