

Математические методы анализа текстов

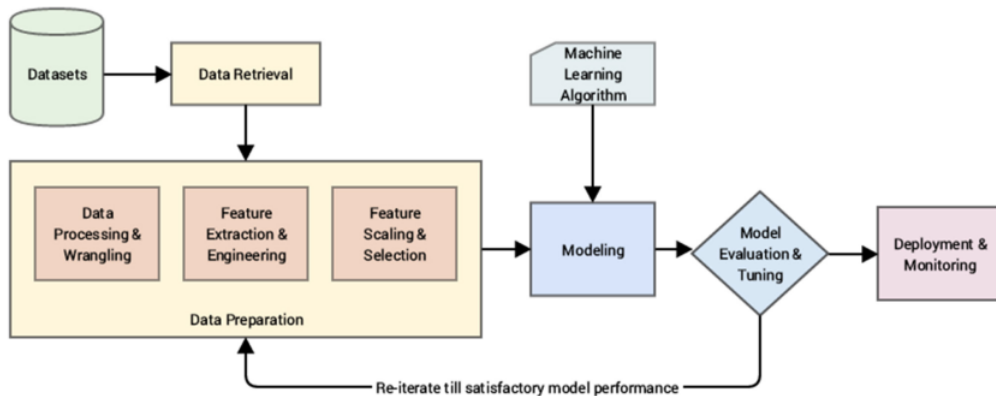
Лекция

Индустриальный ML-пайплайн на примере классификации текстов

Мурат Апишев (mel-lain@yandex.ru)

5 ноября, 2019

Как выглядит стандартный пайплайн



Решим продуктовую задачу

Контекст:

делаем сервис по агрегации новостного контента

Неформальная постановка задачи:

борьба с «чернушными» новостями

Формальная постановка задачи:

нужно придумать

Метод решения:

нужно придумать

Метрика:

нужно придумать

Внедрение:

открытый вопрос

Формализация постановки

- ▶ Прежде всего нужно решить, какая именно задача будет решаться
- ▶ В нашей ситуации задача очевидно делиться на две части:
 - ▶ создание классификатора «чёрного контента»
 - ▶ подбор стратегии работы с ним
- ▶ Нужно дать формальное описание текстового контента, который нужно уметь выявлять
- ▶ Попробуем для начала такое:
 - ▶ тексты, описывающие убийства, пытки и т.п.
 - ▶ тексты про бытовые конфликты, оскорбления
 - ▶ скандальные новости, низкосортный «жёлтый» контент
- ▶ Скорее всего, эти формулировки будут меняться в процессе решения, это нужно иметь ввиду

Метрика

- ▶ Важно с самого начала решить, по какой метрике будет вестись приёмка
- ▶ Приёмочные метрики классификатора: F1-мера, точность, полнота и ROC-AUC
- ▶ Бизнесу, как правило, эти числа неинтересны
- ▶ Важно не только то, как хорошо мы определяем «чёрный» контент, но и то, как мы пользуемся этим знанием в продакшене
- ▶ Метрики, по которым будет определяться стратегия показа выявленного контента, и будет целевой для всей системы
- ▶ Считают такие вещи обычно в А/Б-тесте
- ▶ Что это могут быть за метрики:
 - ▶ CTR
 - ▶ Доля жалоб на контент
 - ▶ Клики на рекламу
 - ▶ Все эти метрики в разных когортах пользователей
 - ▶ ...

Предварительный итог

- ▶ Задача разделена на две последовательные части, есть общее видение решения
- ▶ Сперва будет строиться классификатор текстов, понятно, как измерять качество
- ▶ Дано первое определение целевого контента

Что дальше:

- ▶ найти данные
- ▶ определиться с предобработкой и фичами
- ▶ выбрать модели и обучить их
- ▶ убедиться, что всё ОК согласно и по метрике, и здравому смыслу

Данные

- ▶ Ключевое отличие академических курсов от реальной жизни – отсутствие в общем случае данных для обучения
- ▶ Поскольку у нас новостной сервис, сырые данные есть
- ▶ Но для них нет разметки, как можно её получить?

