Извлечение информации из неструктурированных источников с использованием имеющихся табличных данных

Антон Бабичев

Научный руководитель Недумов Ярослав Ростиславович

Факультет ВМК МГУ им. М.В. Ломоносова
Кафедра Системного Программирования

20 мая 2014 г.

Неформальная постановка задачи

Дан неструктурированный текстовый пост — описание товара в интернет-магазине.

Например, “HP ENVY TouchSmart 15.6 inches Touch-Screen Laptop Intel Core i5 8GB Memory 750GB Hard Drive Natural Silver”

Необходимо исследовать его содержимое, руководствуясь структурированной базой знаний о товарах того же типа.

Применение

- Построение базы знаний по содержанию веб-страниц
- Связывание данных на разных страницах, посвященных одному и тому же объекту
- Извлечение дополнительных сведений об объекте(цена товара, дата публикации и т.д.)

Формальная постановка задачи

Входные данные: текстовый пост как набор токенов $\{t_1, t_2, \ldots t_n\}$
База знаний: множество наборов пар $\{attr : val\}$
Один набор отражает информацию об одной единице товара
Выходные данные: набор меток-атрибутов для токенов поста либо метка “пустышка”
$\{tok : attr|junk\}$

Задачи

- Провести обзор существующих методов извлечения структурированных данных
- Выбрать наиболее перспективный метод и реализовать его
- Протестировать метод на домене, отличном от авторского

Методы решения

- Методы на основе NLP
  Предполагается, что текст написан на естественном языке
- Методы на основе Information Extraction
  Предполагается наличие какой-либо структуры текста
- Метод на основе множества элементарных исходов
  Разработан специально для данной задачи

Michelson, Knoblock, Creating Relational Data from Unstructured and Ungrammatical Data Sources, 2008

Использованные данные

- Домен: описания ноутбуков на английском языке
- База знаний: 8055 записей с *www.amazon.com*
  Запись — описание и набор характеристик
- Данные для тестирования: 104 размеченных вручную записей с *www.bestbuy.com*
  Запись — описание с набором меток для токенов

Описание метода

1. Предобработка
2. Связывание
3. Разметка

Описание метода — предобработка

Приблизительное сравнение поста с элементами базы знаний $\longrightarrow$ подмножество базы знаний
Записи базы знаний кластеризуются по группам атрибутов.
Для описания одного кластера используется правило в виде ДНФ:

$$(\{attr_1 = val_1\} \wedge \{attr_2 = val_2\}) \vee \{attr_3 = val_3\} \vee \ldots$$

Для оптимального разбиения используется алгоритм BlockSchemeLearner.

Описание метода — Связывание

Подмножество базы знанй $\longrightarrow$ единственный элемент базы знанй — схема поста

$\{post, candidate\} \longrightarrow$ дескриптор

Дескриптор $\longrightarrow$ SVM классификатор

Описание метода — Разметка

Токены поста → массив меток атрибутов и “пустышки”
{токен, схема поста} → дескриптор → Multi-Class SVM классификатор
Multi-Class SVM классификатор обучен ставить токену в соответствие метку атрибута
После проводится процедура чистки размеченных токенов, где отсеиваются “пустышки”

Реализация

В качестве языка реализациии был выбран язык программирования *Java*
Для классификации используется библиотека *libsvm*

Разработаны и реализованы 2 основных класса:

- BloskSchemeLearner выполняет предобработку базы знаний
- PostExplorer проводит связывание и разметку поста

Загрузка данных осуществлена с помощью краулера, написанного на Python с использованием библиотеки scrapy

Результаты

- Был реализован и протестирован метод, описанный в статье
- Тестирование проводилось для шести атрибутов ноутбуков (производитель, диагональ, цвет...)
- В среднем
  $F_1 = 0.67, precision = 0.58, recall = 0.82$
- Лучший результат (диагональ):
  $F_1 = 0.94, precision = 0.92, recall = 0.96$
- Худший результат (линейка):
  $F_1 = 0.38, precision = 0.26, recall = 0.66$
- Основные ошибки связаны со спецификой выборки и процедурой чистки

Дальнейшие планы

- Адаптация метода — изменение дескриптора, классификаторов, поддержка русского языка и т.д.
- Улучшение процедуры чистки размеченных токенов
- Тестирование на большей базе

Спасибо за внимание!