Соревнование 1: предсказание плохих тем на StackOverflow

Евгений Соколов
sokolov.evg@gmail.com

9 Октября 2015

Содержание

Данные

Задача: предсказать, закроют ли тему на StackOverflow.

Признаки:

- ID автора
- Репутация автора
- Число ответов у автора
- Заголовок
- Текст вопроса
- Тэги

Целевая переменная: была ли тема закрыта по одной из причин (оффтопик, неконструктивный вопрос, не вопрос, слишком частный вопрос).

Метрика качества: AUC-ROC.

Правила

Срок: до 1 ноября.

Баллы:

- 1-е место — 15 баллов
- 2-е место — 13 баллов
- 3-е место — 11 баллов
- Остальные места выше бейзлайна — от 1 до 10 баллов по равномерной сетке
- Если решения будут тривиальными, то максимальный балл будет уменьшен

Форма отчетности:

- Код, воспроизводящий решение
- Краткий отчет
- Для занявших первые три места — доклад на семинаре

Leaderboard и форум

Таблица результатов:

- Результаты в таблице вычисляются по 30% тестовой выборки
- Опасно настраиваться на это число, появляется риск переобучения

На странице конкурса есть форум!

- Можно делиться интересными идеями или кодом
- Активность будет поощряться

Какие у нас данные?

В основном у нас есть тексты и категориальные признаки.

- Категориальный признак x_{ij} с областью значений $\{c_1, \ldots, c_m\}$ кодируется бинарным вектором

$$([x^j = c_k])_{k=1}^m.$$

- Текстовый признак x_{ij} — это множество $\{w_1, \ldots, w_{n_i}\}$, который можно закодировать бинарным вектором

$$([w \in x_{ij}])_{w \in W},$$

где $W = \{w_1, \ldots, w_{|W|}\}$ — множество всех возможных слов.

- В обоих случаях получаем разреженные признаки.

Bag-of-words

А как еще можно кодировать тексты?

- Бинарно:
$$([w \in x_{ij}])_{w \in W}.$$
- Счетчики:
$$(n_{dw})_{w \in W},$$
где n_{dw} — число вхождений слова w в документ d.
- TF-IDF:
$$(-n_{dw} \log(|D|/n_w))_{w \in W},$$
где n_w — число документов со словом w, $|D|$ — число документов.

Работа с текстами

Тексты можно предобрабатывать:

- удалять редкие/частотные слова;
- делать стэмминг или лемматизацию;
- **Hashing Trick**: заменить каждое слово w на $h(w)$, где h — хэш-функция с 2^b возможными значениями;
  получаем «кластеризацию» слов, если $2^b < |W|$;
- кластеризовать по-умному: например, объединить все редкие слова в одно;
- строить тематическую модель.

Работа с текстами

Можно генерировать новые признаки:

- индикаторы вхождения всех пар слов:

$$[w_k \in x_{ij}][w_l \in x_{ij}]$$

- n-граммы — индикаторы того, что данные два слова встретились рядом; для текста «мама мыла раму» получаем биграммы «мама мыла» и «мыла раму»
- k-skip-n-граммы — как n-граммы, только разрешаем словам быть отдаленными не больше чем на k

Что делать с разреженными признаками?

- kNN: ввести метрику на текстах
  - слишком большая размерность (десятки тысяч признаков);
  - LSH.
- Решающие деревья: плохо подходят
  - ответ зависит от сложных сочетаний слов;
  - понадобится большая глубина и много данных;
  - легко переобучиться.
- Линейные методы: то, что нужно!
  - легко настраиваются;
  - Vowpal Wabbit — отличная реализация;
  - приспособлены для разреженных признаков;
  - в следующих сериях.

Что еще делать с разреженными признаками?

А если хочется применить kNN, деревья, другие нелинейные методы?

- Понижение размерности:
  - посчитать хэши от признаков

$$h(x; w) = \langle w, x \rangle, \quad w \sim \mathcal{N}(0, 1);$$

  - применить word2vec (уже сегодня!)
  - воспользоваться нейросетями (в следующем семестре)
- Получим небольшое число плотных признаков
- Можно пробовать kNN, деревья, бустинг и т.д.

Blending

Алгоритмы можно объединять в один:

- Пусть даны два алгоритма $b_1(x)$ и $b_2(x)$
- Построим их выпуклую комбинацию:

$$a(x) = \alpha b_1(x) + (1 - \alpha) b_2(x); \quad \alpha \in [0, 1]$$

- Пример: kNN и линейная модель
- Параметр α настраивается с помощью кросс-валидации
- Можно объединять и больше алгоритмов
- Почти всегда решения-победители на Kaggle представляют собой линейную комбинацию нескольких алгоритмов

Feature Engineering

Самое ценное в анализе данных — умение придумывать признаки!

- Подумайте, по каким признакам вы сами стали бы делать классификацию
- Читайте статьи
- Читайте форумы на kaggle

Напутствие

- На слайдах было много идей — экспериментируйте!
- Не забывайте настраивать параметры в ваших алгоритмах
- Изучайте данные, делитесь находками
- Если используете нестандартные алгоритмы или библиотеки — расскажите всем

Удачи!

Содержание

Векторные представления слов

Хотим каждое слово представить как вещественный вектор:

$$w \to \vec{w} \in \mathbb{R}^d$$

Какие требования?

- Размерность d должна быть не очень велика
- Похожие слова должны иметь близкие векторы
- Арифметические операции над векторами должны иметь смысл

word2vec

- Будем обучать представления слов так, чтобы они хорошо предсказывали свой контекст
- Выборка состоит из текстов, каждый представляет собой последовательность слов $w_1, \ldots, w_i, \ldots, w_n$
- Контекст слова w_i: $c(w_i) = (w_{i-k}, \ldots, w_{i-1}, w_{i+1}, \ldots, w_{i+k})$

Функционал для каждого текста:

$$\sum_{i=1}^{n} \sum_{\substack{j=-k \\ j \neq 0}}^{k} \log p(w_{i+j} \mid w_i) \to \max,$$

где вероятность вычисляется через soft-max:

$$p(w_i \mid w_j) = \frac{\exp(\langle \vec{w}_i, \vec{w}_j \rangle)}{\sum_w \exp(\langle \vec{w}, \vec{w}_j \rangle)}$$

(сумма в знаменателе — по всем словам из словаря)

Свойства представлений

- Косинусное расстояние хорошо отражает схожесть слов по тематике (в зависимости от корпуса)
- $\vec{\text{king}} - \vec{\text{man}} + \vec{\text{woman}} \approx \vec{\text{queen}}$
- $\vec{\text{Moscow}} - \vec{\text{Russia}} + \vec{\text{England}} \approx \vec{\text{London}}$
- Перевод: $\vec{\text{one}} - \vec{\text{uno}} + \vec{\text{four}} \approx \vec{\text{quatro}}$
- Среднее представление по всем словам в тексте — хорошее признаковое описание