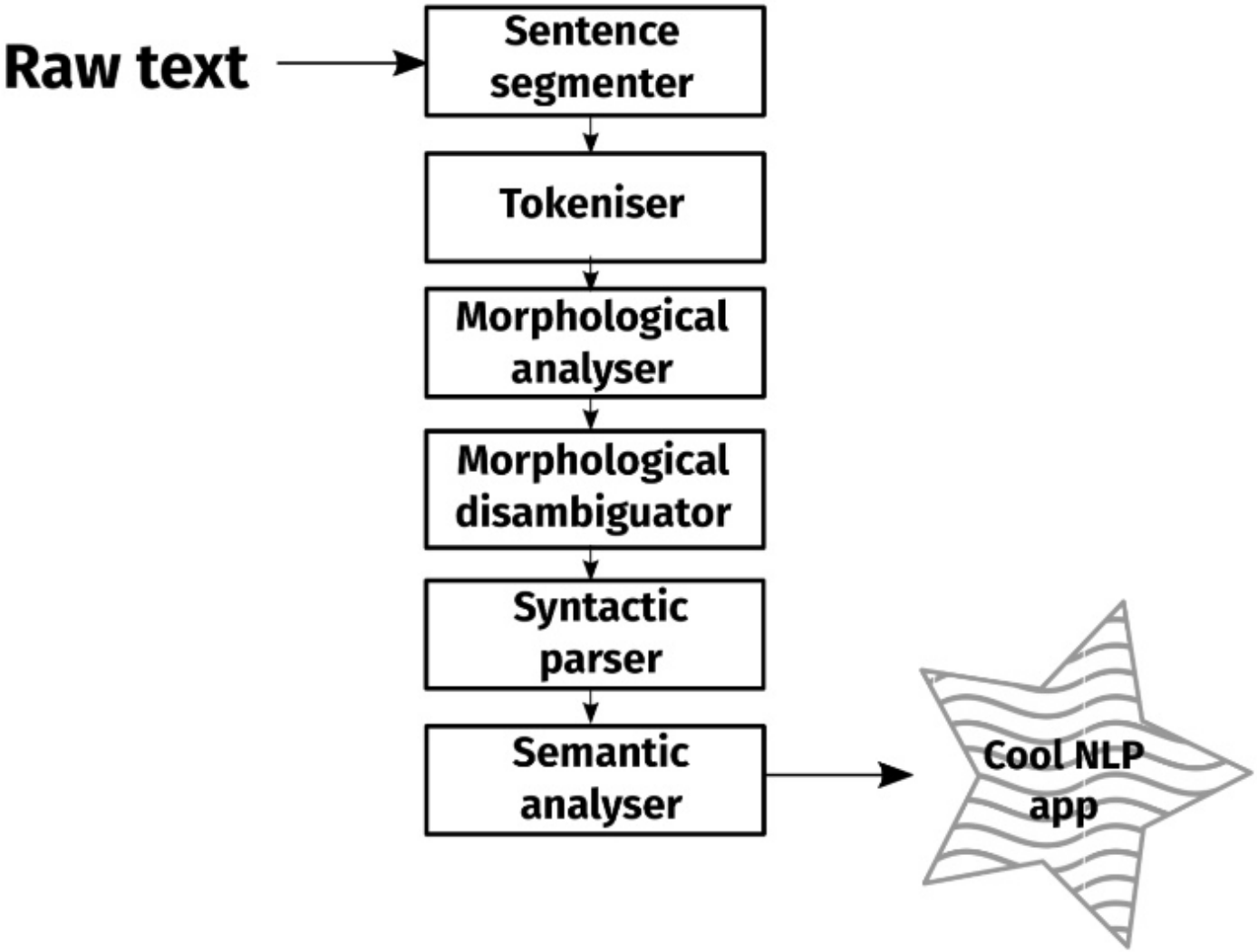
Сегментация, токенизация и регулярные выражения

Маша Шеянова, masha.shejanova@gmail.com

October 29, 2018

НИУ ВШЭ

ЛИНГВИСТИЧЕСКИЙ ПАЙПЛАЙН



СЕГМЕНТАЦИЯ

СЕГМЕНТАЦИЯ ТЕКСТА

(токенизация предложений, определение границ предложений)
У нас есть ”сырой” (не обработанный) текст.
Наша цель: **получить список предложений**.
Необходима для некторых следующих этапов пайплайна (синтаксический, семантический анализ).
Примеры приложения:

- машинный перевод
- создание параллельных корпусов (тексты на двух языках, выровненные по проедложениям)

Сегментация текста. Как выделить предложения?