Варианты:

- ▶ Поискать тексты, на которые были жалобы пользователей
- ▶ Изучить «чернушные» тексты и определить, какими словами они часто характеризуются
- ▶ Разметить часть документов руками
- ▶ Воспользоваться сервисом для разметки (Yandex.Toloka, Amazon Mechanical Turk)

- ▶ Сервис, объединяющий сообщество людей, делающих разметку, и их потенциальных заказчиков
- ▶ Предоставляет интерфейсы для решения большинства задач разметки:
 - ▶ Категоризация
 - ▶ Парное сравнение
 - ▶ Сегментация изображений
 - ▶ Генерация текста
 - ▶ Разметка последовательности
 - ▶ ...
- ▶ Даёт возможности отбора размещающих, контроля качества разметки, отложенной оплаты и проч.
- ▶ Имеет API, вокруг которого можно написать библиотеки для автоматизации процессов загрузки данных и отгрузки результатов

Этапы работы с Толокой

- ▶ **Написание инструкции**

Мы хотим показать текст и попросить выбрать класс

- ▶ **Подготовка визуального шаблона**

- ▶ **Настройки фильтров пользователей и контроля качества**

Выбираем наиболее качественных пользователей и ставим ограничения на количество ошибок и скорость разметки

- ▶ **Генерация и загрузка заданий в виде пулов**

Собираем множество текстов на разметку, добавляем немного правильных ответов для онлайн-контроля разметки

- ▶ **Запуск разметки, валидация результата, оплата**

- ▶ **Выгрузка и использование результатов**

Но сперва...

- ▶ Прежде, чем тратить деньги на Толоку, нужно понять, что размечаем
- ▶ В нашей задаче классы очень несбалансированные
- ▶ Нельзя просто набрать случайные документы и отправить в разметку, получится слишком мало примеров целевого класса

Что можно сделать:

- ▶ Можем собрать грязную небольшую выборку с помощью регулярок и априорной информации
- ▶ Обучить на этом очень простую модель, например, лог-регрессию
- ▶ Прогнать через неё документы и набрать выборку для разметки из документов с низким значением уверенности

Возвращаясь к Толоке

Рассмотрим процесс разметки по шагам.

Написание инструкции: задание должно быть написано максимально чётко, всеобъемлюще, однозначно и с примерами, по возможности – кратко

Как надо писать:

В задании показывается текст, нужно определить, «чернушный» он или нет. Текст «чернушный», если он . . .

Как НЕ надо писать:







В данном задании Вам предстоит решать задачу классификации новостей. Классификация бинарная, то есть на два класса, один из которых назовём условно «чернушным» . . .

Подготовка визуального шаблона

Шаблоны

Шаблоны позволяют сформировать и запустить задания, исходя из ваших потребностей. Вы можете использовать шаблон как есть или адаптировать его под ваши входные данные и формат получаемых ответов.

Классификация

 <p>Оценка видео</p> <p>Позволяет просмотреть видео и выбрать один из возможных вариантов. В шаблоне видео-плеер и несколько радио-кнопок.</p> <p><input type="radio"/> <input type="radio"/></p> <p><input type="button" value="Выбрать"/> <input type="button" value="Предпросмотр"/></p>	 <p>Категоризация изображений</p> <p>Подходит для классификации изображений и проставления тегов. В шаблоне изображение и радиокнопки.</p> <p><input type="radio"/> <input type="radio"/> <input checked="" type="radio"/></p> <p><input type="button" value="Выбрать"/> <input type="button" value="Предпросмотр"/></p>
 <p>Модерация контента</p> <p>Позволяет классифицировать текстовые комментарии. В шаблоне текст и радиокнопки.</p> <p><input checked="" type="checkbox"/> <input type="checkbox"/></p> <p><input type="button" value="Выбрать"/> <input type="button" value="Предпросмотр"/></p>	 <p>Категоризация изображений с заголовком на странице</p> <p>В шаблоне шапка с общей информацией для всей страницы с заданиями, картинка и радиокнопки для каждого задания.</p> <p><input type="radio"/> <input type="radio"/> <input type="radio"/></p> <p><input type="button" value="Выбрать"/> <input type="button" value="Предпросмотр"/></p>
 <p>Категоризация текста с дополнительными опциями</p>	 <p>Категоризация поисковых запросов</p> <p>В шаблоне текст, кнопки для поиска в интернете и несколько</p>

