Лабораторная работа «Распознавание эмоциональной окраски текстовый сообщений»

Цель работы: научиться применять метод байесовской классификации для распознавания эмоциональной окраски текстовых сообщений

1. Теория

Sentiment analysis (анализ тональности) — это область компьютерной лингвистики, которая занимается изучением мнений и эмоций в текстовых документах.
Целью *анализа тональности* является нахождение мнений в тексте и определение их свойств. В зависимости от поставленной задачи нас могут интересовать разные свойства, например:

1. автор — кому принадлежит это мнение
2. тема — о чем говорится во мнении
3. тональность — позиция автора относительно упомянутой темы (обычно «положительная» или «отрицательная»)

Пример: "Главный итог завершившихся Игр XXX Олимпиады в Лондоне — то чувство гордости за нашу страну, которое испытывали болельщики благодаря выступлениям российских олимпийцев», — считает Александр Жуков"
автор: Александр Жуков
тема: "выступление российских олимпийцев"
тональность: «положительная»

Анализ тональности находит свое практическое применение в разных областях:

1. социология — собрать данные из соц. сетей (например, о религиозных взглядах)
2. политология — собрать данные из блогов о политических взглядах населения
3. маркетинг — проанализировать Твиттер, чтобы узнать какая модель ноутбуков пользуется наибольшим спросом
4. медицина и психология — определить уровень депрессии у пользователей соц. сетей.

Задача анализа эмоциональной окраски текстов сводится к задаче классификации. В нашем случае имеется набор текстов, каждый из которых нужно отнести к одной из двух категорий: положительные или отрицательные.

Подготовка текстов к анализу (Bag of words)

Для создания классификатора текстов данные необходимо предобработать. В качестве этапов предобработки текстов могу выступать:

1. Получение «чистого текста» - удаление html-тегов, знаков пунктуации.

2. Удаление стоп-слов, которые не несут смысловой нагрузки на текст (это предлоги, союзы, артикли в иностранных текстах).

3. Приведение слов в тексте к нормальной форме - стемминг.

В рамках данной лабораторной работы тексты используются как есть, без дополнительных этапов предобработки.

Для обучения классификатора необходимо представить текст в виде вектора из чисел. Для этого можно составить словарь со всеми словами, т.е. объединить все слова встречающийся в текстах в один большой словарь, или же использовать готовые словари (Даля или Зализняка), и заменить слова из текста индексом в словаре. То есть допустим мы имеем всего три коротких сообщения со следующими векторами слов:

1 [biography, part, feature]
2 [film, remember, going]
3 [see, cinema, originally]

Объединяя все слова из списка в один мы получим следующий отсортированный словарь (назовем его как базис вектор):

[biography, cinema, feature, film, going, originally, part, remember, see]

Заменяя предыдущие вектора на индекс слова в словаре получаем следующее:
1 [1, 0, 1, 0, 0, 0, 1, 0, 0]
2 [0, 0, 0, 1, 1, 0, 0, 1, 0]
3 [0, 1, 0, 0, 1, 0, 0, 0, 1]

Проделав такую работу для всех текстов мы можем получить достаточно большой список. Эти вектора называют «векторами свойств» или же «features vector». Таким образом, мы получаем вектора для каждого текста и дальше можем использовать в качестве атрибутов, которые описывают текст, и на основе которых классификатор вырабатывает правила классфикации. Данный подход называется «мешок слов» или же “Bag-Of-Words”.

Наивный байесовский классификатор

В качестве метода классификации в работе предлагается использовать наивный байесовский классификатор (НБК). Наивный байесовский классификатор работает с условными вероятностями, наивно предполагая, что слова в предложении независимы. Этот простой классификатор хорошо показывает себя в решении задачи классификации текстов.

Байесовские алгоритмы классификации основаны на формуле Байеса (1) и принципа максимума апостериорный вероятности c*.

$$c^* = \arg\max_c \frac{P(c) \sum_{i=1}^{m} P(x|c)^{x_i(t)}}{P(t)} \quad (1)$$

где *P(c)* - априорная вероятность того, что выпадет класс *c; P(x|c)* - апостериорная вероятность того, что при выпадении класса *c* будет объект (слово) *x; $x_i(t)$* - закон распределения исходных данных.