Наивный способ: поделить весь текст по знакам препинания, стоящим в конце предложения (.!?)
Главная проблема: точка как конец предложения vs. в середине предложения.
Например:

- сокращения: г. Москва
- внутри дат: 23.05.2018
- внутри пунктов: согласно п. 2.1.13
- супер проблема: сокращение в конце предложения

подходы

Правила:

- заменяем всё, что не конец предложения, но имеет точку, на что-то другое
- используем много регулярных выражений
- пример правилового сегментатора: https://github.com/diasks2/pragmatic_segmenter

Машинное обучение: для каждой точке в предложении сопоставляем метку EOS (end of sentence) / не EOS.
Пример — `sent_tokenize` в `nltk`.

Обучение без учителя

Обучение без учителя (unsupervised machine learning) — направление машинного обучения, в котором мы пытаемся обойтись без расмеченного людьми набора данных.
Как сделать unsupervised ML для сегментации?

- берём огромный корпус текстов
- выучиваем аббревиатуры: считаем, какие слова встречаются перед точкой сильно чаще, чем обычно (коллокации с точками)

ЧТО ПОЧИТАТЬ

Два примера из журнала Computational Linguistics:

- Mikheev, A. (1994) “Periods, Capitalized Words, etc.” Computational Linguistics 16(4) — работает с определением границ предложения, словами с заглавной буквы и аббревиатурами
- Kiss, T. and Strunk, J. (2006) “Unsupervised Multilingual Sentence Boundary Detection”. Computational Linguistics 32(4) — обучение без учителя для обнарудения аббревиатур

ТОКЕНИЗАЦИЯ

ТОКЕНИЗАЦИЯ

У нас есть ”сырой” (не обработанный) текст.
Наша цель: **получить список слов (токенов)**.

Токен — в первую очередь слово, но к ним также относятся знаки препинания, даты и прочие сегменты предложения.

Необходима для практически всех[1] лингвистических задач и всех следующих шагов пайплайна.

[1] кроме тех, где задача решается на уровне символов

Токенизация. Как выделить токены?

Наивный способ: кусок строки от пробела до пробела.
Дает нормальное качество, но обычно нужно лучше. **Чем хуже токенизация, тем хуже работает вся система в целом.**
Проблемы:

- знаки препинания – удалить, оставить?
- сокращения и другие апострофы (don’t, we’re, Smith’s)
- дефисы (Санкт-Петербург vs мальчик-программист)
- составные предлоги (в течение, не работает)
- и многие, многие другие детали

ТОКЕНИЗАЦИЯ. ПРАВИЛА

Очень простой подход: используем набор эвристик и регулярные выражения

- окружаем always-separating punctuation (;!?) пробелами
- специально обрабатываем числа
- специально обрабатываем аббревиатуры
- специально обрабатываем слова типа что-то
- ... и разделяем по пробелам

Но что делать с языками без пробелов?
MaxMatch: находим самую длинную последовательность! (Для этого нужен словарь).

Токенизация. Статистическая

Классификация:

- разделяем текст на символы
- учим классификатор предсказывать, следует ли за символом граница

Графовый подход!

NLTK

ЧТО ТАКОЕ NLTK

nltk (natural language toolkit) — питоновская библиотека, в которой есть много инструментов для естественного языка
документация: http://www.nltk.org/

nltk умеет:

- сегментировать (sent_tokenize)
- токенизировать (word_tokenize)
- убирать стоп-слова
- и многое-многое другое

РЕГУЛЯРНЫЕ ВЫРАЖЕНИЯ

Что такое регулярные выражения?

регулярные выражения (regular expressions, RegExp, регулярки) — удобный инструмент, с помощью которого можно задавать, вылавливать из строк и заменять определённые паттерны

Что можно сделать регулярками:

- найти в тексте все эмейлы, даты, слова в кавычках
- найти все формы одного слова
- убрать из текста все переводы строк
- и многое другое

РЕГУЛЯРНЫЕ ВЫРАЖЕНИЯ

РЕГУЛЯРНЫЕ ВЫРАЖЕНИЯ В PYTHON

```
> import re
> my_string = '123 lskd 145ddd kzjsn 67996'
> re.findall('[0-9]+', my_string)
['123', '145', '67996']

> re.sub('[0-9]+', 'aaa', my_string)
'aaa lskd aaaddd kzjsn aaa'
```

РЕСУРСЫ

Шпаргалка: http://www.cbs.dtu.dk/courses/27610/regular-expressions-cheat-sheet-v2.pdf
Интерфейс, в котором можно проверять регулярки:
https://regex101.com/
Туториал:
https://developers.google.com/edu/python/regular-expressions