Практическое задание для студентов 4 курса кафедры СП. Осень 2017

Постановка задачи

Целью работы является создание алгоритмов, определяющих возраст и образование людей по текстам, которые они пишут.

Предлагается разработать и реализовать два алгоритма:

- Определение возраста по тексту. На вход подается список текстов, принадлежащих одному автору. Выходом алгоритма должен быть один из классов: "<=17","18-24", "25-34", "35-44", ">=45".
- Определения образования по тексту. Вход такой же, как в предыдущей задаче. Выход один из классов: "lower", "middle", "high".

Решение задачи

Практические аспекты

Решения проверяются удаленно. Интерфейс автоматической системы находится по адресу: https://tpc2017.at.ispras.ru.

Решения должны быть написаны на языке Python (версия 3.6.3). Можно использовать все стандартные библиотеки, а также

- NLTK - инструменты для обработки текстов
- scikit-learn - алгоритмы машинного обучения
- numpy - работа с многомерными массивами
- pytorch, keras - библиотеки для работы с искусственными нейронными сетями
- fasttext - библиотека для работы с векторными представлениями слов, полученными одноименным алгоритмом

Также можно использовать модели векторного представления слов. Для экспериментов и тренировки алгоритмов модели можно скачать по адресу:

http://tpc.at.ispras.ru/dopolnitelnye-materialy/

Для тестирования решения эти модели присылать не нужно, они уже есть на проверяющей машине в папке, куда будет распаковано ваше решение. Использовать можно, например, так:

```
model100 = fasttext.load_model('fasttext100.bin')
```

При этом не надо добавлять в архив решения файлы с такими именами, потому что они перетрут модели.

Доступ в Интернет на проверяющей машине закрыт.

Теоретические аспекты

Предполагается использование алгоритмов машинного обучения. Для обучения алгоритма требуется придумать признаки и дать ему на вход правильные примеры - обучающий корпус. В качестве обучающего корпуса дается множество текстов, собранных из социальной сети Вконтакте, и написанных людьми, которые в своем профиле указывали возраст и/или образование.

Считается, что чем больше обучающий корпус, тем лучше работает алгоритм. Так что не запрещается добавлять к обучающему корпусу свои примеры. Об этом должно быть указано в описании решения.

Тренировочный корпус

Тренировочный корпус доступен для скачивания в формате json_lines (кнопка "Datasets" в правом верхнем углу интерфейса).

Тестирование

Загрузка решения

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

- классификатор в файле solution.py. В файле должен содержаться класс Solution. В классе должны присутствовать методы
    - train(self, training_corpus). На вход метод train получает тренировочный корпус: json.loads каждой строчки из обучающего файла. Метод train ничего не возвращает. **Внимание:** метод train будет вызываться отдельно, так что не стоит вызывать его в конструкторе класса. Также для ускорения проверки рекомендуется сохранять натренированные модели с помощью библиотеки Pickle, и присылать их вместе с решением, если позволяет размер. При наличии сохраненных моделей функция train должна автоматически загружать их.
    - get_age (self, texts), который получает на вход список сообщений от одного автора и возвращает интервал, в котором находится его возраст. Допустимые значения: "<=17","18-24", "25-34", "35-44", ">=45"
    - get_education (self, texts), который на вход получает список сообщений от одного автора и возвращает уровень его образования: "high" - высшее, "middle" - среднее, "lower" - ниже среднего.
- (Пустой) файл __init__.py в корне архива. (Требования к пакетам Python).
- Описание применяемых алгоритмов в файле description.txt. Пожалуйста, напишите подробное описание, какие методы и признаки использовались. Это описание будет выложено вместе с решением после завершения курса.
- Все используемые внешние библиотеки, кроме библиотек описанных выше.

Ограничения

1. каждую неделю можно послать только 10 версий программы (внимание! Итоговое тестирование будет проводится на последнем загруженном решении)
2. размер архива не может превышать 15Мб
3. Время тестирования одного решения не может превышать 20 минут

4. На машине с тестами доступно 16Gb RAM

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. Рекомендуется использовать метод StratifiedKFold().

Оценка качества

Предлагаемые задачи будут оцениваться независимо. Однако это не означает, что вы не можете использовать результаты одного классификатора для улучшения качества другого.

