Санкт-Петербургский государственный университет

Программная инженерия

Ахмедов Гаджи Омар оглы

Модели обработки естественного языка в задачах информационного поиска

Отчёт по учебной (ознакомительной) практике

Научный руководитель: к.ф.-м.н., доцент кафедры СП Д.В.Луцив

Санкт-Петербург

2021

ОГЛАВЛЕНИЕ

Страница

АННОТАЦИЯ

В течение последних нескольких месяцев я продолжал концентрироваться на построении своих базовых знаний о NLP. И я считаю, что завершил эту фазу, закончив изучение большинства двух основных содержание книг и реализация связанных приложений на Python. Введение в информацию Retrieval и Python Natural Language Processing, эти две книги описывают разные аспекты NLP, одно для теоретического объяснения, другое для практического руководства, что в совокупности дает мне опыт обучения. Что касается первого, я многое узнал о фундаментальном описании относительного терминологии и теории. От базовой булевой модели поиска до построения инвертированного индекса, от математической логики SVM до реализации кластеризации рисует огромное изображение различных частей информационного поиска. И самое поразительное в этой книге то, что она уделяет большое внимание оптимизации алгоритма и корреляции между аппаратными производительность и использование приложений, что делает концепцию IR более распространенной.

По сравнению с предыдущим, обработка естественного языка Python в основном фокусируется на фактическом действие NLP. И я думаю, что эта книга хорошо разработана, поскольку она представляет собой инженерный путь, который мне очень удобно создавать приложение NLP с точными шагами. Когда дело доходит до алгоритм для применения NLP, он сравнивает два типа: система, основанная на правилах, и машина. Система обучения. Хотя машинное обучение — довольно продвинутая техника, которую почти каждый хотел бы освоить. Принести это в свою систему, в которых мы должны рассмотреть систему RB.

Кроме того, я глубоко изучаю Word2vec и Deep Learning, которые, как мне кажется, я собираюсь применить их в моей системе.

Помимо приложений, упомянутых в этих книгах, я обновляю Spark и Tensorflow.

Среды, на следующем этапе я планирую построить приложение на основе distributed computing platform (распределенного вычислительная платформа) с учетом большого объема данных.

Ключевые слова: NLP, IR, булевская модель поиска, Word2vec, Deep Learning, Spark, Tensorflow.

ГЛАВА 1. ИНФОРМАЦИОННЫЙ ПОИСК

В качестве моего первого руководства по IR «Введение в поиск информации» не только объясняются основные определение терминологии IR, но также показывает множество интересных моделей, идей и исследований, которые
глубоко укоренились в развитии IR.

1.1 Перевернутый индекс

Мы сосредоточимся на том, в чем разница между традиционным индексом и инвертированным индексом, как мы
можно построить инвертированный индекс и как можно быстрее вычислить пересечение списка проводок.

1.1.1 Традиционный индекс

Учитывая, что мы строим IR-систему для определения того, какой документ содержит несколько целевые слова. Если мы играем традиционным способом, это означает, что нам нужно построить терм документировать матрицу заранее. Смысл матрицы терминов-документов вполне очевиден: cij = 1
если термин t, представленный строкой i, фактически находится в документе d, представленном столбцом j. Однако, для разных стилей документов совершенно невозможно найти большое количество одинаковых конкретных слов в этих файлах, что указывает на то, что это займет огромное неиспользуемое пространство в памяти.

1.1.2 Инвертированный индекс

1.1.2.1 Способ изготовления

A. Соберите документы для индексации.

B. Разметить текст, превратив каждый документ в список токенов

C. Выполните лингвистическую предварительную обработку, создав список нормализованных токенов, которые являются терминами индексации.

D. Проиндексируйте документ, в котором встречается каждый термин, создав инвертированный индекс, состоящий из словарь и проводки.

Теперь предположим, что у нас есть словарь, содержащий два термина, соответствующие списку публикаций t1: p1[n] и t2 : p2[m]. Тогда пересечение p1[n] и p2[m] демонстрирует коллекцию документов в котором встречаются оба термина t1 и t2.

1.1.2.2 Более быстрое пересечение списка сообщений с помощью указателей пропуска

Несмотря на то, что инвертированный индекс значительно уменьшает объем памяти и увеличивает эффективность работы алгоритма по сравнению с традиционным способом, мы все еще можем улучшить производительность алгоритм, внеся расширение в структуру данных списка сообщений.