Настройка фильтров и контроля

- ▶ Ограничить качество пользователей
- ▶ Определить навыки пользователей (например, знание языка)
- ▶ Установить значение *перекрытия* для каждого задания (обычно 3-5)
- ▶ Ограничить число или долю ошибок относительно перекрытия
- ▶ Ограничить число или долю ошибок в капче (соответственно, выставить частоту капчи)
- ▶ Ограничить число или долю ошибок в проверочных заданиях
- ▶ Установить верхние и нижние границы времени на задание, банить за слишком быстрые ответы
- ▶ Установить стоимость выполнения заданий

Создание и загрузка пула

- ▶ Пул представляет собой набор заданий для разметки внутри текущего проекта
- ▶ Для его формирования нужно создать tsv-файл, в котором будут содержаться необходимые данные
- ▶ В нашем случае это могут быть колонки с текстом, его идентификатором и правильным ответом
- ▶ Ответ для большинства заданий будет пустым
- ▶ Задания, у которых он проставлен, будут считаться контрольными
- ▶ После загрузки пула с учётом перекрытия будут сформированы итоговые страницы с заданиями и подсчитаны статистики

Запуск разметки и мониторинг

ЗАДАНИЯ ПУЛА (Пример загрузочного файла (tsv, кодировка UTF-8))

Загрузить

файлы

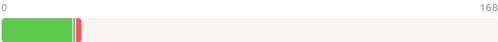
Предпросмотр

56 страниц заданий	0 обучающее задание
221 заданий	0 контрольное задание

16 %

Сделано 27, принято 24

Проверить задания 1



СТАТИСТИКА ПУЛА

<div>38сек</div> <div>Среднее время выполнения задания</div>		<div>—</div> <div>Приблизительное время завершения</div>	<div>0,72 (+ 0,14)</div> <div>Израсходованные средства (+ надбавка)</div>		<div>4,98 (+ 1,00)</div> <div>Приблизительный бюджет (+ надбавка)</div>	
<div>116 чел.</div> <div>Активные пользователи, которым доступны задания</div>	<div>14 чел.</div> <div>Заинтересованные пользователи</div>	<div>6 чел.</div> <div>Взявшие задания пользователи</div>	<div>4,50</div> <div>Выполненные задания на одного пользователя</div>	<div>9 шт.</div> <div>Просроченные задания</div>	<div>1 шт.</div> <div>Пропущенные задания</div>	

Предобработка данных

- ▶ Теперь у нас есть результаты разметки, с учётом перекрытия можно получить ответы, которым можно доверять
- ▶ Далее начинается обычная предобработка:
 - ▶ Токенизация
 - ▶ Лемматизация
 - ▶ Фильтрация
 - ▶ Выделение N-грамм
 - ▶ ...
- ▶ Желательно сформировать обучающую выборку так, чтобы классы были представлены в примерно равных пропорциях
- ▶ Если даже после описанного выше способа организации разметки дисбаланс ещё существенный – можно аугментировать данные

Аугментация текстов (опционально)

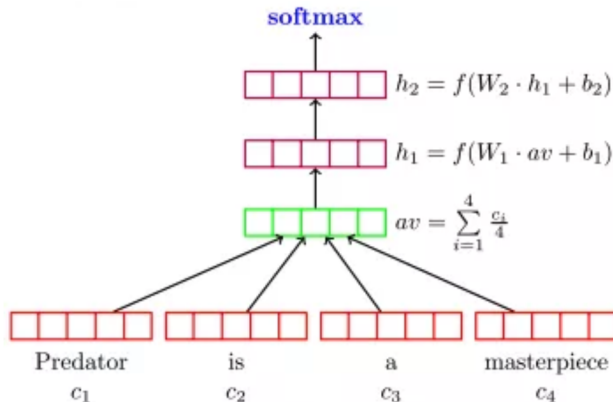
- ▶ В простейшем случае можно делать перевзвешивание объектов, увеличивая веса объектов меньшего класса
- ▶ При наличии тезауруса с синсетами, можно делать случайные замены слов в текстах меньшего класса на синонимы
- ▶ При использовании сжатых векторных представлений текстов можно применять методы аугментации в линейном пространстве (SMOTE, ASMO)
- ▶ Для работы с несбалансированными выборками есть библиотека `imbalanced-learn`

