**Слайд 1**

Добрый день. Меня зовут Александр Широков и в рамках доклада я расскажу о том, как получить качественное векторное представление текста и применить его к решению прикладной задачи на примере соревнования по машинному обучению.

**Слайд 2**

Выучивание векторного представления текста в каком-либо виде является весьма популярной и важной задачей. С помощью векторного представления текста ищутся релевантные документы и проводится анализ тональности текста; с помощью векторного представления слова решаются задачи предсказания следующего слова и распознавания именованных сущностей и.т.д. Но для решения данной задачи текстовые модели необходимо обучить на некотором наборе слов, который не всегда является идеальным – хорошо, если мы можем обучаться на наборе слов википедии или каком-нибудь официальном источнике данных (например, лента новостей) – данные являются не зашумлёнными, состоят из последовательно идущих предложений с очень маленькой вероятностью опечаток. Но что делать, если приходится работать с чрезвычайно зашумлёнными данными – например, речь человека в диалогах вконтакте либо информация о товаре в чеках? Если обучать модели в исходном виде, то ввиду большой зашумлённости качество наверняка будет низким, поэтому в данной работе я заостряю большое внимание на том, чтобы подготовить данные к обучению.

**Слайд 3**

Даже для качественных корпусов слов все равно необходима первичная обработка – в неё входит:

- **Токенизация** – разделение текста на более мелкие части, токены (обычно ими являются слова, но возможны и другие разбиения)

- **Удаление лишних символов –** в данном пункте текст очищается от различных ненужных символов пунктуации с помощью REGEX (Regular Expressions) выражений, текст приводится в нижний регистр и убираются слова-архаизмы – неиспользуемые слова не являются важными признаками текста

- **Удаление стоп-слов –** таких слов, которые часто встречаются, но не несут особого смысла – они могут быть мешать

- в некоторых моделях может потребоваться **Лемматизация -**  приведение слов к начальной морфологической форме с помощью словаря и грамматики языка (хорошо работает в русском языке) и **Стемминг –** нормализация слов путём отбрасывания окончаний, согласно правилам, основанным на грамматике языка

Данные методы первичной предобработки обычно проводят с любым набором текстов. Разберёмся теперь с менее поверхностными алгоритмами, которые позволяют сделать зашумлённый текст ещё чище, а также позволяют выделить некоторые важные новые сущности – последовательно разберём как разбивать слово на сегменты, как исправлять опечатки в тексте и как искать и исправлять сокращения слов.

**Слайд 4**

Задача сегментации текста возникает в различных прикладных задачах: при анализе хэштегов в социальных сетях (хэштегов идут без пробелов), в письменной речи пользователей часто залипает клавиша backspace, в зашумлённых данных (например, товарных чеках) и.т.д. На вход подаётся строка без пробелов, а на выходе необходимо найти наилучшую подпоследовательность слов внутри данной строки. Разберём алгоритмы, основанные на корпусе слов – так называется заранее определенный словарь известных слов, которые могут встретиться в тексте.

**Алгоритм максимального соответствия**  и **обратного максимального соответствия** — это жадные алгоритм, в котором в тексте последовательно выделяются слова **наибольшей длины**, которые встретились в словаре, только в первом случае поиск происходит с начала строки, а **во втором – с конца.**

**Двунаправленный алгоритм максимального соответствия является надстройкой над предыдущими алгоритмами – он позволяет выбрать, какая подпоследовательность в данной ситуации является более вероятной. Для этого на первом шаге применяются к входящей строке алгоритмы максимального соответствия и обратного максимального соответствия, а затем выбирается наименее сегментированная последовательность (с наименьшим количеством неизвестных слов).**

**До сих пор рассматривались лишь методы основанные на длине слов, но можно использовать ещё и вероятностные модели – из заданной строки выделим всевозможные подпоследовательности, подсчитаем вероятности каждой подпоследовательности как произведение вероятностей встречаемости каждого слова, входящего в подпоследовательность, и выберем ту, у которой вероятность встречаемости наибольшая. При отсутствии слова в словаре чтобы избежать обнуления произведений вероятностей применяется сглаживание Лапласа – искусственное прибавление единицы к встречаемости каждого слова.**

**На данной последовательности действий работают два последних алгоритма на слайде с единственным отличием – в последнем случае перемножаются вероятности биграмм – последовательностей из двух слов. Данный вариант позволяет лучше учесть контекст, в котором находятся слова, но занимает больше памяти.**

**Слайд 5**

Для исправления опечаток использовался алгоритм Питера Норвига, с добавленной мною N-грам моделью. Пусть есть некоторое слово и для данного слова необходимо найти наиболее вероятную правку . Алгоритм прост – находим всех кандидатов , которые достаточно близки к и выбираем наиболее вероятный среди них. Осталось разобраться что такое достаточно близкое слово и для этого существует расстояние Левенштейна – как написано на слайде, минимальное необходимое количество удалений, перестановок, вставок и замен символов, необходимых, чтобы одно слово превратить в другое. В предположении, что пользователь сделал не больше двух из данных действий при написании неправильного слова, мы можем сгенерировать (внимание на таблицу на слайде) всевозможные удаления букв, перемена мест букв, замены букв на другую и вставку букв в слово, а затем выбрать наиболее вероятное с учётом контекста – в этом помогает прикрученная сверху мною надстройка в виде биграмной модели.