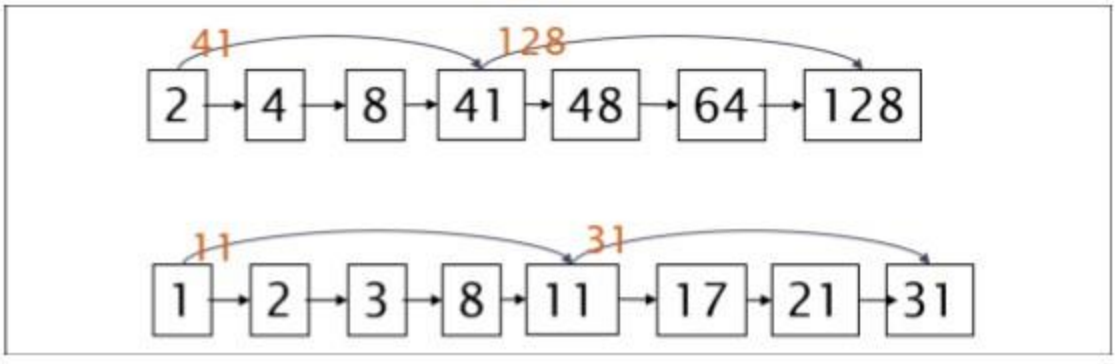

Рисунок 1.1 Указатель пропуска

В этом случае мы не будем перебирать каждый элемент один за другим, чтобы соответствовать одному и тому же идентификатору

документа, вместо этого мы сравним указатель списка пропуска и решим, начинать ли с указателя пропуска или текущего
элемент, как показано на рисунке 1.1.

1.2 Машина опорных векторов

SVM уже давно используется в классификации анализа данных. Теперь, когда дело доходит до NLP, его также можно применять для вычисления сходства разных слов.

1.2.1 Линейный классификатор

Представьте себе, что в геометрическом пространстве мы хотели бы найти поверхность решений для разделения членов, которые
относятся к разным классам. Для определения маржи классификатора необходимо найти ближайшие точки, которые закрыты для классификатора, эти точки агрегируют как опорный вектор.

Обозначим член пересечения b и вектор нормали ω~, перпендикулярный гиперплоскости определение гиперплоскости решения. Тогда для любой точки ~x на гиперплоскости она имеет ω~T~x + b = 0.

Соответственно, тогда линейный классификатор равен f(~x) = sign(ω~T~x + b). Теперь определим функциональный запас как yi(ω~T~xi + b). Согласно евклидову расстоянию точка на гиперплоскости, близкая к ~x, так как ~x0, ~x0 = ~x−yr ω~ |ω~| . Поскольку оно также удовлетворяет условию ω~T~x0 + b = 0, мы можем найти r = yω~T~x+b |ω~| . Для удобства решения больших SVM мы требуем, чтобы функциональный запас всех точек данных был не меньше 1, что означает, что yi(ω~~xi + b) ≥ 1. И нахождение подходящих ω~ и b, чтобы сделать 1 2ω~Tω~ минимизируется для всех (~xi,yi), yi(ω~T~xi +

b) ≥ 1 является конечной целью. Здесь мы используем двойственную задачу и Лагранжа для решения этой задачи квадратичной оптимизации.

Будем писать $L(\omega,b,\alpha) = 1\ 2\|\omega\|2 - PN\ i=1\ \alpha iyi(\omega xi + b) + PN\ i=1\ \alpha i,(\alpha i \geq 0)$. В итоге мы могли бы получить уравнение гиперплоскости as$PN\ i=1\ \alpha * i\ yi <$ xi,xj > $+b* = 0$.

