Работа с текстами

Елтышев Женя, DMIA 2016

Этапы работы

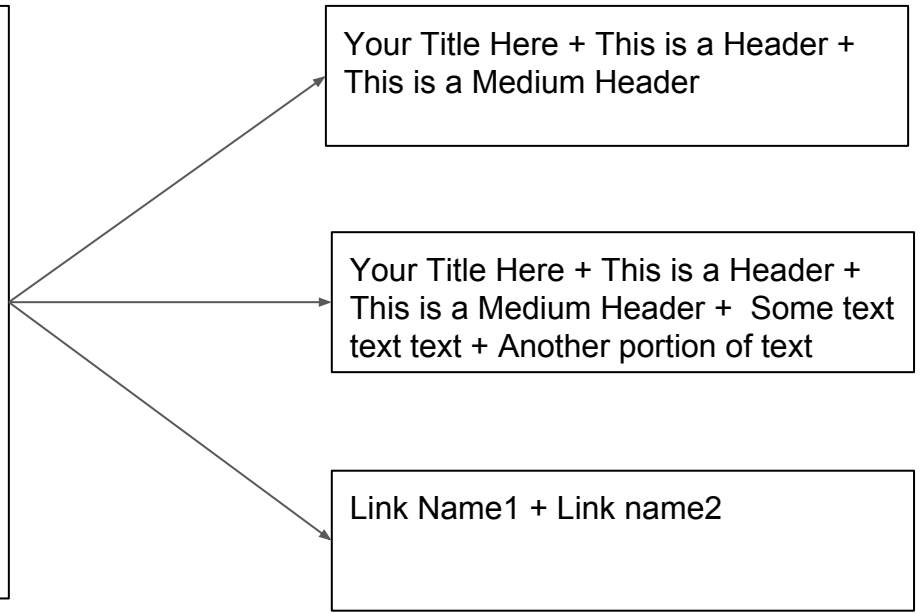
- Предобработка
- Токенизация
- Извлечение признаков
- ~~XGBoost~~ Машинное обучение

Преобработка

- Извлечение текста (из HTML, PDF, etc)
  - Для HTML - BeautifulSoup
  - Можно варьировать способ извлечения для создания отличающихся наборов фичей

Извлечение текста из HTML

```
<HTML>
<HEAD>
<TITLE>Your Title Here</TITLE>
</HEAD>
<BODY>
<a href="http://somegreatsite.com">Link Name1</a>
is a link to another nifty site
<a href="http://somegreatsite.com">Link Name2</a>
is a link to another nifty site
<H1>This is a Header</H1>
<H2>This is a Medium Header</H2>
<P> Some text text text </P>
<P> <B>Another portion of text </B> </P>
</BODY>
</HTML>
```



Преобработка

- Извлечение текста (из HTML, PDF, etc)
  - Для HTML - BeautifulSoup
  - Можно варьировать способ извлечения для создания отличающихся наборов фичей
- Удаление стоп-слов
  - Готовые списки - `from nltk.corpus import stopwords`
  - Фильтрация по частоте - убрать топ100 самых частых слов
  - Можно сделать автоматически - TF-IDF

Преобработка

- Извлечение текста (из HTML, PDF, etc)
  - Для HTML - BeautifulSoup
  - Можно варьировать способ извлечения для создания отличающихся наборов фичей
- Удаление стоп-слов
  - Готовые списки - `from nltk.corpus import stopwords`
  - Фильтрация по частоте - убрать топ100 самых частых слов
  - Можно сделать автоматически - TF-IDF
- Удаление пунктуации

Преобработка

- Извлечение текста (из HTML, PDF, etc)
  - Для HTML - BeautifulSoup
  - Можно варьировать способ извлечения для создания отличающихся наборов фичей
- Удаление стоп-слов
  - Готовые списки - `from nltk.corpus import stopwords`
  - Фильтрация по частоте - убрать топ100 самых частых слов
  - Можно сделать автоматически - TF-IDF
- Удаление пунктуации
- Лемматизация: быстрая -> быстрый
  - Библиотека pymorphy2 (для русского)

Преобработка

- Извлечение текста (из HTML, PDF, etc)
  - Для HTML - BeautifulSoup
  - Можно варьировать способ извлечения для создания отличающихся наборов фичей
- Удаление стоп-слов
  - Готовые списки - `from nltk.corpus import stopwords`
  - Фильтрация по частоте - убрать топ100 самых частых слов
  - Можно сделать автоматически - TF-IDF
- Удаление пунктуации
- Лемматизация: быстрая -> быстрый
  - Библиотека pymorphy2 (для русского)
- Стемминг: classes -> class
  - nltk.stem

Токенизация

- Bag of Words
  - sklearn.feature_extraction.text.**CountVectorizer**

Токенизация

- Bag of Words

Феррари врезалась в Запорожец ->

Феррари	врезаться	Запорожец
1	1	1

Запорожец врезался в Феррари ->

Феррари	врезаться	Запорожец
1	1	1

Токенизация

- Bag of Words
  - sklearn.feature_extraction.text.**CountVectorizer**
- N-gramm BoW
  - sklearn.feature_extraction.text.**CountVectorizer** с параметром **ngram_range** = (1, 3)

Большой недостаток Bag of Words

Опечатки!