Для оценки качества будет использоваться F_1-мера, которая равна среднему гармоническому точности и полноты.

$$F_1 = \frac{2PR}{P+R};\ P = \frac{|correct\ answers|}{|total\ answers|};\ R = \frac{|correct\ answers|}{|expected\ answers|};$$

Baseline

Для обеих задач использован SVM с линейным ядром, и юниграмами по словам, взвешенными с помощью метрики tf-idf, в качестве признаков.

Подсчет очков

В конце каждой недели (неделя кончается в 23:59:59 каждого понедельника) вы сможете посмотреть, насколько хороший метод вы сделали по сравнению с другими предложенными решениями. Эти результаты нужны только для понимания текущей ситуации.

В течение семестра будет два дедлайна, когда текущие результаты преобразуются в очки, которые повлияют на итоговую оценку за курс.

Расписание дедлайнов:

1. 20 ноября (учитываются все решения, присланные до 23:59:59, 20 ноября)
2. 18 декабря

При наступлении дедлайнов, так же как и в конце обычной недели производится обучение и тестирование всех присланных решений. Далее производится ранжирование результатов соответствующей мере, и начисляются очки: за 1 место – 10 очков, 2-9 и т.д. Все программы выше лучшего baseline получают минимум по 2 очка, выше худшего - минимум по одному очку (с учетом ранжирования). Если ни один baseline не преодолен - 0 очков. После этого результаты становятся доступны всем на главной странице.

Первое задание можно сдавать до первого дедлайна без штрафа. При сдаче после первого делайна количество полученных очков уменьшается в два раза с округлением в большую сторону. За задание выставляется максимальный из полученных баллов.

Второе задание можно сдавать до второго дедлайна без штрафов. В зачет пойдут оценки, полученные при подсчете второго дедлайна.

Выставление оценок

После 19 декабря будут выставляться итоговые оценки.

- Для получения отметки "**Отлично**" (9 баллов для ВШЭ) - необходимо набрать минимум 2 балла за каждое задание и не менее 5 баллов в сумме (решения лучше baseline и хотя бы раз (вовремя) попали в top-8).
- "**Хорошо**" (7 баллов ВШЭ) ставится за 3-4 балла, минимум 1 балл за задание (надо вовремя побить baseline).
- Для получения отметки "**Удовлетворительно**" (5 баллов для ВШЭ) необходимо набрать минимум по 1 баллу за задание (побить baseline 1 для обоих заданий).
- Оценка "**Неудовлетворительно**" ставится, если хотя бы одно задание не сдано.

Для студентов ВМК МГУ

Полученная оценка - это оценка за практикум.

Внимание! Оценку "неудовлетворительно" изменить можно на комиссии!

Экзамен

Для студентов ВМК МГУ

Экзамен будет проходить сразу после завершения практического задания. Оценка за практикум не влияет на оценку за экзамен.

Для студентов ФКН ВШЭ

Итоговая оценка за курс (по 10-бальной шкале) средним арифметическим (с округлением в большую сторону) за практическую часть и за экзамен.

Оценка "неудовлетворительно" за любую часть является блокирующей, то есть итоговая оценка тоже будет "неудовлетворительно".

Дополнительные вопросы

- Все технические вопросы относительно проверки заданий просьба присылать на laguta@ispras.ru либо спрашивать в разделе сайта, посвященном практикуму.
- Все остальные вопросы задавайте на сайте http://tpc.at.ispras.ru, либо пишите на turdakov@ispras.ru
- Для установки внешних модулей (NLTK, scikit-learn) рекомендуется использовать pip install: https://pip.pypa.io/en/latest/installing/

Вспомогательная литература

- Steven Bird, Ewan Klein, and Edward Loper. Natural Language Processing with Python(Книга про обработку текста с помощью библиотеки NLTK для языка Python. Доступна на сайте NLTK)
- Daniel Jurafsky, James H. Martin. Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition (Одна из лучших книг про обработку текстов)
- Christopher D. Manning, Hinrich Schütze. Foundations of Statistical Natural Language Processing (Книга содержит хорошие примеры применения машинного обучения для обработки текстов)

- Тоби Сегаран, “Программируем коллективный разум” (Книга про прикладное применение некоторых технологий искусственного интеллекта, включая машинное обучение, в Web 2.0 с огромным количеством примеров на Python).
- Турдаков Д. Ю. Методы и программные средства разрешения лексической многозначности терминов на основе сетей документов. Диссертация. http://www.ispras.ru/publications/2010/methods_and_software_for_word_sense_disambiguation_based_on_documents_networks/
- https://scholar.google.ru/scholar?hl=ru&q=wikification - статьи по теме