1.2.2 Расширения модели SVM

Возникновение классификации мягких полей происходит из-за незнания нескольких странных шумовые документы, чтобы лучше отделить большую часть данных. В этом случае допустим решение лица возможность сделать несколько ошибок, и из-за этого мы должны платить цену за каждый неправильно классифицированный пример с переменной ζi. И мы перепишем нашу задачу оптимизации: нахождение $\omega\sim,band\zeta i \geq 0$, таких, что $1\ 2\omega\sim T\omega\sim$ $+CPi\ \zeta i$ минимизируется для всех $(\sim xi,yi),yi(\omega\sim T\sim xi\ +b) \geq 1-\zeta i$. Тогда для функции Лагранжа $L(\omega,b,\zeta,\mu) = 1\ 2\|\omega\|2\ + CPN\ i=1\ \zeta i - PN\ i=1\ \alpha i[yi(\omega xi + b)-1$ $+ \zeta i]-PN\ i\ =1\ \mu i\zeta i,(\alpha i \geq 0,\mu i \geq 0)$. Гиперплоскость для мягкой границы равна $b*$ $= yi - PN\ i=1\ \alpha * i\ yj <$ xi,xj >.

ГЛАВА 2. ОБРАБОТКА ЕСТЕСТВЕННОГО ЯЗЫКА

Строго говоря, для Введение в поиск информации демонстрируется в основном технология An, связанной с IR, она имеет связь с NLP, но все же не охватывает столько, сколько
специальный учебник по NLP.

2.1 Горячие векторы

Горячий вектор, показывающий, как результат передачи от переменных к двоичному выражению имеет
применялся во многих сценариях. В NLP однократный вектор — это вектор, используемый для выражения каждого слова.
уникальным способом, который отличается от других слов в словаре, и существует только один
1 в векторе.
В традиционном семантическом анализе NLP с помощью горячих векторов мы можем просто количественно определить
разные слова. Однако возникает серьезная проблема, когда мы хотим искать относительные слова по
эта форма. Например, предположим, что мотель = [0 0 0 0 1 0 0], гостиница = [0 0 1 0 0 0 0], когда мы вводим
«Мотель Сиэтл», мы также хотели бы сопоставить документ, содержащий «Отель Сиэтл», напротив, мы
не может этого сделать, потому что эти два вектора ортогональны.

2.2 Word2vec

Значение слова определяется словами, которые часто встречаются рядом. Так же, как J.R.Firth

сказал раньше: «Вы узнаете слово по компании, которую оно держит».

2.2.1 Введение

Word2vec — продукт дистрибутивной семантики, одна из самых успешных идей.

современного статистического НЛП. Распределительная семантика — это область исследований, которая концентрируется на количественных выявление и категоризация семантического сходства между лингвистическими единицами на основе свойств их распределения.

Word2vec как неконтролируемая техника полностью достигает цели.

2.2.2 Моделирование

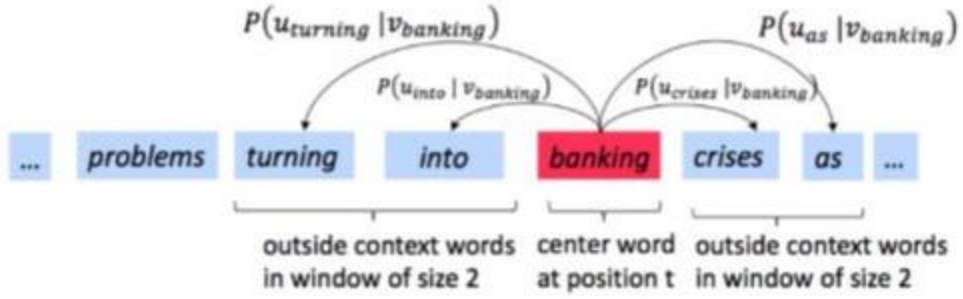

Рисунок 2.1 Предсказание слова

Как мы видим, условная вероятность является основой этой теории. Затем для каждой позиции t = 1,...,T, заданное центральное слово wt, предсказать слова контекста в пределах окна фиксированного размера m, L(θ) = QT t=1Qm≥j≥−m p(wt+j|wt;θ). Теперь мы получаем целевую функцию J(θ) = −1 T log(L(θ))PT t=1Pm≥j≥−m p(wt+j|wt;θ). Наша цель состоит в том, чтобы минимизировать функцию объекта, которая в равной степени максимизирует точность прогнозирования. Мы используем функцию softmax для вычисления p(wt+j|wt;θ). Функция softmax pi = exp(xi) Pn j=1 exp(xj) отображает произвольные значения xi в распределение вероятностей pi.

Достигнув производной целевой функции по вектору центрального слова vc, мы получаем ∂ ∂vc logp(o|c) = uo −Pv x=1 p(x|c)ux. Эта разница, оказывается, точно дает нам наклон, в котором мы должны идти и изменение представления слова, чтобы улучшить способность нашей модели прогнозировать.

