Современные подходы к разработке интеллектуальных систем. Классические методы извлечения знаний из текста. Нейронные сети для анализа текста.

Лекция 3

Киселёв Глеб Андреевич
к.т.н., старший преподаватель
ФФМиЕН РУДН
тел.:+79067993329
email: kiselev@isa.ru

Создание интеллектуальных систем

Интеллектуальные системы 3 типов:

- Общего назначения – базовые механизмы накопления и хранения знаний и вывода по ним;
- Проблемно(предметно) ориентированные – БЗ + иерархия объектов в предметной области;
- Динамические – Все выше перечисленное + изменение во времени.

Примеры систем:

- EMYCIN – первая медицинская ЭС;
- TEIRESIAS – интеллектуальный редактор пополнения БЗ;
- DI*GEN – механизм создания ЭС на основе фреймов;
- CAKE (Computer Aided Knowledge Engineering) – механизм создания БЗ на основе 3 уровней концептов;
- РЕПРОКОД – Механизм построения ЭС на основе семиотический сетей;
- STEPCLASS – механизм структуризации знаний и создания правил вывода по ним;
- SIMER+MIR – система формирования БЗ (получение знаний из текста, логических рассуждений, обучения на примерах);
- G2 – современное средство построения динамических систем. Определяет порядок обработки задач, взаимодействует с источниками данных, запускает процессы, осуществляет коммуникацию с другими процессами в сети. Имеет планировщик задач.

Технологии обработки текста

Язык – множество цепочек символов из некоторого алфавита, удовлетворяющим правилам лексики, грамматики и фонетики.

Какие бывают языки? – естественные и формальные. Естественные: русский, английский, испанский… Формальные: Python, Java, язык формул…

Алфавит – множество символов, которые можно использовать для построения текста.

Как научить компьютер распознавать или генерировать текст? Как передать смысл?

3 группы задач обработки естественного языка – лингвистический анализ, извлечение признаков из текстов, прикладные задачи пользователей.

Лингвистический анализ – набор методов для разбора структуры текста.

Извлечение признаков – требуются для предобработки текста для машинного обучения.

Прикладные задачи – то же самое, что и предыдущие 2 группы + персонализация для задач пользователя.

Лингвистический анализ состоит из основных элементов: графематический анализ (разбиение текста на токены), морфологический и семантический анализ предложений, анализ связи предложений и дискурса.

Для графематического анализа часто используют регулярные выражения (модуль re в Python). Для сокращения неоднозначности разделения предложения используются вероятностные модели.

В морфологическом анализе происходит определение частей речи, падежей, числа, начальной формы для каждого из токенов. (без разрешения омонимии). Используются словари, в которых описана вся информация для наиболее частотных слов. Пример: Гладь озера была неповторимой. Не знаешь, что делать с кошкой – гладь. Морфологический анализатор предложит 2 словоформы от слова «Гладь» - описание глагола и существительного. Для разрешения омонимии применяют как методы основанные на правилах, так и вероятностные модели и рекуррентные нейросети.

Модуль Re

Основной синтаксис

.	Один символ кроме новой строки
\.	Просто точка `.`, обратный слеш `\` убирает магию всех специальных символов.
\d	Одна цифра
\D	Один символ кроме цифры
\w	Один буквенный символ, включая цифры
\W	Один символ кроме буквы и цифры
\s	Один пробельный (включая таб и перенос строки)
\S	Один не пробельный символ
\b	Границы слова
\n	Новая строка
\t	Табуляция

Модификаторы

$	Конец строки
^	Начало строки
ab|cd	Соответствует ab или de.
[ab-d]	Один символ: a, b, c, d
[^ab-d]	Любой символ, кроме: a, b, c, d
()	Извлечение элементов в скобках
(a(bc))	Извлечение элементов в скобках второго уровня

Повторы

[ab]{2}	2 непрерывных появления a или b
[ab]{2,5}	от 2 до 5 непрерывных появления a или b
[ab]{2,}	2 и больше непрерывных появления a или b
+	одно или больше
*	0 или больше
?	0 или 1

Модуль Re

Основные функции модуля **re**:

