Практическое задание №2 по курсу “Обработка текстов”. Осень 2015

Постановка задачи

Целью задания является создание системы, позволяющей определять возраст пользователей социальной сети Вконтакте по текстам их комментариев

Система должна классифицировать пользователей в один из 5 заданных возрастных интервалов:

- <=18 лет (обозначение «00-18»)
- 19-25 лет (обозначение «19-25»)
- 26-35 лет (обозначение «26-35»)
- 36-45 лет (обозначение «36-45»)
- >=45 лет (обозначение «45-99»)

Вход:
список текстов комментариев пользователей
Выход:
возрастной интервал

Решение задачи

Практические аспекты

Решения должны быть написаны на языке Python 3.5. Можно использовать все стандартные библиотеки, а также:

- NLTK - инструменты для обработки текстов
- scikit-learn - алгоритмы машинного обучения
- numpy - работа с многомерными массивами
-

Доступ в Интернет на проверяющей машине закрыт. По требованию может быть предоставлен доступ к https://api.ispras.ru

Теоретические аспекты

Предполагается использование алгоритмов машинного обучения. Для обучения алгоритма требуется придумать признаки и дать ему на вход правильные примеры - обучающий корпус.

Тренировочный корпус

Тренировочный корпус доступен для скачивания в формате json. Для чтения информации из этого файла рекомендуется использовать стандартную библиотеку json.
Тренировочный корпус состоит из двух файлов:

- ***Train.txt.json*** содержит список (в json формате) примеров. Каждый пример – это список текстов (строк). Таким образом, данный файл содержит список списков строк.
  Например:
  [["текст пользователя 1", "еще один текст пользователя 1"],["пользователь 2","написал","3 сообщения"],["сообщения","пользователя 3"]]

- ***Train.lab.json*** содержит список (в json формате) соответствующих каждому пользователю значений возрастного интервала.
  Например:
  ["45-99", "00-18", "26-35"]

Требования к решению

Загружаемый файл должен представлять собой zip архив с любым именем. Архив должен обязательно содержать:

- Файл a*ge_detector.py*. В нем должен содержаться класс AgeDetector. В классе должны присутствовать методы:
  - **train**(self, instances, labels). На вход подается список примеров (список списков строк, instances) и список соответствующих значений возрастного интервала (labels). Метод ничего не возвращает.
  - **classify**(self, instances). На вход подается список примеров (список списков строк, instances). Метод должен вернуть список, содержащий строковые значения возрастных интервалов, соответствующие пользователям из instances.
- (Пустой) файл *__init__.py* (Требования к пакетам Python)
- Описание применяемых алгоритмов в файле *description.txt*
- Все используемые внешние библиотеки, кроме библиотек пакета NTLK, scikit-learn и numpy (они доступны автоматически)

Все файлы должны быть в кодировке UTF-8.
Формат файлов тренировочного корпуса соответствует спецификации методов train и classify.

Проверка решения

Результаты тестирования появятся на личной странице, как только закончится обучение и тестирование. При загрузке нового классификатора обучение будет производиться на корпусе, размеченном автором классификатора, плюс все дискуссии, размеченные в течение предшествующей загрузке недели.
В течение недели студенты не видят прогресс своих коллег и могут посмотреть только свой результат. В конце каждой недели (каждый вторник в 23.59.59) будет производиться переобучение последнего присланного решения. Результаты тестирования будут показаны в сводной таблице.

Ограничения

1. каждую неделю можно послать только 10 версий программы (внимание! Итоговое тестирование будет проводиться на последнем загруженном решении)
2. размер архива не может превышать 15Мб

В связи с первым ограничением, для тестирования на локальной машине рекомендуется использовать метод перекрестной проверки (http://en.wikipedia.org/wiki/Cross-validation_(statistics)). В библиотеке scikit-learn есть функции, которые могут помочь в использовании этого метода (например, Kfold()).

Оценка качества
Для оценки качества используются мера accuracy. Описание в документации к библиотеке http://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html

Baseline

Baseline 1. В качестве классификатора используется наивный байесовский классификатор. В качестве признаков используются униграммы слов.
Baseline 2. Еще одно решение

Подсчет очков

Как и для первого задания, в конце каждой недели вы сможете посмотреть, насколько хороший классификатор вы сделали по сравнению с другими предложенными решениями. Эти результаты нужны только для понимания текущей ситуации. Окончательный подсчет очков будет произведен при наступлении второго дедлайна.