Отчёт о выполненной работе
Классификация русской Википедии

Максим Дерюгин, 475гр

1 Постановка задачи

Цель работы заключается в исследовнии методов автоматической классификации статей ресурса ru.wikipedia.org с помощбю методов машинного обучения. Схожие задачи были поставлены в рамках LSHTC Challenge и WISE 2014.
Неформальное описание: на основе датасета из статей русской Википедии построить классификатор, присваивающий предлагаемому тексту категории, с помощью которых организована структура Википедии.
Формальное описание: на вход подаётся датасет, содержащий статьи из интернет ресурса ru.wikipedia.org. Датасет имеет следующий вид: каждая статья представлена в виде множества {текст статьи; заголовок статьи; соответствующие статье категории} (другие параметры статей в этой работе не используются). Требуется найти наилуший алгоритм машинного обучения (классификатор), который для предложенного текста определяет категории, наиболее точно описывающие этот тескт. Для проверки качества классификатора используется метод кросс-валидации, в качестве критериев качества были выбраны метрики точность (precision), полнота (recall) и F-мера (F-measure или F1-score).

2 Описание выборки

Прежде чем приступить к приступить к описанию решений, стоит рассказать о различных способах представления пространства объектов, используемых в работе, а также о препроцессинге тесковых данных.
Прежде всего датасет очищается от стоп-слов (общие, "шумные"слова, которые ничего не превносят в классификацию, например предлоги и частицы) и знаков препинания. Далее производится нормализация слов: они приводятся в начальную, их символы - в нижний регистр, например "Список лучших саксафонисток Москвы"→ "список лучший саксафонист москва".
Простейшей моделью для представления текстов на естественном языке является "мешок слов"(bag of words). Такая модель имеет существенный недостаток: самые частотные слова имеют слишком большой вес. Эта проблема решается использованием tf-idf преобразования, которое уменьшает вес слов, встречающихся в большом количестве документов. В данной работе используется реализация TfidfVectorizer с параметрами min_df=5 (минимальная частота слова), max_df=0.98 (2% самых частотных слов отбрасывается) из библиотеки sklearn. Полученную матрицу в дальнейшем будем называть **tf-idf** и она будет основным представлением пространства объектов. Стоит отметить, что данная матрица является очень разреженной. Для рабочего датасета матрица tf-idf имеет размеры (96794, 124767), при этом всего 0.15% элементов являются ненулевыми. Чтобы уменьшить размерность пространства объектов исползуется метод приближения матрицей меньшего ранга с помощью сингулярного разложения. Используемая реализация - метод TruncatedSVD из библиотеки sklearn. В данной работе используется сжатие до размерностей (N, 500) и (N, 1000). Полученные матрицы назовём **svd500** и **svd1000** соответственно.
Отдельно стоит отметить, что в некоторых решениях задачи используются заголовки статей. Они проходят такую-же первоначальную обработку, как и тексты статей (удаление стоп-слов, нормализация). Для их представления используется tf-idf на модели мешка слов и на модели биграмах (частоты последовательностей из двух слов). Получившиеся матрицы будут называться **titles** и **titles-bigrams**. Полезность заголовков при анализе тестов была показана в статье [3].
В целом в работе матрица пространства объектов будет указываться как $\mathbf{X}$, её тренировочная выборка - $\mathbf{X_{train}}$, а тестовая - $\mathbf{X_{test}}$.

3 Базовое решение

Базовым решением является использование метода k ближайших соседей (используется реализации NearestNeighbors библиотеки sklearn, с помощью которой можно легко получить k ближайших соседей), где в качестве множества объектов берётся матрица **tf-idf**, **svd500** или **svd1000**. Основной причной выбора именно этого классификатора является его приемленая ресурсоёмкость относительно используемой памяти и быстродействия. Также преимуществами этого метода являются наличие одно-

го наглядного гиперпараметра и простая настройка работы в виде multilabel-multioutput (когда каждый элемент может иметь одновременно несколько классов и сам классификатор предлагает несколько классов при предсказании).

4 Основное решение

Основное решение заключается в различных модификациях базового, призванных повысить точность классификации. Главным шагом в тут является использование центроидов категорий в качестве дополнительного пространства объектов.

Первой модификацией будет создание аналога классификатора Роккио [2] (он же классификатор k ближайших центроидов для **tf-idf**). Центроид каждой *категории* получается следующим образом:

1. В качестве множества объектов берётся матрица $\mathbf{X}$.
2. Для каждой статьи (строка $\mathbf{X[i]}$ из $\mathbf{X}$) проверяется, есть ли *эта категория* среди категорий статьи. Если есть, то $\mathbf{y[i]} = 1$, иначе $\mathbf{y[i]} = 0$. Тогда полный вектор $\mathbf{y} = (\mathbf{y}[\mathbf{0}], \mathbf{y}[\mathbf{1}], ..., \mathbf{y}[\mathbf{N-1}])^\top$, где N - число статей, - множество ответов.
3. Обучаем классификатор clf = NearestCentroid() из sklearn на $\mathbf{X}$ и $\mathbf{y}$.
4. Из поля clf.centroids_[1] получаем вектор центроида *категории* **class_centroids[j]** (j - номер *категории*).