**Слайд 6**

Исправление сокращений тоже весьма популярная и пока что не до конца доработанная тема, однако нахождение является важной частью по выделению важных сущностей в тексте и сокращению словаря уникальных слов. Проблема заключается в том, что человеку свойственно придумывать что-то новое и иногда там, где этого делать не стоит – например, придумывать новые сокращения для существующих слов. Так, например, для слова шампунь можно придумать около 7-ми различных сокращений – и все они будут подразумевать под собою исходное слово.

Для исправления сокращений я написал следующий простой метод:

- если сокращение состоит из одного слова, то ищем, в словаре слова, начинающиеся с данного слова (шоко.)

- если сокращение написано через дефис (ш-лад), то я ищу слова, которые начинаются на первую часть слова, разделённую дефисом, и заканчиваются на вторую часть.

- если слова написаны через backslash, то я проверяю в словаре биграмм те биграммы, в которых слова начинаются с данных частей, разделённых backslash-ем.

Над данным алгоритмом я опять же прикручиваю сверху N-грам модель, чтобы попытаться учитывать контекст. В некоторых ситуациях, как видно на слайде, сделать текст чище данный алгоритм помогал.

**Слайд 7**

Рассмотренные методы вторичной обработки и очистки текста удалось применить на недавнем соревновании Data Fusion Contest, проводимым банком ВТБ, в котором по текстовому описанию чека необходимо было определить, к какой из 96-ти уникальных категории относится товар (некоторые из них представлены в правой части слайда). В качестве исходных данных выдавался датасет, содержащий более 8 миллионов уникальных чеков, практически каждый из которых представлял из себя чрезвычайно зашумлённые и неструктурированные текстовые данные. В 30 процентах чеках возможно было произвести сегментацию текста, в 55 процентах были допущены опечатки, а в 81 проценте были использованы сокращения товаров. В качестве метрики качества была использована взвешенная F1-мера.

**Слайд 8**

При решении задачи я использовал весьма типичные для машинного обучения подходы векторного представления текста. В первом подходе было использовано представление текста в виде TF-IDF (Team Frequency – Inverted Document Frequency), в котором TF отвечает за то, насколько часто слово встречалось в чеке , IDF отвечает за то, насколько часто слово встречалось во всем корпусе чеков. После построения разреженной матрицы над нею обучался классификатор метод опорных векторов (SVM).

Второй подход заключался в использовании модели Word2Vec, предназначенная для получения векторных представлений слов, обучаясь следующим образом – Word2Vec принимает на вход большой текстовый корпус (в данном случае обученный на 8 миллионах чеках) в качестве входных данных и для каждого слова получает векторное представление слова, основываясь на контекстной близости слов: слова, встречающиеся в тексте рядом с одинаковыми словами (следовательно, имеющие похожий смысл), будут иметь близкие (по косинусному расстоянию) векторы. Над полученным эмбеддингами – векторными представлениями слов – строилась модель K-ближайших соседей с косинусной метрикой расстояния.

И третьим способом была обучена модель FastText, которая по своей идее очень похожа на Word2Vec, но с тем отличием, что каждое слово дробится на символьные n-граммы, что позволяет решить проблему редких слов, а также способна выучивать не только эмбеддинги слов, но и предложений.

**Слайд 9**

В качестве итоговых результатов я хотел сравнить, насколько предложенные мною методы вторичной обработки текстов способны улучшить качество векторного представления текста и его разделимости, относительно других классов. После проведения экспериментов выяснилось, что тщательная предобработка именно в данном соревновании помогла улучшить качество на более чем на 3 процента в среднем, чем без обработки – повысилось качество эмбеддингов и понизилось количество уникальных слов в словаре для обучения, так как были убраны опечатки и слова, состоящие из сегментов, а также сокращения.

**Слайд 10**

Выводы: были реализованы методы первичной и вторичной предобработки текстов, а также протестирована их боеспособность на соревновании по машинному обучению, из чего можно сделать рекомендацию – перед обучением текстовых моделей весьма желательно тщательно предобработать текст. В качестве бонуса скажу, что данное решение заняло 3-е место среди публичных решений соревнования и, несмотря на то, что в соревновании победили более продвинутые и современные нейросетевые методы, решение позволило зацепиться за бронзовую медальку соревнования (не путать с 3-им местом), попав в 50 лучших моделей.

Решение соревнования, как и реализация алгоритмов предобработки текстов находится на моем GitHub, с которыми можно ознакомиться, перейдя по ссылкам.

**Слайд 11**

Спасибо за внимание.