Практическое задание для студентов 4 курса кафедры СП. Осень 2016

Постановка задачи

Целью работы является создание системы, позволяющей извлекать упоминания объектов из текстов на русском языке. Задачу извлечения объектов можно разделить на две части: обнаружение терминов (англ. term detection) и определение значений терминов (англ. disambiguation). Извлечение терминов заключается в выделении в тексте подстрок, являющихся упоминаниями конкретных объектов (люди, предметы, города, и т.п.). Под определением значений терминов подразумевается выбор конкретного значения каждого термина (назначение каждому термину идентификатора объекта из некоторой базы знаний).

Предлагается разработать две системы:

1. Систему, которая на вход получает текст на русском языке, а на выходе отдает координаты терминов в этом тексте.
   Например,
   «Спортивный арбитраж сократил срок дисквалификации Марии Шараповой» → (0, 19), (50, 65)
2. Систему, которая на вход получает текст на русском языке и координаты терминов в этом тексте, а на выходе отдает значение для каждого термина (идентификатор статьи русской Википедии).
   Например, «Спортивный арбитраж сократил срок дисквалификации Марии Шараповой», (0, 19), (50, 65) → {(0, 19): https://ru.wikipedia.org/wiki/Спортивный_арбитражный_суд, (50, 65): https://ru.wikipedia.org/wiki/Шарапова,_Мария_Юрьевна}

Решение задачи

Практические аспекты

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

- NLTK - инструменты для обработки текстов
- scikit-learn - алгоритмы машинного обучения
- numpy - работа с многомерными массивами
- Keras + Theano - библиотека для работы с искусственными нейронными сетями
- ISPRAS API (https://api.ispras.ru) - программный интерфейс к системе Texterra. Можно использовать любые функции за исключением функций поиска терминов и их значений (KB-based NLP Services, Terms search and features), доступ к ним будет закрыт на проверяющей машине.

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. Считается, что чем больше обучающий корпус, тем лучше работает алгоритм. Однако создание большого обучающего корпуса - довольно трудоемкая задача, непосильная одному человеку. Поэтому предлагается создать его с помощью коллективной работы. Чтобы облегчить эту работу, был сделан сайт: http://objects.at.ispras.ru/.

Разметка обучающего корпуса

Инструкция по разметке находится в отдельном документе.

Тренировочный корпус

Тренировочный корпус будет доступен для скачивания в формате json. Для извлечения информации из этого файла рекомендуется использовать стандартную библиотеку Python с одноименным названием.

Для синхронизации обучения и тестирования в течение недели, корпус будет состоять из текстов, размеченных автором классификатора, плюс все тексты, размеченные в течение предшествующей недели.

Тестирование

Загрузка решения

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

- 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 список размеченных сообщений. 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 должа автоматически загружать их.
    - get_terms (self, text), который получает на вход неразмеченное сообщение и возвращает список аннотаций для терминов, найденных в тексте.
    - get_meanings(self, text, terms), который на вход получает текст и список терминов (аннотаций) и возвращает хэш-таблицу с отображением из аннотации термина в значение термина. (см. пример в начале файла).
- (Пустой) файл __init__.py в корне архива. (Требования к пакетам Python).
- Описание применяемых алгоритмов в файле description.txt
- 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

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.

Задача 1

Для оценки качества обнаружения терминов будет использоваться F_1-мера по точному совпадению границ терминов, которая равна среднему гармоническому точности и полноты.

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

Задача 2

В рамках задачи мы выделили два типа значений: точные и неточные. Приоритетно выделение именно точных значений, поэтому ответ должен считаться правильным, если найдены все точные значения (которых может быть несколько). При этом нахождение неточного значения также должно поощряться. Ситуация, когда найдены некоторые точные значения (не все) и некоторые неточные значения должна быть оценена выше, чем нахождение тех же точных значений без неточных. Таким образом, предлагаю оценивать правильность каждого ответа по следующей формуле:

- T – термин
- $Exact(T)$– множество точных значений термина;
- $Inexact(T)$– множество неточных значений термина;
- $Actual(T)$– множество значений термина, выявленное тестируемой системой;

$$W(T) = \begin{cases} \frac{|Exact(T) \cap Actual(T)|}{|Exact(T) \cup Actual(T)|} + \alpha(T)\frac{|Inexact(T) \cap Actual(T)|}{\left|Inexact(T) \cup \left(Actual(T) \backslash Exact(T)\right)\right|}, & Inexact(T) \neq \emptyset \\ \frac{|Exact(T) \cap Actual(T)|}{|Exact(T) \cup Actual(T)|}, & Inexact(T) = \emptyset \end{cases}$$

$$\alpha(T) = \frac{1}{|Exact(T)| + 2};$$

$W(T)$ достигает единицы в случае, если обнаружены все точные значения (при этом неточные значения не обнаружены), в остальных случаях $W(T) < 1$.

Общее качество считается, как средняя правильность определения терминов в тексте:

$$Q = \frac{1}{n}\sum_{i=1}^{n} W(T_i)$$

Baseline

Задача 1

Baseline 1. SVM с n-граммами в качестве признаков (BIO разметка[1][2])

Baseline 2. Метод выделения терминов системы Texterra.

Первый классификатор будет тренироваться на том же корпусе, что и присланные алгоритмы.

Задача 2

Baseline 1. Алгоритм, выбирающий наиболее частое значение, натренированный на Википедии.

Baseline 2. Метод определения значений из Texterra.

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. 22 ноября (учитываются все решения, присланные до 23:59:59, 21 ноября)
2. 20 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 20 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).

[1] https://lingpipe-blog.com/2009/10/14/coding-chunkers-as-taggers-io-bio-bmewo-and-bmewo/

[2] https://pythonprogramming.net/using-bio-tags-create-named-entity-lists/

- "**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.
- Вопросы по разметке данных пишите на vmayorov@ispras.ru
- 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