МИНОБРНАУКИ РОССИИ

Федеральное государственное автономное образовательное учреждение высшего образования   
«Национальный исследовательский университет   
«Московский институт электронной техники»

Институт cистемной и программной инженерии и информационных технологий

(СПИНТех)

Постоев Игорь Евгеньевич

«Методы извлечения сущностей заданных классов из текста»

Москва 2020

Введение

В большинстве текстовых документов присутствуют группы слов, которые являются наиболее информативными и описывают объекты реального мира. Например, такими объектами могут быть люди, локации, временные промежутки, имеющие конкретные описания-значения. Данные объекты – сущности в тексте можно определить, и, затем, сформировать структурированный набор полезных данных текста. Такой результат можно использовать, например, для поиска того или иного текста на основе заданных требований к его содержанию, в том числе – для предоставления рекомендаций к прочтению, для категорирования текстов.

В направлении обработки естественного языка существуют разделы, в которых решается та или иная задача, будь то перевод текста, его генерирование, определение общего настроения текста. Также, существует раздел, в рамках которого из текста извлекаются значимые объекты-слова – сущности. Раздел называют «извлечение именованных сущностей» или NER(named entity recognition). Под извлечением понимается поиск в тексте групп слов, характеризующих объект требуемого класса.

Пусть дан текст: «Джо Байден — уже точно следующий президент США. А Виктор Прокофьев — один из самых опытных советских и российских переводчиков, который работал с сенатором Байденом во время его визита в СССР в 1988 году.»

Здесь можно выделить следующие слова-сущности и соответствующие им классы:

|  |  |
| --- | --- |
| Сущность | Класс |
| Джо Байден | Персона |
| Виктор Прокофьев | Персона |
| США | Локация |
| СССР | Локация |
| в 1988 году | Дата |

Методы NER

1. Правила

Человеком задаются правила-требования к словам, которые определяют слово к тому или иному классу. Такими правилами могут быть «начало с заглавной буквы и окончание на –ова || –ов || -ев для того чтобы отнести слово к классу «персона»».В совокупности с таким подходом, могут быть определены словари, в которых некоторые слова уже отнесены к классу: США – локация.

Правила могут быть представлены регулярными выражениями. Например, при наличии шаблона «^(19|20)\d\d**[**- /.**]**(0**[**1-9**]**|1**[**012**]**)**[**- /.**]**(0**[**1-9**]**|**[**12**][**0-9**]**|3**[**01**]**)» можно извлечь дату с 1900-01-01 до 2099-12-31.

Примером решения основанного на правилах является Pilot NER, достигшим точности 75% в задаче поиска имен поставщиков из отчетов денежных транзакций.

1. Статистические модели

Актуальность метода

Для человека ручное составление правил поиска слов класса может быть сложной задачей. Относительно несложно вручную создать парсер дат, ведь они имеют ограниченный набор форматов. А как, например, составить правило определения слов-локаций, имен? Дано предложение: «Куйбышев на этой неделе побил новый антирекорд по заболеваемости». В таком случае определить, чем является слово «Куйбышев» - городом или фамилией, можно только из контекста. Вариантов такого содержания может быть настолько много, что описать это правилами вручную является трудновыполнимой задачей. В таком случае можно прибегнуть к статистическим методам, с помощью которых можно определить вероятность отнесения слова к тому или иному классу. Примером такого подхода является использование математических моделей, действующих по принципу обучения с учителем.

Подготовка текста

Так как слова могут приводиться к разным формам путем добавления окончаний и аффиксов, что сильно увеличивает количество вариантов для одного и того же слова, проводят нормализацию текста. Стемминг – процесс поиска корня слова путем отбрасывания аффиксов/окончаний. В силу того, что в стеммнинге не учитывается контекст, нормальная форма слова может быть определена неправильно, например в слове «Caring» стемминг может оставить слово «Car», когда подразумевалось «Care». В процессе же лемматизации учитывается контекст на основе части речи и в предыдущем примере норма для слова была бы определена как «Сare», если бы слово являлось глаголом. Примерами программ, которые выполняют лемматизацию являются Wordnet Lemmatizer,Spacy Lemmatizer, TextBlob.