- `match` - ищет последовательность в начале строки
- `search` - ищет первое совпадение с шаблоном
- `findall` - ищет все совпадения с шаблоном. Возвращает результирующие строки в виде списка
- `finditer` - ищет все совпадения с шаблоном. Возвращает итератор
- `compile` - компилирует регулярное выражение. К этому объекту затем можно применять все перечисленные функции
- `fullmatch` - вся строка должна соответствовать описанному регулярному выражению

Кроме функций для поиска совпадений, в модуле есть такие функции:

- `re.sub` - для замены в строках
- `re.split` - для разделения строки на части

Более подробно можно прочитать тут:

1. https://habr.com/ru/post/349860/
2. https://pythonru.com/primery/primery-primeneniya-regulyarnyh-vyrazheniy-v-python
3. https://docs.python.org/3/library/re.html

Примеры:

```
In [1]: import re
        text = "мама мыла раму очень чисто в 2004 году, \
        но дальше мыл раму папа."
```

```
In [2]: re.split('\s+', text)
```

```
Out[2]: ['мама',
         'мыла',
         'раму',
         'очень',
         'чисто',
         'в',
         '2004',
         'году,',
         'но',
         'дальше',
         'мыл',
         'раму',
         'папа.']
```

```
In [3]: regex_num = re.compile('\d+')
        regex_num.findall(text)
```

```
Out[3]: ['2004']
```

```
In [4]: text = 'почта glebkiselev777@gmail.com, \
        ФИО: Киселёв Глеб Андреевич, тел:  +79999999999, \
        2 email: kiselev@isa.ru'
        re.findall('[a-zA-Z0-9-_.]+@[a-zA-Z0-9-_.]+', text)
```

```
Out[4]: ['glebkiselev777@gmail.com', 'kiselev@isa.ru']
```

Извлечение именованных сущностей

Выявление классов принадлежности сущностей. Механизмы – словари, системы правил, регулярные выражения, вероятностные модели (Conditional Random Fields, Hidden Markov Models), нейросети.

Синтаксический анализ

Результат – дерево зависимостей, в рамках которого для каждого токена существует ссылка на его предка. Механизмы – анализаторы на основе решающих правил и нейросетей.

Семантический анализ

Выделение главного слова (предиката) – глагол или отглагольное существительное. Далее выделяется набор семантических ролей, связанных с предикатом. Механизмы – анализаторы на основе правил или нейросетей. После этого выявляются неявные отношения между сущностями (принадлежность одной сущности другой, вхождение 1 сущности в другую, местонахождения и т.д.). Механизмы – специальные классификаторы естественных языков.

Извлечение анафорических связей

Анафора – местоимение, ссылающееся на ранее упоминаемый объект. Механизм – извлекаются все возможные пары местоимение-референт и применяется классификатор.

Код для ознакомления: https://github.com/IINemo/isanlp

Дискурсивный анализ

Синтаксический анализ между документами. Механизмы – анализаторы класса «сдвиг-свертка».

Генерация текста

Современные механизмы основаны на нейросетевом подходе. На выходе нейросети – последовательность лексем или текст. Смысл предложения задается вектором признаков (эмбеддингом).

Извлечение признаков

Методы основаны на машинном обучении. Для применения методов требуется представить данные в виде векторов и матриц (тензоров), либо графов.

Разреженное векторное представление

Двоичный вектор – элемент равен 1, если слово присутствует в тексте и 0, если нет.

Пример: «Нет, ребята, я не гордый! Не загадывая в даль, так скажу: зачем мне орден? Я согласен на медаль!»

Вектор:

Нет	ребята	обезьяна	…	танк	медаль
1	1	0		0	1

Плюсы: простота, размерность векторов большая.

Минусы: не учитывается вес слова, синонимы кодируются по-разному, чувствителен к шуму, слишком большая размерность вектора приводит к переобучению.

Измерение частотности слов

Нет	ребята	обезьяна	…	танк	медаль
0.001	1	0		0	0.1

Связанный механизм – формула Term Frequency – Inverse Document Frequency (TF-IDF)

Плюсы: простота, хорошо описывает тематику документов, большая размерность векторов, легко настроить модель знаний.

Минусы: чувствительность к опечаткам, остальные недостатки предыдущего подхода.

Модель описания документа

Каждый документ описывается длинным вектором, по документу составлен словарь. Вектор слов документа сильно разрежен – большинство значений нули.