Моделирование

- ▶ Базовая модель, с которой нужно начинать решение – логистическая регрессия на «мешке слова» и TF-IDF
 - ▶ легковесная модель, быстрое обучение
 - ▶ вероятностная интерпретация выхода классификатора
 - ▶ интерпретируемость весов признаков
 - ▶ куча реализаций (sklearn, Vowpal Wabbit)
- ▶ Хороший вариант – FastText
 - ▶ более сложная модель, но обучение всё равно быстрое и параллельное
 - ▶ строит хорошие векторные представления
 - ▶ работает с символьными N-граммами
 - ▶ можно работать из Python, C++, Java
- ▶ Ещё неплохо работают свёрточные сети
 - ▶ тоже работает на уровне символов и строит эмбединги
 - ▶ хорошо подходит для коротких документов
 - ▶ интегрируемость зависит от фреймворка

FastText как классификатор

- ▶ Сперва строятся эмбединги слов из обучающей коллекции
- ▶ Далее поверх них обучается классифицирующая сеть
- ▶ Входом являются усреднённые эмбединги слов документа



FastText: пример

Данные подаются в виде файла, каждый документ – одна строка вида

```
--label__food-safety --label__acidity Dangerous pathogens capable of  
growing in acidic environments
```

Обучение и сохранение модели:

```
1 classifier = fasttext.supervised('data.train.txt', 'model')
```

Применение обученной модели к тестовым данным:

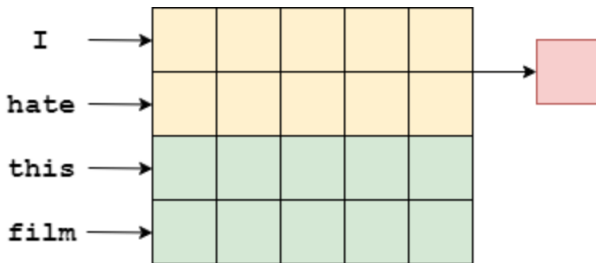
```
1 result = classifier.test('test.txt')
```

Наиболее вероятные классы (и их вероятности):

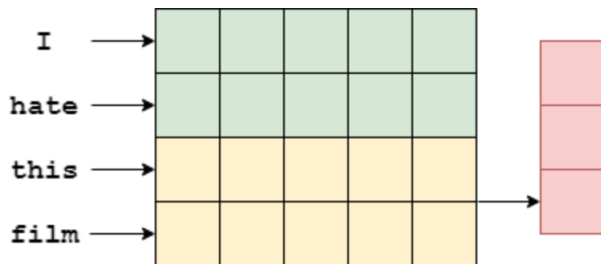
```
1 labels = classifier.predict(texts, k=3)  
2 labels = classifier.predict_proba(texts, k=3)
```

CNN для текстов

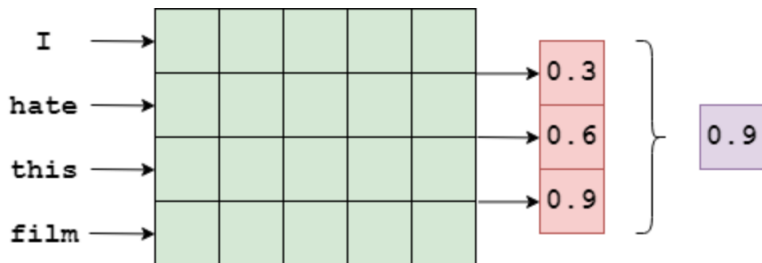
- ▶ Для применения свёрточных сетей к текстам необходимо привести текст к матричному виду
- ▶ Для этого можно воспользоваться эмбедингами
- ▶ Возьмём батчи фиксированного размера, при необходимости используем либо нарезку документов, либо, наоборот паддинг
- ▶ Для работы с текстами используются одномерные свёртки (вторая размерность всегда равна размерности эмбедингов)