2.2.3 Нейронная сеть Word2vec

Word2vec использует нейронную сеть для обучения. Существует один входной слой, который имеет столько же нейроны как есть слова в словаре для обучения. Второй слой — это скрытый слой, последний слой — это выходной слой, который имеет то же количество нейронов, что и входной слой.

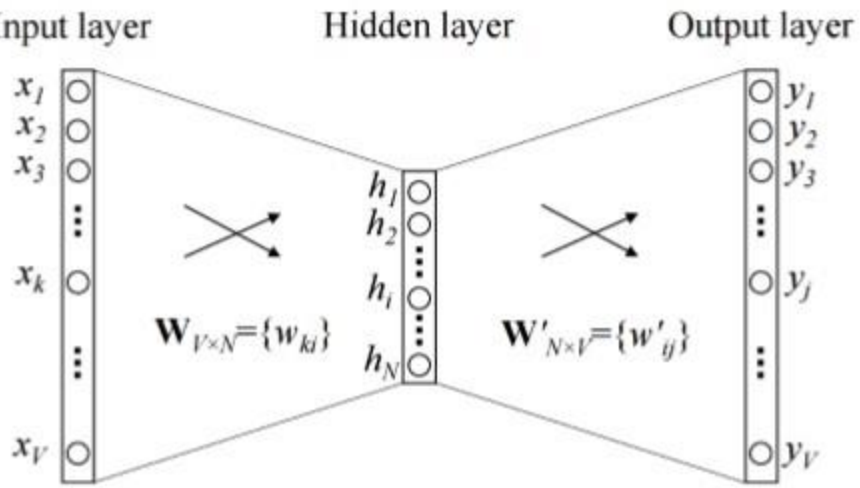

Рисунок 2.2 Модель CBOW

Word2vec имеет две разные версии: непрерывный пакет слов и Skip-Gram. Структура изменения нейронной сети с различными версиями. На рис. 2.2 мы показываем модель CBOW, в которой контекст представлен

несколькими словами для заданных целевых слов, в то время как модель SG переворачивает

использование целевых слов и контекстных слов.

ГЛАВА 3. РАЗВЕДКА ПЛАТФОРМЫ

Учитывая большой объем данных, с которыми мне предстоит столкнуться в процессе обучения, я снова обновляю среду Spark, чтобы она соответствовала высоким требованиям эксплуатацииSpark работает с устойчивыми распределенными наборами данных, которые представляют собой один тип последовательного типа объекта.
объект. Эти RDD будут развернуты на разных подчиненных узлах, когда они будут работать в рамках распределенной вычислительной структуры.
А операции с СДР можно разделить на две формы: преобразование и действие. При использовании преобразования нам не нужно фактически запускать программу, вместо этого нам просто нужно получить логику работы. Когда есть необходимость показать нам результат, то это вызовет действие.

Кроме того, Spark также имеет уникальную стратегию кэширования. После вызова части данных он сохранит
данные в кеш, чтобы следующий вызов был более быстрым. Между тем, у него есть некоторые другие вспомогательные
символы, которые помогают усилить способность больших данных распределенных вычислений.

ГЛАВА 4. РЕЗЮМЕ И ОБСУЖДЕНИЕ

Оглядываясь назад, можно сказать, что теоретическая часть всегда была изюминкой моей работы. И различные теорий, идей, алгоритмов чрезвычайно много, и я могу не распознать истинное значение некоторых приложений. Поэтому я хотел бы продолжить изучение математическое объяснение NLP, а также перейти к построению структуры моего проекта в следующем шаге.

A: Набросать первоначальный план анализа настроений.

B : Выбрать подходящий язык, платформу и начните сборку системы.

C : Провести глубокое исследование критической области.

D : Развить расширяемости системы.

ИСПОЛЬЗОВАННАЯ ЛИТЕРАТУРА

Christopher D.Manning, Prabhakar Raghavan, Hinrich Schutze. (2009). An Introduction to Infor

mation Retrieval, England: Cambridge.

Jalaj Thanaki. (2017). Python Natural Language Processing. Packt Publishing.

Online course: (CS224n) Natural Language Processing with Deep Learning.