Таким образом получаем матрицу **class_centroids**, которая имеет размерность (*количество категорий, чилсо признаков из матрицы* $\mathbf{X}$). Таким образом её можно использовать в качестве тренировочного множества $\mathbf{X_{train}}$. Множеством ответов будет матрица единичная матрица размера C×C, где C - количество категорий. Тогда с помощью метода k ближайших соседей, расмотренного в базовом решении, можно найти k ближайших вектору $\mathbf{X_{test}}[\mathbf{i}]$ центроидов, т.е. ближайшие категории для статьи, представленной $\mathbf{X_{test}}[\mathbf{i}]$.

В рамках **второй модификации** были предприняты попытки отфильтровать результаты, полученные в первой модификации. Одним из очевидных спобов это сделать является бинарный классификатор (фильтр), который для пары векторов $\mathbf{X_{test}}[\mathbf{i}]$ и **class_centroids[j]** будет говорить, подходит ли j-тая категория i-той статье. Такой подход был описан[1] одним из участников LSHTC Challenge.
Рассмотрим алгоритм построения такого фильтра:

1. На множестве **class_centroids** обучается модель NearestNeighbors .
2. Составляются матрицы $X_{full} = (X_{left}, X_{right})$ и $\mathbf{y}$. Для каждой статьи :
   (a) берём её представление $\mathbf{X[i]}$ из матрицы $\mathbf{X}$, обозначим его a_i
   (b) с помощью классификатора из пункта 1 k центроидов **C[i1]**, **C[i2]**, ..., **C[ik]**. Обозначим их c_1, c_2, ..., c_k
   (c) определим вектор $\mathbf{y}[i] = (\sigma_1, \sigma_2, ..., \sigma_k)^\top$, где $\sigma_j = 1$, если категория, представленная центроидом c_j, принадлежит статье a_i, и $\sigma_j = 0$ иначе.
   (d) получили

$$X_{left}[i] = \begin{pmatrix} a_i \\ a_i \\ \vdots \\ a_i \end{pmatrix}; \quad X_{right}[i] = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_k \end{pmatrix}; \quad X_{full}[i] = \begin{pmatrix} a_i & c_1 \\ a_i & c_2 \\ \vdots & \vdots \\ a_i & c_k \end{pmatrix}; \quad \mathbf{y}[i] = \begin{pmatrix} \sigma_1 \\ \sigma_2 \\ \vdots \\ \sigma_k \end{pmatrix}$$

В итоге:

$$X_{full} = \begin{pmatrix} X_{full}[1] \\ X_{full}[2] \\ \vdots \\ X_{full}[N] \end{pmatrix}; \quad \mathbf{y} = \begin{pmatrix} \mathbf{y}[1] \\ \mathbf{y}[2] \\ \vdots \\ \mathbf{y}[N] \end{pmatrix}$$

где N - количество статей.

3. На основе полученных X_{full} и $\mathbf{y}$ тренируем модель Logistic Regression и получаем необходимый нам фильтр.

Чтобы профильтровать реузльтаты, полученные в первой модификации, нужно построить аналогичную матрицу $X_{full} = (X_{left}, X_{right})$, где X_{left} будет состоять из векторов $X_{test}[i]$, а X_{right} - из центроидов полученных категорий.

Особенность этой модификации заключается в том, что есть возможность совмещать различные признаки в фильтре. В качестве основной X_{left} выборки можно брать, например, матрицу **tf-idf**, а в качестве центроидов и X_{right} - матрицу центроидов, построенных по **titles** (будут обозначаться как **titles centroids**). В работе рассмотриваются следующие комбинации (X по A $\Leftrightarrow$ матрица X составлена по матрице A согласно пункту 2 агоритма):

1. X_{left} по **tf-idf**, X_{right} по **tf-idf centroids**
2. X_{left} по **tf-idf**, X_{right} по **titles centroids**
3. X_{left} по **tf-idf**, X_{right} по **titles-bigrams centroids**

5 Эксперименты

Ниже приведены экспериментальные результаты измерения точностей приведённых выше решений. Для оценки решений при различных гиперпараметрых используется кросс-валидация с 3 различными разбиениями на множества *train* и *test*. В данной работе использовался датасет из 96794 статей, содержащих 14697 категорий.

