

ВНИМАНИЕ! ЭТО ПРЕДВАРИТЕЛЬНАЯ ВЕРСИЯ!!!!

Алгоритмы для рекомендательной системы: технология LENKOR¹

Дьяконов А.Г.

ф-т ВМК, Московский государственный университет

имени М.В. Ломоносова, Россия

djakonov@mail.ru, <http://alexanderdyakonov.narod.ru>

Описаны алгоритмы, которые заняли первые места на Международном соревновании «ECML/PKDD Discovery Challenge 2011 (VideoLectures.Net Recommender System Challenge)» по написанию рекомендательной системы для ресурса VideoLectures.net (научного репозитория лекций). Соревнование состояло из двух независимых подконкурсов. В первом подконкурсе необходимо по одной просмотренной лекции рекомендовать новую лекцию из списка недавно выложенных на сайт, для которой известно только подробное описание и нет статистики популярности. Во втором подконкурсе надо рекомендовать лекции, основываясь на статистической информации о популярности, но представленной не в классическом виде, а усреднённом.

Введение

Задача рекомендательных систем – предлагать пользователям некоторого ресурса услуги (или товары), которые могут быть им полезны в будущем, что облегчает пользователям проблему выбора. Подобные рекомендательные системы существуют практически на всех крупных ресурсах: на Grouplens.org и Netflix.com (для выбора фильмов), Youtube.com (для выбора видео), facebook.com (для выбора друзей), Amazon.com (для выбора товаров), Last.fm (для выбора музыкальных треков) и т.д. Математические методы, которые применяются при разработке рекомендательных систем, можно разбить на две группы: методы коллаборативной фильтрации (collaborative filtering, [a1]) и контентные методы (content-based, information filtering, [a2]). Первые используют статистику поведения пользователей (например, рекомендуют товары и услуги, которые были интересны для похожих пользователей), а вторые – описания товаров и услуг (например, рекомендуют товары из той же категории, ценовой группы, сопутствующие товары и т.д.). Естественно, возможно одновременное использование методов двух групп (hybrid prediction), выделяют также рекомендации, основанные на априорном знании потребностей пользователей (knowledge-based), см. [a3].

Ресурс VideoLectures.net [s1] является открытым репозитарием видеолекций, прочитанных ведущими преподавателями и учёными на конференциях, летних научных школах, фестивалях науки и т.д. Конкурс по разработке алгоритмов «ECML/PKDD Discovery Challenge 2011 (VideoLectures.Net Recommender System Challenge)» [s2] был проведён с целью улучшить существующую рекомендательную систему ресурса, стимулировать разработку новых рекомендательных алгоритмов и предоставить данные

¹ Работа выполнена при финансовой поддержке РФФИ (код проекта: 12-07-00187) и гранта Президента РФ (МД-757.2011.9).

реальной прикладной задачи научному сообществу. Конкурс являлся (очередной ежегодной) соревновательной частью конференции ECML-PKDD 2011 [s3], прошедшей в Афинах в сентябре 2011 года, а также базировался на платформе TunedIT [s4] (предоставляющей удобный интерфейс для проведения соревнований по анализу данных).

Ниже мы опишем постановки решаемых задач, а также алгоритмы, предложенные для их решения, которые заняли первые места во всех подконкурсах соревнования.



Рис. 1. Демонстрация лекции на сайте VideoLectures.net с пометками признаков.

Первая задача «холодный старт» (cold start)

Для некоторого множества лекций ресурса VideoLectures.net заданы их описания: номер лекции (идентификационный номер), язык лекции, категория лекции (например «Machine Learning» или «Biology», всего 350 категорий), число просмотров, дата выкладывания на сайт, автор лекции (его номер, есть также его имя, e-mail, сайт в Интернете), название лекции (предложение на языке лекции), описание лекции (небольшой текст). Аналогичная информация имеется по событиям, к которым относятся лекции (конференция, на которой была прочитана лекция, школа-семинар, цикл лекций и т.д.). Здесь мы не перечисляем всей информации, которая была задана по каждой из лекций, её можно найти на сайте соревнования [s2] (например описания слайдов лекций, дата съёмки и т.д.), поскольку она не была использована в финальной версии алгоритма решения. Отметим также, что некоторые данные для лекций не известны (например, не для всех лекций даны описания). Кроме контентной информации задана также статистические данные: известно сколько

пользователей одновременно просмотрели i -ю и j -ю лекции для всех i, j (если это число больше единицы).