Определение части речи слова называют POS-тегированием(POS – Part of speech).Так как POS тегирование это, по сути, та же задача соотнесения слова классу (части речи), то в этом процесс могут использоваться те же методы что и в NER.

Представление слов в машинно-читаемом виде

После нормализации, словам можно назначить некоторое численное представление, например, позицию в созданном предварительно словаре. Обычно, для целей машинного обучения, слова представляют в виде векторов. Например, в унитарном коде представлен вектор, размер которого равен размеру словаря. Значения вектора нулевые, за исключения позиции слова в словаре, в которой значение = 1. Вместо единицы может быть частота употребления слова в данном и других текстах(TF-IDF).

Также, слова могут быть представлены в виде эмбеддингов. Метод основан на идее дистрибутивной семантики: допускается, что слова, которые встречаются в похожих контекстах, семантически близки, то есть имеют похожие значения. Тогда можно превращать слова в вектора произвольной длины (обычно 200-300 чисел), и чем чаще слова встречаются в похожем контексте — тем ближе эти вектора будут в пространстве. Для проведения данной операции, существует алгоритм word2vec и его модификации CBOW и skip-gram.

Частотный анализ текста

При данном подходе слову присваивается тот или иной класс на основе вероятности, что слово будет соответствовать ему. В тренировочном наборе данных определяется, с каким классом данное слово возникает чаще всего и на основе этого делается вывод о принадлежности слова классу.

Метод N-грамм

В данном случае вероятность отнесения слова классу определяется из вероятностей, рассчитанных для слов, следующих до или после текущего слова. Таким образом, определяется вероятность возникновения последовательности из классов слов.

Идея основана на теореме Байеса. Пусть C – множество классов для слов, W – множество слов тренировочного набора. Согласно теореме, вероятность возникновения последовательности классов определяется как:

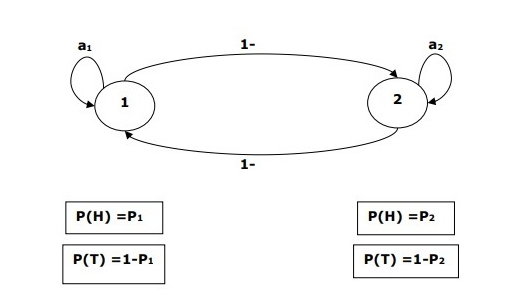
Так как задача состоит в том, чтобы найти такую последовательность C, при которой достигался бы максимум искомой вероятности, то знаменатель P(W) не влияет на итоговое решение. Для поиска вероятности последовательности делается предположение:

, где n – количество предшествующих вероятностей классов, которые нужно учесть при расчете текущей.

Для второго члена делается предположение, что вероятность отнесения слова к какому либо классу не зависит от вероятностей для других слов:

HMM

На основе этой идеи были созданы другие генеративные методы, которые используются для NER, POS и других целей машинного обучения. Одним из них является скрытая марковская цепь(HMM), которая представляет собой автомат, состояния которого могут характеризовать искомые классы. На выходе HMM видны только транслированные в слова значения состояний, когда вероятности переходов и сами переходы автомата скрыты от наблюдателя результата работы модели. По ходу работы автомата формируется такая последовательность классов, которая максимизирует . Пример автомата HMM с двумя состояниями:

