Московский Государственный Университет
им. М.В. Ломоносова

Определение авторства текстов методом опорных векторов(SVM)

Отчёт по спецкурсу «Математические методы анализа текста»

Студент: Косырев С.А
Группа: ВВО 3 курс
Факультет: ВМК
Преподаватель:
Dr Масленников М.В

Москва, апрель 2013 г.

Мотивация

В этой работе исследуется проблема идентификации авторства коротких текстов. Такие тексты активно используются в повседневной жизни в различных интернет-мессенджерах, электронной почте, интернет-форумах, блогах и.т.д. Пользователи имеют возможность отправлять сообщения анонимно, либо регистрационные данные не позволяют однозначно идентифицировать личность собеседника. При этом проблема идентификации авторства коротких текстов исследована менее подробно, чем проблема с длинными текстами.

Метод решения

Проблему идентификации авторства текста сформулируем следующим образом: имеется множество авторов $A=\{a_1,\dots,a_n\}$ и множество текстов $T=\{t_1,\dots,t_k\}$ с известным авторством. Необходимо установить, кто из множества A является автором остальных текстов из множества T'.

В этой работе проблема идентификации сводится к задаче классификации с несколькими классами. Пусть множество A составляет множество классов, T – обучающие примеры, а T' – классифицируемые объекты. Целью является построение классификатора, решающего данную задачу, т.е. нахождение некоторой функции F, относящей произвольный текст множества T' к его истинному автору из A.

Стратегии выбора решения:

- *«Один против остальных»(one-vs-the-rest):* Для решения задачи строится n классификаторов таким образом, что каждый класс a_i сопоставляется с остальными $(n-1)$ классами, т.е. в каждом из i случаев выбор осуществляется из двух вариантов: «класс a_i» и «не класс a_i». Решение по классам принимается по схеме «победитель забирает всё» (winner takes all) – победителем считается класс, имеющий максимальное значение функции F.

- *«Каждый против каждого»(one-vs-one):* Классификаторы строятся для каждой пары классов для того, чтобы можно было однозначно разделить любые два класса из множества A. Количество классификаторов в этом случае равно $n\frac{(n-1)}{2}$. После подачи на вход каждого из классификаторов тестового образца получаем ответы, содержащие информацию о его соответствии одному из двух классов. К полученному множеству ответов применяется схема мажоритарного голосования и класс, выбранный большинством классификаторов, принимается как итоговое решение.

В исследовании используется классификатор на основе метода опорных векторов (Support Vector Machine, SVM) в реализации scikit-learn[4]. SVM строит линейный классификатор в пространстве признаков с высокой размерностью таким образом, чтобы зазор между граничными точками двух классов, называемых опорными векторами, был максимальным.

На сегодняшний день SVM является одним из лучших методов классификации. Преимущества и недостатки SVM[5]:

- Наиболее быстрый метод нахождения решающих функций;
- Метод сводится к решению задачи квадратичного программирования в выпуклой области, которая всегда имеет единственное решение;
- Метод находит разделяющую полосу максимальной ширины, что позволяет в дальнейшем осуществлять более уверенную классификацию;
- Метод чувствителен к шумам и стандартизации данных;
- Не существует общего подхода к автоматическому выбору ядра (и построению спрямляющего подпространства в целом) в случае линейной неразделимости классов.

Общая информация и исходные данные

Данные для исследования взяты с публично доступной части форума[1]. Сообщения выбранных пользователей хранятся в текстовых файлах вида `username.txt`, в качестве разделителя сообщений используется символ `'\n'`.

Для исследования было выбрано 5 авторов:

Автор	Количество сообщений	Сообщений в обучающей выборке	Сообщений в тестовой выборке	Средняя длина сообщения, симв.
Apparition	3656	3270	386	123.0
Hayate	3784	3421	363	416.0
Next	7674	6914	760	565.4
Vermillion	4549	4058	491	305.2
ZloeAloe	6167	5545	622	584.3
Всего:	25830	23208	2622	398.78

В проекте применен модуль scikit-learn[4].

- `clf = svm.LinearSVC(params)` – задание параметров классификатора.
- clf.fit(X, Y) – обучение классификатора.
  X — набор вектор-признаков, Y — набор соответствующих им классов.
- clf.predict(X) – определить класс вектора X.
- `clf.score(X, Y)` – проверить классификатор по набору векторов X и соотвествующих им классов Y.
- C - параметр регуляризации, отвечающий за соотношение между величиной зазора и количеством ошибок обучающего множества.