Множество описанных лекций разбито на два непересекающихся подмножества: 6983 «старые лекции», которые были опубликованы на сайте до 1 июля 2009 года (по ним известна вся статистика) и 1122 «новые лекции», которые выложены на сайте после этой даты (по ним не известна информация о просмотрах). Есть также контрольная выборка – это подмножество множества старых лекций. Для каждой лекции из контрольной выборки необходимо предложить список рекомендуемых 30 новых лекций (это рекомендация новых лекций пользователю, который посмотрел одну старую). Такая задача моделирует режим «новый пользователь» – «новая лекция», при котором человеку, впервые зашедшему на сайт, рекомендуют что-то из новинок.

Опишем функционал качества, который использовался организаторами для оценки и сравнения алгоритмов [s2]. Пусть алгоритм рекомендует лекции с номерами $r_1, \dots, r_R$, а верная рекомендация (что из новинок чаще интересует пользователей ресурса после просмотра конкретной «старой» лекции) – лекции с номерами $s_1, \dots, s_S$, тогда качество такой рекомендации вычисляется по формуле

$$\frac{1}{|Z|} \sum_{z \in Z} \frac{|\{r_1, \dots, r_{\min(S, R, z)}\} \cap \{s_1, \dots, s_{\min(S, R, z)}\}|}{\min(S, R, z)},$$

где Z – некоторое множество целых чисел. Для первой задачи организаторы использовали $Z = \{5, 10, 15, 20, 25, 30\}$. Качество алгоритма вычислялась как среднее арифметическое по качеству всех рекомендаций. Ясно, что функционал поощряет угадывание множества интересных новинок и порядка на этом множестве, поскольку лекции $s_1, \dots, s_S$ упорядочены по числу просмотров.

Алгоритм решения первой задачи

Пусть некоторую информацию о лекции можно записать в виде n -мерного вектора $f = (f_1, \dots, f_n)$. Например, если n – число авторов всех лекций, то бинарный вектор f описывает авторов конкретной лекции: единицы помечают номера соответствующих авторов (такой вектор логично назвать «характеристическим вектором авторов лекции»). Аналогично можно описать язык лекции, её категорию и т.д. Естественно, размерности векторов при разных описаниях (отвечающих разным типам информации) различны. В каждом случае можно оценить, насколько похожи лекции по представленной информации. Например, для i -й лекции, представленной вектором $f(i) = (f_1(i), \dots, f_n(i))$, и j -й лекции, представленной вектором $f(j) = (f_1(j), \dots, f_n(j))$ их близость оцениваем с помощью измененной косинусной меры [s5]

$$\langle f(i), f(j) \rangle = \frac{f_1(i)f_1(j) + \dots + f_n(i)f_n(j)}{\sqrt{f_1(i)^2 + \dots + f_n(i)^2 + \varepsilon} \sqrt{f_1(j)^2 + \dots + f_n(j)^2 + \varepsilon}}. \quad (1)$$

Замечания. Подобные изменения « $+\varepsilon$ » (см. (1)) будут встречаться и дальше. Они связаны с предотвращением деления на ноль без потери скорости вычисления в среде MATLAB (поэтому не использовалось простое сравнение знаменателя с нулём). Кроме того, добавление подобных новых параметров меняет качество алгоритма. Оптимизация качества привела к выбору $\varepsilon = 0.01$ в финальной версии алгоритма.

Идея алгоритма: для заданной лекции из контрольной выборки пройти по всем новым лекциям и вычислить близость к каждой, просуммировав (с какими-то коэффициентами) значения (1) для всех представленных видов информации (категории, авторы, язык и т.д.). Сразу отметим главную модификацию алгоритма, которая существенно улучшает качество. Вместе с близостью к рассматриваемой лекции из контрольной выборки необходимо учитывать близость к похожим лекциям с точки зрения поведения пользователей, ниже формализуем эту идею.