В зависимости от закона распределения имеются различные реализации наивного байесовского классификатора, которые различаются расчетом вероятности:

1) Гауссовский закон распределения - используется для данных, которые являются непрерывными величинами;

2) Мультиномиальный закон распределния - используется для данных, являющихся дискретными величинами;

3) Закон Бернулли - используется для бинарных данных.

Основные преимущества *наивного байесовского классификатора* — простота реализации и низкие вычислительные затраты при обучении и классификации. В тех редких случаях, когда признаки действительно независимы (или почти независимы), наивный байесовский классификатор (почти) оптимален.

2. Исходные данные

Массив 1 «Сообщения твиттера»
Массив данных состоит из 2 файлов (твиты с положительной окраской и отрицательной). Каждый файл следующего формата:
– id: уникальный номер сообщения в системе twitter;
– tdate: дата публикации сообщения (твита);
– tmane: имя пользователя, опубликовавшего сообщение;
– ttext: текст сообщения (твита);
– ttype: поле в котором в дальнейшем будет указано к кому классу относится твит (положительный, отрицательный, нейтральный);
– trep: количество реплаев к данному сообщению. В настоящий момент API твиттера не отдает эту информацию;
– tfav: число сколько раз данное сообщение было добавлено в избранное другими пользователями;
– tstcount: число всех сообщений пользователя в сети twitter;
– tfol: количество фоловеров пользователя (тех людей, которые читают пользователя);
– tfrien: количество друзей пользователя (те люди, которых читает пользователь);
– listcount: количество листов-подписок в которые добавлен твиттер-пользователь.

В работе используются только поля: ttext и ttype.

Массив 2 «Отзывы о фильмах»
Массив данных состоит из двух папок pos и neg (положительная и отрицательная окраска), где хранятся в исходном виде в отдельных файлах англоязычные отзывы на фильмы.

4. Инструменты

NumPy

NumPy - это библиотека языка Python, добавляющая поддержку больших многомерных массивов и матриц, вместе с большой библиотекой высокоуровневых (и очень быстрых) математических функций для операций с этими массивами.

Основным объектом при работе с numpy является массив, который можно создать следующим образом:

```
>>> import numpy as np
>>> a = np.array([1, 2, 3])
>>> a
array([1, 2, 3])
```

Также numpy предоставляет средства для загрузки данных непосредственно из csv-файла:

```
from numpy import genfromtxt
my_data = genfromtxt('my_file.csv', delimiter=';')
```

Одной из часто используемых операций при работе с многомерными массивами, является индексирование и взятие срезов. Например, для двухмерного массива:

```
>>> b = np.array([[  0, 1, 2, 3],
...               [10, 11, 12, 13],
...               [20, 21, 22, 23],
...               [30, 31, 32, 33],
...               [40, 41, 42, 43]])
...
>>> b[2,3]  # Вторая строка, третий столбец
23
>>> b[(2,3)]
23
>>> b[2][3]  # Можно и так
23
>>> b[:,2]  # Третий столбец
array([ 2, 12, 22, 32, 42])
>>> b[:2]  # Первые две строки
array([[ 0,  1,  2,  3],
       [10, 11, 12, 13]])
>>> b[1:3, : : ]  # Вторая и третья строки
array([[10, 11, 12, 13],
       [20, 21, 22, 23]])
```

Scikit-learn

Scikit-learn - это одна из самых популярных библиотек, используемых для анализа данных, и написанная с использованием python.

В рамках выполнения данной лабораторной работы вам понадобятся следующие пакеты и методы:

CountVectorizer - для преобразования текста в массив с подсчитанным количеством слов.

Пример использования:

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(analyzer = "word",
                             tokenizer = None,
                             preprocessor = None,
                             stop_words = None,
                             max_features = 3000)
train_data_features = vectorizer.fit_transform(text)
```