Вектор-признак сообщения формируется следующим образом:

1. Формируется словарь(`globaldict.txt` – *см. Приложение 1*) наиболее часто употребляемых каждым пользователем слов по всей обучающей выборке. Структура словаря:

```
{ UserName1: [sorted_list_of_K_top_used_words] ,…,
  UserNameN: [sorted_list_of_K_top_used_words]                    }
```

Таким образом, размерность вектора: N*K. где K = `vectorSize`.

2. Для каждого сообщения определяется список использованных в нем слов и их количество.
3. Если слово входит в словарь из п.1, количество его вхождений ставится на соответствующее место из словая.

Подробнее алгоритм: *`toolkit.py`* → `class Vectorizer` *(см. Приложение 1)*

По результатам исследований

- **Лучше всего работают классификаторы с линейным ядром:**

Сравнение различных классификаторов[4] *для 5 авторов:*

LinearSVC: Алгоритм использует SVC с линейным ядром и построен на библиотеке *liblinear.* Также, он обеспечивает настройку параметров loss и penalty, влияющих на штрафы за ошибки классификатора.

```
svm.LinearSVC(C=0.2, class_weight = 'auto')
```

SGDClassifier: Классификатор, работающий по методу стохастического градиентного спуска(*Stochastic Gradient Descent*).

```
linear_model.SGDClassifier(class_weight = 'auto', fit_intercept=False, n_iter=100, shuffle=True, n_jobs=-1)
```

SVC: Базовый классификатор на основе библиотеки *libsvm.* Он поддерживает параметр регуляризации (*C*), влияющий на величину зазора между гиперплоскостями.

```
svm.SVC(C=0.2, class_weight='auto', kernel='linear', cache_size=1000)
```

NuSVC: Аналогичен SVC, и также обеспечивает контроль количества опорных векторов(*Nu* — доля опорных векторов в обучающей выборке).

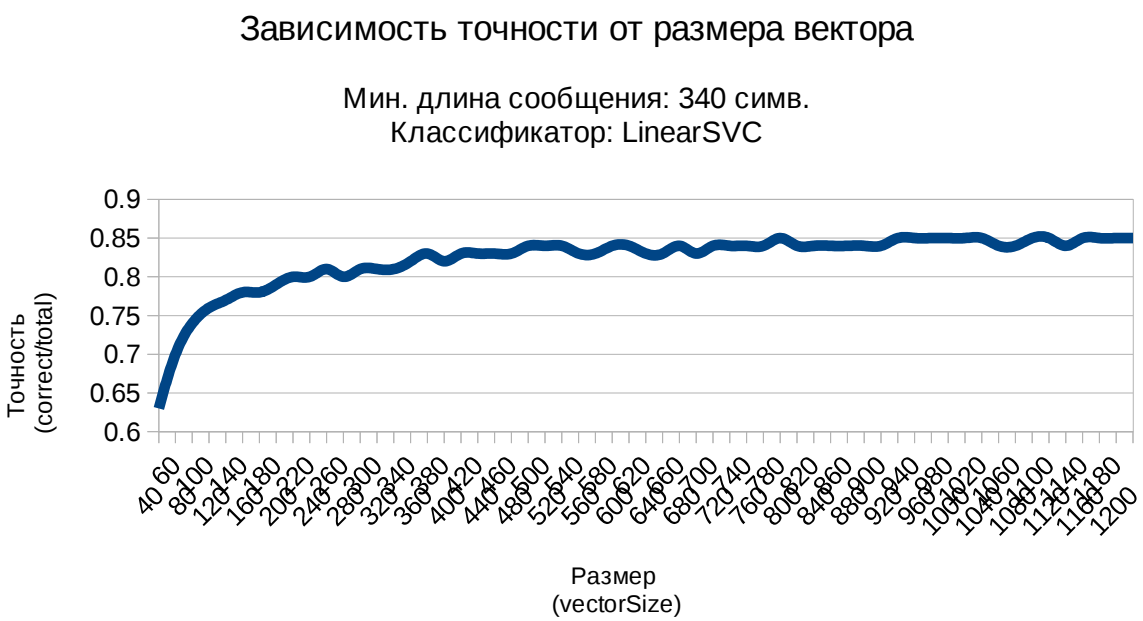
```
svm.NuSVC(nu=0.02, kernel='linear', cache_size=1000)
```

Стратегия выбора	**Классификатор**	**Score**
«one-vs-the-rest»	LinearSVC	**0.85**
«one-vs-the-rest»	SGDClassifier	**0.82**
«one-vs-one»	SVC	**0.83**
«one-vs-one»	NuSVC	**0.81**