1) Шол мидведь по лесу, видет — машина гарит. Сел в ние и сгарел.
2) Шел медведь по лесу, видит — машина горит. Сел в нее и сгорел.

cosine_similarity(text1, text2) = 0.4

Символьные N-граммы

Шол мидведь по лесу, видет — машина гарит. Сел в ние и сгарел.

шолмидведьполесувидетмашинагаритселвниеисгарел

{“шо”: 1, “ол”: 2, “лм”: 1, “ми”: 1, “ид”: 2, …}

Символьные N-граммы

мидведь ⟶ [1 1 2 1 1 1 1 1 1 1 0 1 1]

медведь ⟶ [1 1 2 1 1 2 2 0 0 1 1 0 1]

cosine_similarity(“медведь”, “мидведь”) = 0.83

Токенизация

- Bag of Words
  - sklearn.feature_extraction.text.**CountVectorizer**
- N-gramm BoW
  - sklearn.feature_extraction.text.**CountVectorizer** с параметром **ngram_range** = (1, 3)
- Char ngramms
  - Устойчивы к опечаткам
  - Не требуют лемматизацию и стемминг
  - Небольшая размерность

Извлечение признаков

- Bag of words
- TF-IDF
- Word embeddings

TF-IDF

$$\mathrm{tf}(t,d) = \frac{n_i}{\sum_k n_k}$$

$$\mathrm{idf}(t,D) = \log \frac{|D|}{|(d_i \supset t_i)|}$$

TF-IDF

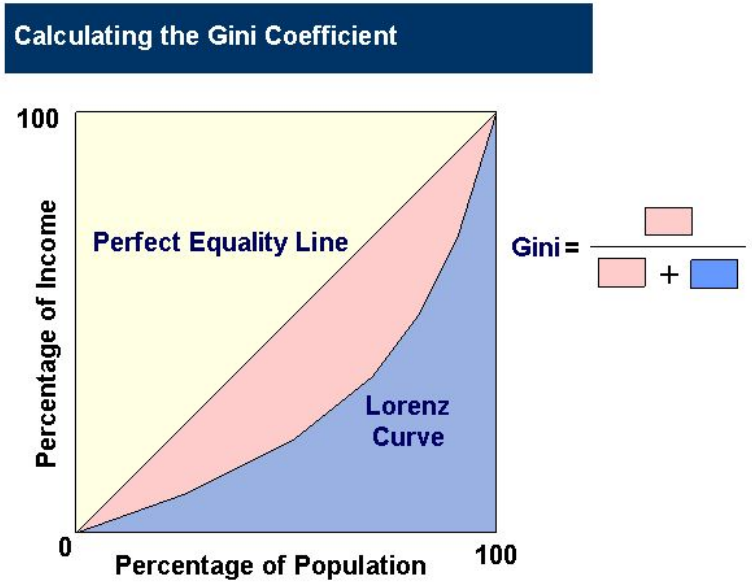
Рассмотрим задачу классификации новостей по категориям:

- Спорт
- Политика
- Экономика
- Технологии

TF-IDF

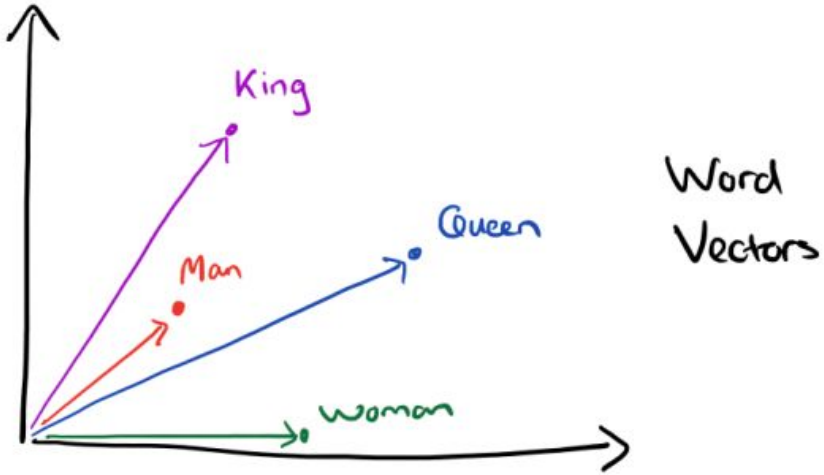
- Подсчитаем распределение слова по темам
- Чем более равномерное - тем менее “информативное слово”

IDF(t, D) = 1 / GiniCoefficient(t, D)



Word embeddings

Переводит BoW вектора в пространство меньшей размерности



Проблемы:

- Как сравнивать два текста?

