Университет ИТМО, факультет программной инженерии и компьютерной техники

Двухнедельная отчётная работа по «Информатике»: аннотация к статье

Дата лекции: 15.10.20 Дата сдачи: 29.10.20

Выполнил(а) Романов А.М , № группы *P3110* , оценка

Фамилия И.О. студента не заполнять

|  |  |  |
| --- | --- | --- |
| 1. Частотный анализ русского текста и облако слов на Python | | |
| **ФИО автора статьи (или e-mail)**  *metaformus* | **Дата публикации**  "1" Сентября 2020 г. | **Размер статьи**  **1116** |
| **Прямая полная ссылка на источник и сокращённая ссылка**  *https://habr.com/ru/post/517410/*  http://bit.do/fKDLw | | |
| **Теги, ключевые слова или словосочетания**  Python, Машинное обучение | | |
| **Перечень фактов, упомянутых в статье**   1. Частотный анализ является одним из наиболее простых методов обработки текста. В качестве результата он выдаёт список наиболее часто встречающихся слов 2. Для такого метода обработки текста удобнее всего использовать Python-библиотеку NLTK 3. Есть достаточно большое количество для анализа текста на английском языке, однако случай с русским языком имеет некоторые особенности 4. Анализ текста делится на несколько этапов: Загрузка и сбор данных, очистка и предварительная обработка текста, удаление стоп-слов, перевод слов в основную форму, подсчёт частоты встречаемости слов в тексте 5. Для начала надо сделать провести стандартную операцию open c правами на чтение и перевести содержимое файла в строку   f = open('pushkin-metel.txt', "r", encoding="utf-8")  text = f.read()   1. Далее нужно обработать текст, убрав лишние   пробельные символы, запятые и цифры,  используя регулярные выражения и стандартный набор символов пунктуации модуля String   1. Задание строки фильтруемых символов будет   spec\_chars = string.punctuation + '\n\xa0«»\t—…'  выглядеть так:   1. Далее исключаем символы из заданной ранее   text = "".join([ch for ch in text if ch not in spec\_chars])  строки   1. Следующий этап – токенизация текста, стандартная функция   from nltk import word\_tokenize text\_tokens = word\_tokenize(text)  библиотеки NLTK. По своей сути является разбиением текста на  символы, слова и предложения   1. Далее идёт подсчёт статистики встречаемости слов в тексте. Для этого используется класс FreqDist библиотеки NLTK, который даёт на выводе словарь, элементы которого включают слово и его частоту Словарь:   FreqDist({'и': 146, 'в': 101, 'не': 69, 'что': 54, 'с': 44, 'он': 42, 'она': 39, 'ее': 39, 'на': 31, 'было': 27, ...})  from nltk.probability import FreqDist fdist = FreqDist(text) | | |
| **Позитивные следствия и/или достоинства описанной в статье технологии (минимум три пункта)**   * Возможность анализа текстов неограниченного объёма * Возможность гибко настраивать фильтрацию текста с помощью регулярных выражений * Универсальный формат вывода в виде словаря позволяет комфортно работать с данными дальше | | |
| **Негативные следствия и/или недостатки описанной в статье технологии (минимум три пункта)**   * Минус описанной в статье технологии в том, я уже битый час сижу и ищу в ней минусы, изучая остальные методы частотного анализа текстов * Как негативное следствие могу обозначить то, что я опять буду спать 4 часа, ведь доклад по истории никуда не денется. А минусы, тем не менее, искать надо * Как итог, главный недостаток описанной в статье технологии в том, что в ней нет недостатков * Я выяснил, что язык R “из коробки” предоставляет больше возможностей для аналогичной задачи... | | |