Как считать вес слова?

1. Вес – количество употреблений слова в документе (самый простой способ. В длинных документах слова имеют очень большой вес, союзы предлоги, местоимения и тд – имеют гигансткий вес)
2. Вес – количество употреблений слова в документе, деленное на длину документа. По L2 норме: $nw_i = \frac{w_i}{\sqrt{\sum_j w_j^2}}$ плюсы – веса слов зависят от длины документа слабее, но все еще зависят.
3. Распределение Ципфа (https://ru.wikipedia.org/wiki/Закон_Ципфа): rank – порядковый номер слова после сортировки по убыванию частоты, s – коэффициент скорости убывания вероятности, N – количество слов, Z(s, N) - $\sum_{i=1}^{N} i^{-S}$ нормализационная константа.

$$f(rank; s, N) = \frac{1}{Z(s, N) rank^s}$$

(изображение взято с https://medium.com/@tigran.baseyan/закон-ципфа-какой-закон-какого-ципфа-975155b1cd03)

Суть распределения (закона) – распределение вероятностей описывающее взаимоотношение частоты события и количества событий с такой частотой. 2 параметра – s – скорость убывания и N – количество слов в нашем словаре.

Модель описания корпуса документов

Чем чаще слово встречается в документе, тем больше оно описывает тематику документа. Чем реже слово встречается в корпусе, тем более оно информативно. За это отвечают 2 величины: TF (term frequency) и IDF (inverse document frequency).

$TF(w,d) = \frac{WordCount(w,d)}{Length(d)}$, где $WordCount(w,d)$ - количество употреблений слова w в документе d, а $Length(d)$ - длина документа d в словах.

$IDF(w,c) = \frac{Size(c)}{DocCount(w,c)}$, где $DocCount(w,c)$ - количество документов в коллекции c, в которых встречается слово w, а $Size(c)$ – размер коллекции c в документах.

$TFIDF(w,d,c) = TF(w,d) \cdot IDF(w,c)$ - итоговый вес слова.

Алгоритм взвешивания признаков по TF-IDF:

1. Текст нормализуется (стемминг или лемматизация), выделяются базовые элементы (символы, токены,…);
2. Строиться частотный словарь DocCount(w,c) для всех w;
3. Словарь прореживается, удаляются слишком редкие и слишком частые слова;
4. Для каждого документа d строится вектор статистики встречаемости слов w (TFIDF). Вектора всех документов объединяются в матрицу;

Для общего ознакомления, еще 1 способ описать корпус документов:

- Pointwise mutual information pmi(A,B). Подробнее: https://medium.com/dataseries/understanding-pointwise-mutual-information-in-nlp-e4ef75ecb57a

Полезные ссылки:

http://nlpx.net/archives/57

https://stepik.org/course/54098/syllabus

N-граммы

Это объект, состоящий из идущих подряд нескольких символов или токенов.

Пример: «Нет, ребята, я не гордый! Не загадывая в даль, так скажу: зачем мне орден? Я согласен на медаль!»

- Символьные 3-граммы: Нет, реб, б_т, гор, рды, орд, с_г, даль,…
- Словарные 2-граммы: я гордый, загадывая даль, зачем орден, согласен на,…

Где применяются N-граммы: TF-IDF, дистрибутивной семантике (FastText – плотные векторные представления)

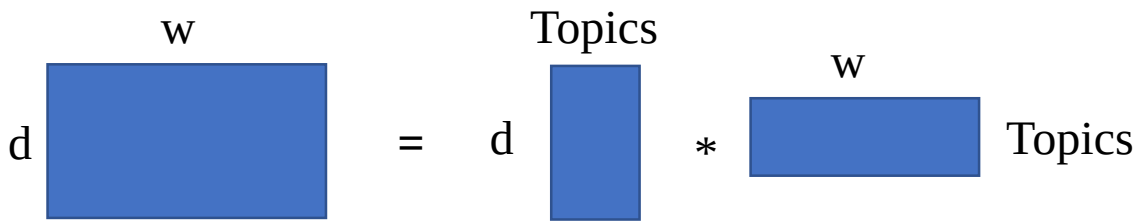
Основные бонусы: Простота, устойчивость к опечаткам, словоизменению, отсутствует надобность лемматизации и морфологического анализа, более специфичные по сравнению с отдельными словами;

