1. Лекции

1.1 28.09.2023

При фильтрации на основе контента мы опираемся на описание. Из контента извлекается и обрабатывается информация. Два этапа: подготовка данных (автономное обучение, весь контент прогоняется через анализатор и получаем формальную структуру); пользователь взаимодействует с контентом и мы получаем информацию, на основании которой мы получаем профиль пользователя.

Три основных блока:

- Анализатор создает профиль системы;
- Блок создания профиля пользователя;
- Блок извлечения нужных элементов (сравнивает профиль пользователя и профиль элемента и подбирает подходящий).

1.1.1 Блок анализатор контента

Анализатор контента – блок, который формализует описательные данные в формальные, обработку которых уже можно автоматизировать.

На этом этапе мы определяем, что является контентом и какие признаки мы хотим выделять.

Мета-данные – это данные о данных. Спасибо. Если элементами являются фильмы – то мета-данные – это жанр, актеры и прочее. Можно выделить два основных типа: факты и теги. Факты – объективная, неоспоримая информация об объекте (год выхода, например). Теги – ключевые слова – это оценка субъективная, кто-то скажет, что экшн, кто-то, что классическая английская комедия и прочее. Теги могут быть простыми и широкими, а могут быть узконаправленная. Одна из проблем тегов – это разное мнение людей и разное их выражение.

Центральная проблема в обучении рек-системы – это получение мета-данных.

Входные данные – описание документов (объектов). В общем случае – свободный текст на естественном языке. Самый простой способ – считаем частотность слов. Можно слова объединять в темы.

Самый простой способ обработать текст – собрать по нему словарь. Ок. Но как только текст токенизируется, мы теряем часть информации. Второй

момент связан с тем, что в русском языке есть зависимость смысла от расположения слов в предложении.

Также не забываем, что нужно использовать стоп-слова, набор их зависит от предметной области и задачи. Затем советуют убрать минимумы и максимумы, но с минимумами неоднозначно: если слово встречается 1 раз, то аналогию мы навряд ли найдем, однако оно может очень хорошо помочь в определении специфики предмета.

Вторая задача словаря – разобраться с лексическими формами. $pussy == kitty$. Простой способ – стэмминг, отрезаем конец слов, он чаще всего отвечает за изменение формы, однако под раздачу часто отрезается суффикс, коты на равны котлетам. Второй подход – лемматизация. И кот, и котенок, и кошка сведутся к общему знаменателю. Но способ более трудоемкий.

tf-idf – крута. В модели два показателя: частота слов в документе и обратная частота документа. Обычно частота – это количествона на общее количество, а в обратном – обратно, етить-колотить.

$$tf(x,y) = \frac{n_x}{N_y} \tag{1}$$

n – сколько раз слово встретилось; N – сколько всего слов в документе.

В самых простых случаях без x и y – будем получать меньшую точность модели.

Есть еще логорифмическая мера:

$$tf(x,y) = log_{10}(1 + n_{x,y}) \tag{2}$$

$$IDF = \frac{numberofdocuments}{numberofdocumentswithwordx} \tag{3}$$

Эти величины перемножаются. Чтобы уравнять их от IDF берут десятичный логорифм.

$$IDF(x,y) = log_{10}(\frac{numberofdocuments}{numberofdocuments_with_word_x})//left - right \tag{4}$$

$$\textbf{tf-idf}(x,y) = tf(x,y) \cdot idf(x,y) \tag{5}$$

Если какое-то слово имеет высокий вес – значит у него высокая частота в одном документе и низкая частота во всем документе. Чем меньше вес – тем более распространенный.

100 слов. кот = 12 раз. $tf(cat) = 0.12$.

10 000 слов. 300 котов. $idf(cat) = log_{10}\frac{10000}{300} \approx 1.52$. $w(cat) = tf(cat) \cdot idf(cat) \approx 0.18$.

Тыры-пыры мы тут уже обсуждаем LDA. word2vec как-то по итогу обернулся в LDA – Latent Dirichlet Allocation (скрытое распределение Дирихле). То есть анализируем не слова, а косвенные признаки и обобщения.

Дирихле отсылает нас к математическому аппарату метода.

Берем слова документа и строим по ним корпус тем. Корпус – совокупность документов. Задача LDA – обнаружить темы в документе и выполнить автоматическую классификацию документов по этим темам. Классификация – подходит документ или нет. Тема – набор терминов, отдельных слов или фраз, которые вместе описывают тему. Перед запуском алгосика лучше причесать текст, конечно же.

В подходе LDA используется 4 предположения:

- Смысл текста определяется набором ключевых слов тех или иных тем;
- Некоторые термины могут быть неоднозначными, поэтому слово будет относить к теме на процент;
- Большинство документов разные темы встречаются с разной частотой;
- В рамках темы определенные термины используются чаще, чем другие.

пупа и лупа пошли получать зарплату, однако в бухгалтерии все перепутали. По итогу пупа получил за пупу, а лупа за лупу. Ой, то есть ничего не перепутали.

LDA – генеративная модель. А я дегенеративная модель. Хе. Хе. Хе.

LDA – статистическая модель совместного распределения вероятностей.

$$P(X, Y)$$

P(X | Y=y) (6)

X – наблюдаемая переменная, Y – целевая переменная. Нужно перейти

от наблюдаемой X к метке Y.

Общая идея алгоритма: на вход подаются документы и K – количество тем, которые надо выделить. Результатом работы алгоритма будет список из K тем, каждая тема представляет собой вектор, компоненты которого показывают с какой вероятностью термин встречается в этой теме. Второй результат – список векторов, компоненты которых – вероятность отношение i-го документа к теме.