Подсчет вектора для текста

- Просуммировать
- Усреднить
- Усреднить с весами (например TF-IDF)

Машинное обучение: особенности

- Очень много признаков - лучше использовать линейные методы
- Часто хорошо работает снижение размерности - PCA, Feature hashing
-

Пример: Avito Duplicate Ads Detection

Ford Thunderbird, 1974

№ 821490758, размещено сегодня в 17:18 8925 (+24)







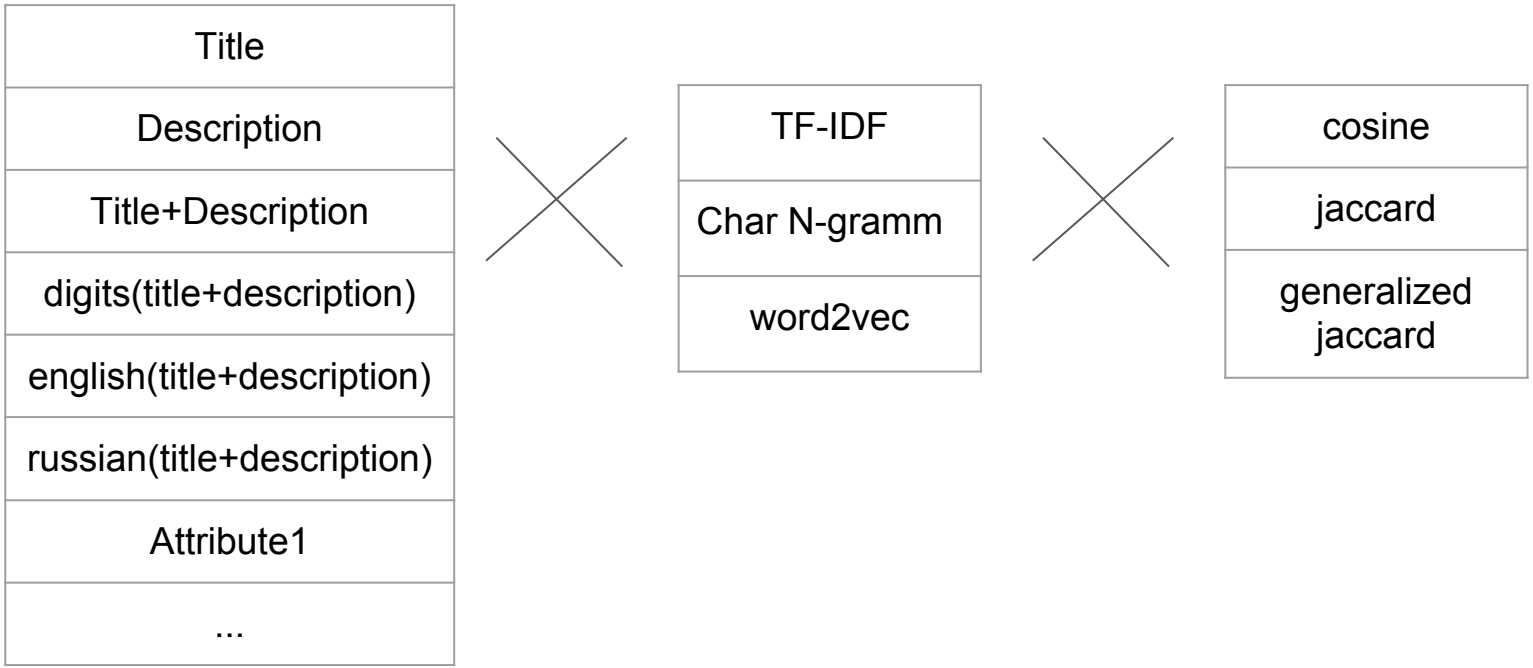

Марка: **Ford**
Модель: **Thunderbird**
Год выпуска: **1974**
Пробег: **110000 км**
Тип кузова: **купе**
Цвет: **пурпурный**
Объём двигателя: **6.0+ л**
Коробка передач: **автомат**

Тип двигателя: **бензин**
Привод: **задний**
Руль: **левый**
Состояние: **не битый**
Владельцев по ПТС: **1**
VIN-номер: **1FALP404*RF****38**
Мощность двигателя: **345 л. с.**

Автомобиль отреставрирован.
Этапы восстановления фиксировались на фото и видео. Много новых комплектующих.
Комплектация:
- Центральный замок
- Кондиционер
- Гидроусилитель руля
- Передние электростеклоподъемники
- Кожаная обивка салона

Пример: Avito Duplicate Ads Detection

Текстовые признаки:



Полезные ссылки

- Bag of Words Meets Bags of Popcorn
- Pymorphy2
- NLTK
- Truly Native? @ Kaggle
- Crowdflower Search Results Relevance