Недостатки: высокая размерность и разреженность, близкие по смыслу слова кодируются по-разному;

Как облегчить задачу:

- Матричное разложение

и тематическое моделирование



- Предикативные дистрибутивно-семантические модели (Word2Vec) – предсказывают соседние слова в текстах;
- Предикативные модели текста (Bert, Gpt2/3/4) – предсказывают соседние слова в текстах;

Преимущества предикативных моделей: размеченная выборка не требуется, позволяет учитывать общий смысл фразы, можно сразу рассматривать несколько тематик текста;

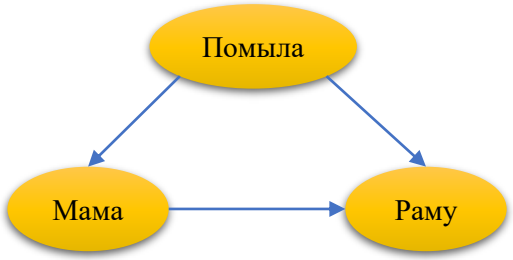
Недостатки:

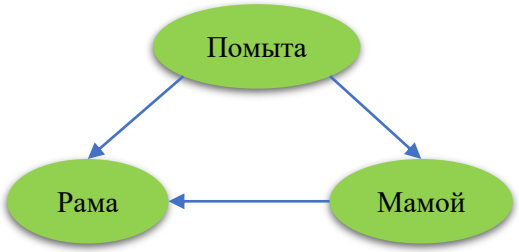
- Требовательность к вычислительной мощности;
- Требовательность к объему обучающих данных;

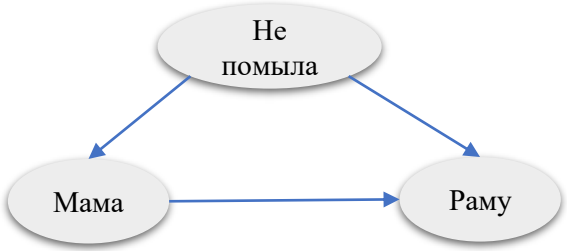
Ядерные методы

Текст – сложная, нелинейная структура, которую сложно закодировать 1 вектором.

Как же представить текст? Допустим, в виде графа:







- Оценить сходство двух векторов: $< a,b >= \frac{\sum_i^N a_i b_i}{\sqrt{\sum_i^N a_i^2}\sqrt{\sum_i^N b_i^2}}$ - скалярное произведение векторов (косинус угла между 2 векторами)
- Оценить сходство 2 графов можно с помощью ядра графа. Как сформировать ядро – несколько случайных обходов различной длины по графам, после которых происходит поиск пересечений проходов по разным графам. Чем больше пересекаются проходы, тем более похожи графы. Либо ядро учат с помощью нейросети.
- После обнаружения ядра можно воспользоваться методами опорных векторов или К-ближайших соседей.

Преимущества:

- Размеченная выборка не требуется;
- Для задач без больших размеченных корпусов;
- Позволяет учитывать сложные отношения слов.

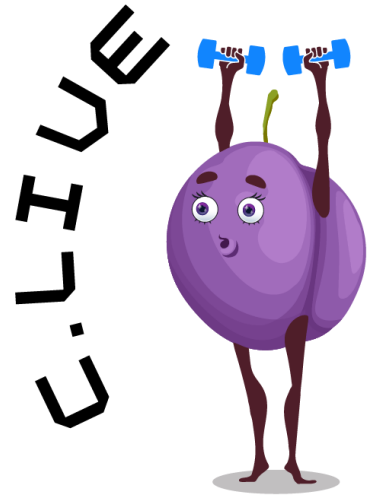
Недостатки:

- Медленные;
- Сильная зависимость от правильного выбора ядра.

ДЗ:

1. L1/L2 регуляризация: https://craftappmobile.com/l1-и-l2-регуляризация-для-логистической-р/
2. Более подробно про анализ текстов: https://stepik.org/course/54098/syllabus

Спасибо за внимание!



Руководитель проекта Когнитивный ассистент
старший преподаватель, к.т.н. Киселёв Г.А.
+79067993329
kiselev@isa.ru