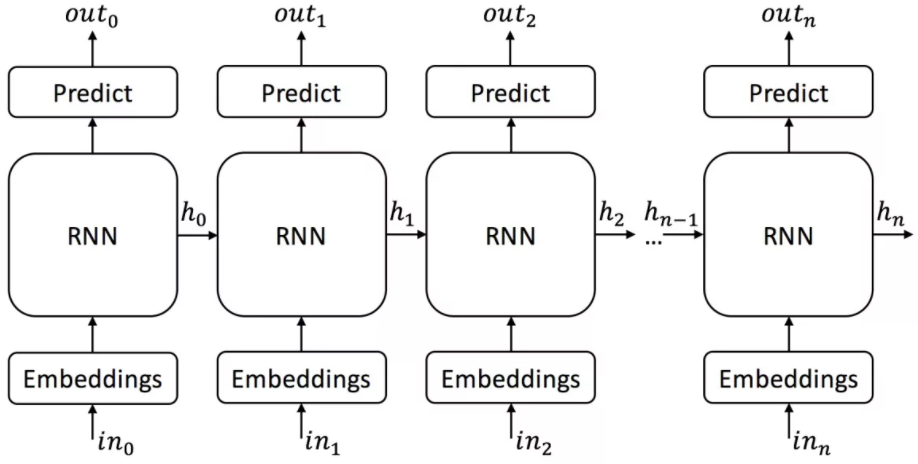


P(H) – вероятность сохранения состояния, P(T) – вероятность перехода.

RNN

Рекуррентная нейронная сеть(RNN – Recurrent neural network) и её модификации: модель с долгой-краткосрочной памятью (LSTM – Long short-term memory), Bi-LSTM(двунаправленная LSTM) представляют собой нейросети с «памятью».

Здесь основная идея состоит в том, что одни и те же матрицы весов переиспользуются для каждого последующего слова. Расчет вероятности текущего слова ведется при использовании информации о вероятностях, рассчитанных на предыдущих шагах (скрытое состояние h). Доля информации о текущем слове и скрытом состоянии, которая будет участвовать в расчете итоговой вероятности на данном шаге регулируется матрицами весов , , . - матрица весов, которая применяется ко входному вектору-слову – определяет степень влияния слово на скрытое состояние. определяет, как предыдущее скрытое состояние влияет на текущее и определяет влияние скрытого состояния на текущий результат. Функцией активации(при которой рассчитывается текущее скрытое состояние) обычно является гиперболический тангенс. Структура ячейки RNN переиспользуемой во времени:



Вывод

Решение задачи извлечения именованных сущностей из текста является полезным инструментом для поиска текстов, построения рекомендаций, создания формализованного представления. Рассмотрены методы решения такой задачи, в их числе: создание правил и использование статистических моделей(HMM, RNN). Данная задача, конечно, не является единственной, которую можно решить рассмотренными методами – она является одной из многих, для которых методы применимы.

Указанные в работе методы на данный момент не являются совершенными и работа в создании новых решений и можификации старых, ведется в настоящий момент. Одним из нововведений стало появление механизма внимания. В нем при анализе текущего слова статистически определяется, например, какая часть из предыдущей последовательности наиболее «интересна» для определения текущей вероятности. На основе идеи механизма внимания была создана архитектура Transformer, на основе которой Google создал решение BERT, в числе решаемых задач которого находится и задача NER.

Литература

1. Rule-based method https://en.wikipedia.org/wiki/Rule-based\_system
2. Rule based example https://github.com/Pilot-NER/Rule-based-Named-Entity-Recognition
3. NLP https://proglib.io/p/fun-nlp/
4. Лемматизация и стемминг https://webdevblog.ru/podhody-lemmatizacii-s-primerami-v-python/
5. POS https://www.tutorialspoint.com/natural\_language\_processing/natural\_language\_processing\_part\_of\_speech\_tagging.htm
6. Bayes https://proglib.io/p/bayes-theorem/
7. RNN https://python-scripts.com/recurrent-neural-network#pre-rnn
8. Embeddings <https://habr.com/ru/company/ods/blog/329410/>
9. BERT <https://arxiv.org/abs/1810.04805>
10. Transformer https://arxiv.org/abs/1706.03762