CNN для текстов



Следующий шаг, как и в свёрточных сетях для изображений – max-пуллинг:



CNN для текстов

- ▶ Основная размерность свёрток может быть разной, что позволяет выделять признаки на N-граммах различной длины
- ▶ Идея в том, что наибольшее значение наиболее важного признака соответствует наиболее важной N-грамме текста
- ▶ Получить его можно с помощью max-пуллинга, за счёт изменения backpropagation-ом весов свёрток для максимизации значения признака, наиболее влияющего на предсказание метки класса
- ▶ Число фильтров будет определять размерность выходного эмбединга текста, который далее передаётся в линейный слой с софтмаксом на выходе для предсказания тональности

Ну что, задача решена?

- ▶ Нет! По факту, ни одна из поставленных задач ещё не является решённой
- ▶ Мы обучили классификатор, но в продакшене он не протестирован
- ▶ Стратегия его применения тоже не отработана, бизнес-метрики не посчитаны
- ▶ Да и сам классификатор пока существует только в нашем ноутбуке
- ▶ Следующий шаг – это интеграция, мониторинг и автоматизация

Звучит прикольно, а что это?

- ▶ **Интеграция** – это внедрение решения в существующую инфраструктуру
- ▶ Например, у компании уже может быть система для оффлайн-обучения моделей и выкатки их в продакшн
- ▶ Тогда весь свой код нужно перенести в него + добавить туда нужные библиотеки, если их не хватает
- ▶ **Мониторинг** – качество работы модели нужно измерять постоянно в онлайн-режиме, сигнализируя о проблемах
- ▶ **Автоматизация** – очевидно, что почти все сделанные выше шаги будут повторяться многократно:
 - ▶ Разметка очередной порции данных
 - ▶ Дообучение классификатора
 - ▶ Валидация его качества
 - ▶ Выкатка новой версии модели в продакшн
 - ▶ Подбор порога бинаризации

Допустим, мы всё сделали правильно

- ▶ Система запущена, сама отправляет данные на разметку, дообучается, мониторит себя и выкатывает в продакшн
- ▶ Хотим использовать результаты для управления потоком показов
- ▶ Какие гипотезы можно пробовать проверять:
 - ▶ Попробуем полностью исключить «чернуху» из потока
 - ▶ Попробуем заменять такой контент на семантически близкий, но менее отталкивающий
 - ▶ Попробуем выделить пользователей, которые склонны к «чёрным» текстам, и показывать их только им
 - ▶ Попробуем этим же пользователям показывать больше подобного контента
- ▶ Принимаемся по какой-нибудь из метрик, например, CTR в А/Б-тесте
- ▶ В одной из групп улучшение получилось статистически значимым
- ▶ **PROFIT!**

Прошло два месяца...

- ▶ Уже неделю показы стабильно падают
- ▶ Смотрим мониторинг, видим, что дело в классификаторе «чёрного» контента
- ▶ Мониторинг показывает, что он выдаёт всё больше и больше положительных ответов
- ▶ Анализ показал, что банятся исторические тексты про войну, потому что в них много негатива
- ▶ Смотрим в календарь – середина апреля, видимо, к 9 мая стали писать больше о ВОВ
- ▶ Расследование завершено, надо действовать

ML – процесс циклический

- ▶ Возвращаемся к этапу формирования выборки
- ▶ Теми же методами формируем выборку для классификатора исторических текстов
- ▶ С его помощью корректируем выборку для классификатора «чёрного» контента
- ▶ Уточняем инструкцию в Толоке
- ▶ Прогоняем весь пайплайн – проблема решена!

Какие в итоге выводы

- ▶ В индустриальных задачах собственно моделирование является самый быстрым и простым этапом решения
- ▶ Очень важно правильно сформулировать задачу, метрики и хотя бы примерно численно оценить пользу от решения
- ▶ Этап сбора и подготовки является ключевым, от качества данных по факту зависит практически всё
- ▶ Решить задачу – это полдела, не менее важно внедрить решение, сделать его прозрачным, поддерживаемым и легко модернизируемым