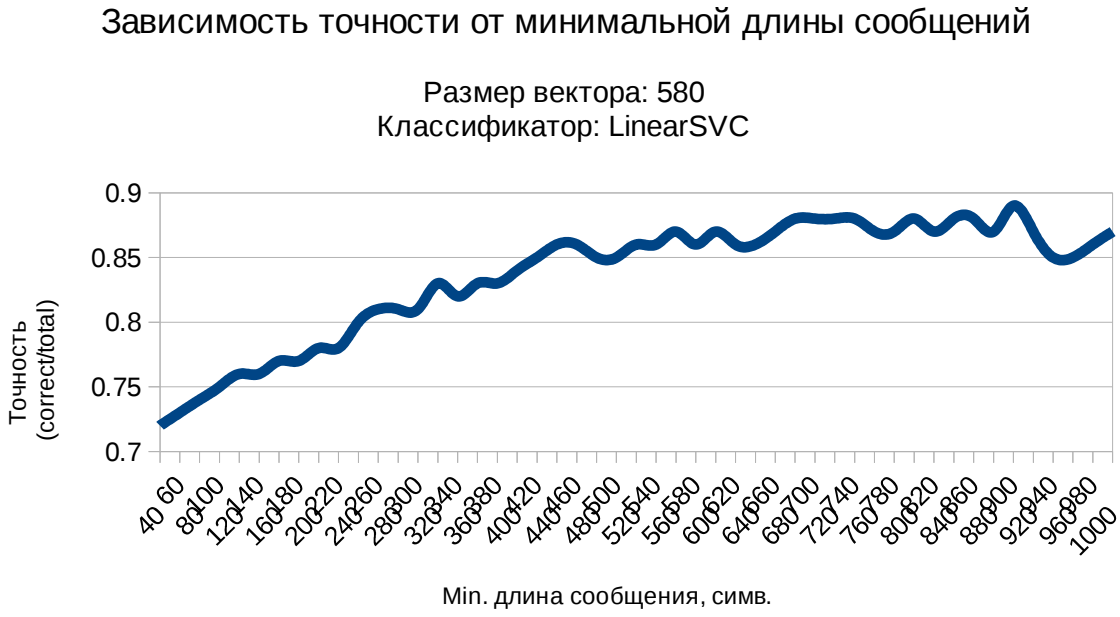
Параметры:

- Размер вектора: 780
- Мин. длина сообщения: 440
- C: 0.2

- **Точность распознавания зависит от размерности вектора:**



- **Значительное влияние на точность оказывает ограничение на минимальную длину сообщения.**
  Чем короче сообщения, тем труднее определить авторство:



Пример:

Author	Score
Apparition	0.69
Hayate	0.74
Next	0.86
Vermillion	0.81
ZloeAloe	0.9
Total score	0.85

Параметры:

- Размер вектора: 780
- Мин. длина сообщения: 440

У автора Apparition(длина сообщения – в среднем 123 симв.) высокая доля коротких сообщений(флуд), что является причиной низкой точности распознавания. И наоборот, автор ZloeAloe(584 симв.) - идентифицируется точнее всех. Чем более «осмысленны» сообщения, тем проще определять их авторство.

Использованные материалы

1. Подопытный форум: http://forum.killmepls.ru/
2. Модуль для парсинга HTML: http://lxml.de/
3. Python 2.7.4: http://docs.python.org/2.7/ https://developers.google.com/edu/python/
4. Модуль SVM для Python: http://scikit-learn.org/stable/modules/svm.html
5. Метод SVM http://www.machinelearning.ru/wiki/index.php?title=SVM

Приложение 1: Состав проекта

db_parser.py	– Парсер сообщений. Сохраняет текст в ./parsed_data/
db_extractor.py	– Для разбиения базы сообщений на обучающую и контрольную выборки. Так же в ./parsed_data/
toolkit.py	– Набор общих функций и классов
classifier.py	– Обучение классификатора
tester.py	– Проверка классификатора по контрольной выборке
brute.sh	– Bash-скрипты для подбора оптимальных параметров
./classifiers/	– Сохранённые обученные классификаторы
./db/	– Обучающая выборка
./test_db/	– Контрольная выборка
./params/classmap.txt	– Файл с сопоставленными именами авторов и номерами классов
./params/globaldict.txt	– Словарь наиболее популярных слов по авторам из обучающей выборки.
./params/users.txt	– Список авторов для парсинга