5.1 Сравнение базового решения с первой модификацией, основанной на центроидах

В данном случае гиперпараметром является k - количество соседей модели k ближайших соседей.

k	X_{train}	X_{test}	precision-score	recall-score	f1-score
5	tf-idf	tf-idf	0.20	0.30	0.24
10	tf-idf	tf-idf	0.17	0.31	0.21
5	tf-idf centroids	tf-idf	0.41	0.88	0.56
10	tf-idf centroids	tf-idf	0.24	0.94	0.38
5	svd500	svd500	0.28	0.56	0.37
10	svd500	svd500	0.23	0.63	0.34
5	svd500 centroids	svd500	0.24	0.60	0.35
10	svd500 centroids	svd500	0.23	0.63	0.34
5	svd1000	svd1000	0.29	0.56	0.38
10	svd1000	svd1000	0.24	0.64	0.35
5	svd1000 centroids	svd1000	0.27	0.65	0.39
10	svd1000 centroids	svd1000	0.23	0.63	0.34

5.2 Применеие бинарной фильрации

Прежде всего оценим точность самого фильтра при различных значениях параметра C логистической регрессии. Здесь $\frac{\sum y}{len(y)}$ - доля правильно предсказанных категорий с помощью модели из пункта 1 второй модификации, ко всей длине получившегося вектора **y**.

1. tf-idf + tf-idf centroids

C	$accuracy_{\ test}$	$accuracy_{\ train}$	$\frac{\sum y}{len(y)}$
0.01	0.62	0.69	0.41
0.1	0.66	0.75	
1.0	0.64	0.80	
10.0	0.64	0.84	

2. tf-idf + titles centroids

C	$accuracy_{\ test}$	$accuracy_{\ train}$	$\frac{\sum y}{len(y)}$
0.1	0.66	0.74	0.10
1.0	0.72	0.81	
5.0	0.74	0.84	
10.0	0.74	0.85	
20.0	0.74	0.87	

3. tf-idf + titles-bigrams centroids

C	$accuracy_{\ test}$	$accuracy_{\ train}$	$\frac{\sum y}{len(y)}$
0.1	0.72	0.77	0.34
1.0	0.73	0.87	
5.0	0.78	0.94	
10.0	0.79	0.96	
20.0	0.80	0.97	

4. svd500 + svd500 centroids

C	$accuracy_{\ test}$	$accuracy_{\ train}$	$\frac{\sum y}{len(y)}$
0.1	0.60	0.63	0.24
1.0	0.60	0.64	
5.0	0.61	0.65	
10.0	0.61	0.65	
20.0	0.61	0.65	

Применим теперь полученные фильтры к результатам из пункта 5.1.

k	X_{left}	X_{right}	precision-score	recall-score	f1-score
5	tf-idf	tf-idf centroids	0.28	0.39	0.32
10	tf-idf	tf-idf centroids	0.17	0.43	0.26
5	tf-idf	titles centroids	0.14	0.15	0.14
10	tf-idf	titles centroids	0.10	0.16	0.12
5	tf-idf	titles-bigrams centroids	0.26	0.30	0.28
10	tf-idf	titles-bigrams centroids	0.17	0.34	0.22

6 Выводы

Как можно видеть, наилучшие результаты были получены при совмещении признаков в виде **tf-idf** и полученных на них центроидах. Благодаря такому подходу получилось значительно улучшить полноту получаемых классификатором данные, то есть класификатор склонен предлагать большинство

правильных ответов. К сожалению, идея с фильтрацией оказалась нежизнеспособной, по крайней мере при данной реализации. Не удалось достичь даже основной цели этой модификации: улучшить точность ответа классификатора. В качетсве альтернативной реализации можно предложить поклассовую бинарную классификацию: для каждой категории обучатся бинарный классификатор (можно также взять логистическую регрессию или воспользоваться методом опорных векторов) на всём корпусе или некоторй его части. Делается это аналогично построению центроидов, как было описано в этой работе. Также стоит отметить, что уменьшение размерности с помощью сингулярного разложения также только ухудшило результаты классификации. Здесь проблема может заключаться в том, что было произведено слишком сильное сжатие. Скорее всего при уменьшении количества признаков документа с изначальных 124 767 до порядка 10 000 не привело бы к ухудшению качества выборки. Также не удалось найти улучшающее классификацию применение заголовков. Темне менее, метод, ипользующий комбинацию **tf-idf** и **tf-idf centroids**, показал хороший результат, то есть первую модификацию базового алгоритма можно назвать успешной.

7 Ссылки на репозиторий и рабочие файлы

1. Репозиторий с решением
2. Подготовка датасета
3. Вычисление центроидов
4. Обучение бинарного классификатора (фильтра)
5. Обучение различных модификаций kNN

Список литературы

[1] *Code for Large Scale Hierarchical Text Classification competition*. URL: `https://github.com/nagadomi/kaggle-lshtc`.

[2] *Nearest centroid classifier*. URL: `https://en.wikipedia.org/wiki/Nearest_centroid_classifier`.

[3] Péter Schönhofen. "Identifying Document Topics Using the Wikipedia Category Network". в: *Research Gate* (2012). DOI: `10.3233/WIA-2009-0162`. URL: `https://www.researchgate.net/publication/233748512`.