Алгоритмов выделения тем несколько. Мы рассматриваем алгоритм генерации выборки Гиббса.

Каждый документ предобрабатываем, анализируем только слова, а не расположение. Задача создания тем – установить связи между словами и темами, между словами и документами. Слова, относящиеся к одной теме чаще всего будут в одном документе.

Схема выборки Гиббса начинается с случайной установки связей. Затем для каждого слова определяется вероятность принадлежности к той или иной теме. Сопоставляя все слова всех документов мы получим вероятности встреч слова в теме.

Первый настроечный коэф – количество тем. Если маленькое, то будет бо-бо. Если очень большое, то тоже. Ну надо же.

Второй – порог, по которому отсекаем слова, принадлежащие и непринадлежащие темам. При отсечении шума неизбежна потеря информации.

Я хочу домой.

Откуда взят хорошее описание? Хороший вопрос, никак, пошел в жопу!

В общем случае берут некоторое количество тем с потолка. Запускают, смотрят, перекрываются ли полученные темы.

Еще два параметра – α, β. – k-компонентный вектор, отвечающий за выраженность тем в документах. При высоком – доки ближе друг к другу. При низких – разделяем узко-специализированные тексты. Чаще берут $\frac{50}{k}$.

β большое – в теме больше терминов. Маленькое – хз. Лучше всего ≈ 0.01.

Мы разобрались с контентом.

Создание профиля пользователя

Если есть LDA, то смотрим, что понравилось юзеру и сравниваем вектора. Темы здесь нам помогут.

В общем случае алгосик: взять все потребленные элементы, для каждого элемента LDA ищет похожие, рассчитываем оценку на основе сходства, отсортировать по убыванию оценки и релевантности. Но можно и транспонировать, то есть создать вектор LDA для пользователя.

В модели tf-idf все проще, у нас есть векторы с тегами и фактами. Есть список, что нравится пользователю, по нему составляем профиль пользователя, можно ввести веса, связанные с оценками, по итогу получаем вектор вкусов пользователя. Лучше всего еще применить нормализацию. Если наш пользователь любит квашеную капусту и жрет торты, навряд ли он их ест вместе.

Плюсы и минусы фильтрации контента. Плюсы: можно рекомендовать что-то после первой оценки или выбора; она менее чувствительно к глобальной популярности. Минусы: те оценки, которые мы вычисляем, приобретают общую силу; выдача результатов строго определена, никаких неожиданностей; ограниченное содержание.

Там формула, но нам важно только, что K – количество тем, V – количество слов в словаре, M – количество документов, $N_d(d = 1..M)$ – количество слов в документе d, N – общее количество слов, $_k, k = 1..K$ – вес темы k в документе, $\beta_w, w = 1..V$ – вес слова w в теме. $\phi_{kw}, k = 1..K, w = 1..V$ – вероятность нахождения слова в документе. Еще “тэта”, вероятность какая-то. Если че ϕ_kподчиняется$Dirichlet(\beta)$, а тета по альфе.

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

$$p(d, w) = \sum_{z\text{принадлежит}Z} \text{тета} \cdot \phi \cdot p(d) \tag{7}$$

1.1.2 05.10.23

Пора сдавать лабки. Всего их 7.

Матричная векторизация

Попытаемся скрестить ежа с ужом.

Для работы потребуется матрица оценок. Есть явные и неявные оценки, помним. Процесс называется матричной факторизацией. Блять, что тут происходит, нихуя не понятно.

При разложение матрицы на произведение трех из трех разных матриц можем получить три группы информации. Эта тема и дает скрытые факторы,

которые мы хотим проанализировать. Самим раскладывать нахрен не надо, взять готовое.

Самая главная проблема – пустота в клетках, потому что для нас пустая клетка не равна нулю. Обработка этих клеток и является из основных проблем. Тут есть модификации SVD++, funcSVD. Матричная факторизация – эо метод обнаружения скрытых факторов в рамках коллабаративной фильтрации.

Чем меньше в матрице данных, тем быстрее считаем. Общая идея в факторизации заключается в том, что мы сжимаем пространство так, чтобы расстояние менялось пропорционально, дальнее остается дальним, ближнее остается ближним. К тому же мы предполагаем, что в данных есть скрытый смысл.

Факторизация позволяет снизить размерность вычислений. Снизив ее, мы можем выделить области.

Есть векторы вкусов пользователя и есть векторы объектов. Все вместе – матрица оценок. Для удобства пользователи – строки, объекты – столбцы. n пользователей и m столбцов. На первом этапе пустоты матрицы заполняем нулями. $R = U \cdot V$, U – матрица пользователей, V – матрица объектов. $R(n \cdot m), U(n \cdot d), V(d \cdot m)$.

$$r_{ij} = \sum_{k=1}^{d} U_{ik} V_{kj} \tag{8}$$

Классический метод СВД чего боже где я

Матрицу будем раскладывать на 3 произведения. $R_{n \cdot m} = U_{n \cdot n} \cdot \Sigma_{n \cdot m} \cdot V^T_{m \cdot m}$. $U^T U = I_n, V^T V = I_m$.

Веса в матрице выстраиваются по невозрастанию, среди множества чисел можно выделить наиболее важных d, а остальные принудтельно зануляем. Все получается круто, но, к сожалению, теряем на этом точность.

$R'_{n \cdot m} = U'_{n \cdot d} \cdot \Sigma'_{d \cdot d} + V'^T_{d \cdot m}$