```
train_data_features = train_data_features.toarray()
```
В результате в train_data_features находится массив с количеством слов.

train_test_split - метод для разделения произвольного массива данных на обучающую и тестовую выборку. Массив данных при этом перемешивается.
Пример использования:
```
from sklearn.cross_validation import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(train_data_features, Y, test_size=0.2, random_state=0)
```
В параметре test_size указывается относительная доля тестовой выборки в исходном массиве данных.

Байесовский классификатор - реализация нескольких видов наивного байесовского классификатора в зависимости от закона распределения:
- BernoulliNB
- GaussianNB
- MultinomialNB

Пример использования:
```
from sklearn.naive_bayes import BernoulliNB, MultinomialNB, GaussianNB
clf = MultinomialNB()
clf.fit(X_train, Y_train)
res = clf.predict(X_test)
```
Метод fit используется для построения модели, где X_train, Y_train - обучающая выборка. Метод predict используется для прогнозирования значений классификатора по тестовой выборке X_test.

accuracy_score - метод для оценки качества классификации.
Пример использования:
```
from sklearn import metrics
print metrics.accuracy_score(Y_test, res_clf)
```
Y_test - реальные значения тестовой выборки, res_clf - результат, полученный чс помощью классификатора.

5. Ход работы

1. Изучить теоретический материал по анализу тональности текстов и байесовской классификации.
2. Изучить структуру исходного массива данных, которые находятся в папке data. Описать структуру исходных данных: классы, количество текстовых сообщений в каждом классе.
3. Создайте пустой проект в PyCharm CE. Напишите метод для загрузки данных из файлов (из CSV - для твитов, из текстовых файлов - для отзывов на фильмы). Объедините массивы положительных и отрицательных текстов в единый исходный массив данных.
4. Реализуйте преобразование исходных текстов сообщений в вектор атрибутов с помощью подхода «Bag of Words». Для реализации используйте класс CountVectorizer библиотеки Scikit-Learn.
5. Согласно вашему варианту разделите исходный массив данных на обучающую и тестовую выборку. Например: 70% текстов - обучающая

выборка, 30% текстов - тестовая выборка. Для разделения используйте метод train_test_split библиотеки Scikit-Learn.

6. Реализуйте эксперименты с учетом параметров вашего варианта (различные виды БК и различные значения параметра max_features для CountVectorizer). Для каждого эксперимента посчитайте точность получившегося классификатора. Для расчета точности используйте метод accuracy_score библиотеки Scikit-Learn.
7. По полученным результатам сделайте выводы: какого вида БК и какое количество признаков наилучшим образом влияет на точность классификатора.

6. Варианты

Вариант	Исходный массив даных	Виды БК	max_features	Обучающая/ тестовая выборка
1	Твиты	BernoulliNB, MultinomialNB	3000;5000;7000	70/30
2	Отзывы	MultinomialNB, GaussianNB	4000;6000;8000	80/20
3	Твиты	BernoulliNB, GaussianNB	5000;7000;9000	90/10
4	Отзывы	BernoulliNB, MultinomialNB	6000;8000;10000	70/30
5	Твиты	MultinomialNB, GaussianNB	3000;5000;7000	80/20
6	Отзывы	BernoulliNB, GaussianNB	4000;6000;8000	90/10
7	Твиты	BernoulliNB, MultinomialNB	5000;7000;9000	70/30
8	Отзывы	MultinomialNB, GaussianNB	6000;8000;10000	80/20
9	Твиты	BernoulliNB, GaussianNB	3000;5000;7000	90/10
10	Отзывы	BernoulliNB, MultinomialNB	4000;6000;8000	70/30
11	Твиты	MultinomialNB, GaussianNB	5000;7000;9000	80/20
12	Отзывы	BernoulliNB, GaussianNB	6000;8000;10000	90/10
13	Твиты	BernoulliNB, MultinomialNB	3000;5000;7000	70/30
14	Отзывы	MultinomialNB, GaussianNB	4000;6000;8000	80/20
15	Твиты	BernoulliNB, GaussianNB	5000;7000;9000	90/10

7. Структура отчета

1. Тит. лист
2. Описание исходных данных
3. Описание схемы экспериментов
4. Результаты экспериментов
5. Исходный код

8. Вопросы на защиту

1. Понятие анализа тональности текстов.
2. В чем заключается подготовка текста к анализу?
3. Априорные и апостериорный вероятности.
4. Наивный байесовский классификатор.
5. Достоинства и недостатки наивного байесовского классификатора.

9. Список литературы

1. Реализация НБК в skLearn http://scikit-learn.org/stable/modules/naive_bayes.html
2. Описание байесовских алгоритмов http://www.machinelearning.ru/wiki/index.php?title=Байесовский_классификатор
3. Лекции по байесовским методам классификации http://www.ccas.ru/voron/download/Bayes.pdf