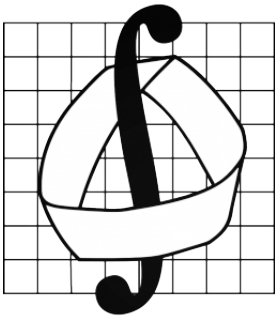
МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
имени М. В. ЛОМОНОСОВА
Механико-математический факультет, 5 курс
Кафедра Теоретической Информатики



Хромов Михаил
КУРСОВАЯ РАБОТА

Семантический анализ русскоязычных текстов

Semantic analysis of russian texts

Научный руководитель – Шокуров А. В.

Москва, 2018

1 Введение

Сейчас в мире активно развивается тема анализа текстов на различных языках для выявления фактов, классификаций, смысла. К примеру, создается множество автоматизированных систем, способных распознавать речь, вести диалог и выполнять команды. Кроме того, в различных секторах рынка необходимы боты, которые бы переписывались с клиентами и принимали заказы, обрабатывали жалобы, принимали предложения.

На данный момент уже есть немало разработок в этой области. К примеру, есть проект Stanford Parser. Но он не работает с русским языком. И некоторые этапы обработки, которые они делают не подходят для нашего языка. Также компания Яндекс занимается этой темой. Но готовых открытых инструментов для анализа текстов на русском языке всё-равно пока нет.

2 Подзадачи и методы их решения

Процесс анализа текста разбивается на множество непростых этапов обработки:

- Разделение на предложения
- Токенизация предложения
- Морфологический анализ отдельного слова
- Морфологический анализ использующий контекст
- Построение дерева зависимостей

2.1 Токенизация текста

Тут на первый взгляд всё просто. Делим текст основываясь на знаках конца предложения (". "? "!"и т.д.). Но сложности возникают, когда встречаются слова: "S.T.A.L.K.E.R."или "ул. Лебедева". Так что надо основываться не только на знаках препинания, но и на том, как "обычно"заканчиваются преложения, и как начинаюся новые. В библиотеке Natural Language Toolkit есть метод под названием sent tokenize, который обучен на многих европейских языках делить текст на предложения. Им мы и будем пользоваться.

Далее надо разделить полученные предложения на токены - слова и знаки препинания. Та же библиотека умеет это делать.

2.2 Морфологический анализ

Для морфологического анализа будем использовать библиотеку pymorphy2. Она обучена определять лексемы слов. Но так как она не использует контекст, то определяется это неоднозначно. Поэтому дальше стоит задача определить из списка возможных лексем нужную. Для предлогов и существительных действуем алгоритмически и находим нужные падежи. Остальные пока оставляем.

2.3 Выявление зависимостей

Для данной задачи уже не найдено ни одного инструмента/библиотеки, которые могли бы нам помочь. Алгоритмы, которые решают эту задачу для английского языка основываются на типичности структур предложений этого языка. Так что действуем самостоятельно. Сначала пробуем определить основу предложений (подлежащие - сказуемое), где подлежащее - существительное это местоимение, а сказуемое это один или несколько глаголов. Подаем на вход классификатору множество пар, некоторые из них подлежащее и сказуемое, некоторые нет и обучаем. Далее делаем аналогично для определения зависимостей слов.

2.4 Метод и входные данные

Для обучения будем использовать метод градиентного бустинга, который будет минимизировать логарифмическую функцию потерь. Сначала мы создаем из слова вектор, каждая компонента которого число, обозначающая номер лексемы. К примеру, если это существительное, то первая компонента вектора будет 1. Список лексем: часть речи, число, род, падеж, склонение, сов./несов. вид, одушевленность, залог, время, переходность, наклонение, включенность говорящего в действие. Для некоторых слов часть лексем может быть не определена (не существует), тогда ставим 0 в соответствующую компоненту. Всего 12 лексем. На вход классификатору подаём вектор размерности 24 (2 слова по 12). На выходе 0 (нет зависимости) или 1 (есть зависимость).

3 Данные для обучения

Отметим, что при поиске готовых баз размеченных зависимостей ничего не было найдено, так что появилась существенная сложность в сборе достаточного количества данных для обучения. Были взяты несколько текстов различной литературы и вручную размечены зависимости (около 2000) и основы простых предложений (около 200).

Для сбора данных было необходимо каким-то образом сгенерировать много словосочетаний из двух слов, так чтобы около половины были с зависимостью, около половины без зависимости.

Как выяснилось эксперементально, если взять достаточно много текста (тестировалось на литературном), поделить его на предложения, а затем выписать в таблицу пары слов идущих друг за другом, то около 44% пар слов зависимы. Таким образом, были взяты несколько томов различной литературы, по несколько десяткой предложений, разбиты как описано выше и напротив каждого словосочетания было отмечено, есть ли у слов зависимость или нет.

4 Результаты

Точность определения зависимостей между двумя словами составила 0.844 Для этого было размечено еще около 200 зависимостей. Определить точность поиска

основы предложения не получилось, так как была как маленькая обучающая выборка, так и тестовая, из-за медленной разметки основ. Но были протестированы некоторые виды предложений:

Простой вариант: Алиса пошла в кино
Результат: алиса пошла

Вариант посложнее: Старый Боб пошел и лег спать
Результат: боб пошёл и лёг

Вход: Боб увидел Алису в саду залезающую на дерево и побежал к ней
Выход: боб увидел и побежал

5 Заключение

После пройденных этапов осталось еще много работы до построения семантического дерева зависимостей. В планах собрать больше данных, применить дополнительно некоторые алгоритмизированные функции, которые помогли бы лучше определять и лексемы и зависимости между словами. Также построить нейросеть, которая будет учитывать контекст, а не брать на вход только по два слова.

Список литературы

[1] Over 80 practical recipes on natural language processing techniques using Python's NLTK 3.0 Jacob Perkins

[2] https://habr.com/post/264339/

[3] http://www.nltk.org

[4] http://nlp.stanford.edu:8080/parser/index.jsp

Содержание