Замечание. Учёт похожих лекций – естественный метод борьбы с дефицитом информации. Например, если «старая» лекция лежит в категории «Биология», а среди «новых» нет лекций из этой категории, то можно рекомендовать из «похожей», например из «Химии», но никак не из «Криминологии». Такую «похожесть» и должен установить алгоритм, основываясь на информации, из каких разных категорий лекции просматриваются одними и теми же пользователями.

Пусть множество старых лекций индексируется номерами из I , пусть $f(i)$ – вектор описания i -й лекции (по фиксированному типу информации), пусть m'_{ij} – оценка близости i -й и j -й лекции с точки зрения поведения пользователя (определено ниже). Тогда пусть

$$f'(i) = \sum_{j \in I} \left(m'_{ij} \frac{f(j)}{\sqrt{f_1(j)^2 + \dots + f_n(j)^2 + \varepsilon}} \right)$$

и близость к новой t -й лекции вычисляем с помощью суммы $\langle f'(i), f(t) \rangle$ по всем видам информации. Опишем, как вычисляются значения m'_{ij} . Пусть L – число лекций, m_{ij} – число пользователей, которые просмотрели и i -ю лекцию и j -ю лекцию при $i \neq j$, и m_{ii} – число пользователей, которые просмотрели i -ю, делённое пополам (такое определение получилось в результате оптимизации качества алгоритма). Тогда

$$m'_{ij} = \frac{m_{ij}}{\sum_{t=1}^L m_{it}}. \quad (2)$$

Замечание. Смысл значения (2) достаточно ясен. Если бы m_{ii} были равны нулю, $i \in \{1, 2, \dots, L\}$, $j \in \{1, 2, \dots, L\}$, то (2) – это оценка вероятности с которой случайный пользователь посмотрел j -ю лекцию при условии, что он точно посмотрел i -ю. Ненулевые элементы m_{ii} нужны, чтобы учитывать также близость к i -й лекции, а не только к тем, которые похожи на неё с точки зрения поведения пользователей.

Ниже перечислим виды информации, по которым оценивалась близость. Для каждого вида будем указывать вектор $\gamma = (\langle f'(i), f(j_1) \rangle, \dots, \langle f'(i), f(j_r) \rangle)$ с соответствующим индексом, где $J = \{j_1, \dots, j_r\}$ – множество номеров новых лекций.

1. Близость категорий – γ_{cat} .

Здесь $f(j)$ – характеристический вектор категорий лекций, т.е. бинарный вектор, в котором t -й элемент равен единице, если лекция принадлежит t -й категории.

2. Близость авторов – γ_{auth} .

Здесь $f(j)$ – характеристический вектор авторов лекций, т.е. бинарный вектор, в котором t -й элемент равен единице, если t -й автор является автором лекции.

3. Близость языков – γ_{lang} .

Здесь $f(j)$ – это характеристический вектор языка лекции, в котором первая координата, соответствующая английскому языку, положена равной единице (чтобы сделать все лекции похожими на английские по соответствующей тематике, поскольку лекции на английском языке смотрят практически все пользователи ресурса).

4. Близость названий – γ_{dic} .

Сначала анализируются все слова, которые входят в названия и описания лекций и приводятся к общей основной форме [s5] (мы использовали стеммер Портера [s6], [s7]). Отметим, что в состав слов не включались спецсимволы (скобки, запятые, знаки арифметических операций, знак доллара и т.д.), но стоп-слова не удалялись (удаление практически не меняло качества). Название каждой лекции описывается вектором $(h_1, \dots, h_w)$, в котором h_i – число слов с i -й основной формой. Затем делается преобразование похожее на idf:

$$f_i = \frac{h_i}{\sqrt{w_i + \varepsilon}}, \quad (3)$$

где w_i – сколько слов с i -й основной формой есть вообще в названиях и описаниях (с учётом кратности). Именно такие векторы $(f_1, \dots, f_w)$ и используются для итогового вычисления.

5. Близость названий, описаний, названий и описаний событий – γ_{dic2} .

Каждая лекция имеет название, описание, название соответствующего события и описание события (если информации о событии нет, то считаем, что название и описание события совпадают с названием и описанием лекции). Всё это

объединяется в один текст, он описывается вектором $(h_1, \dots, h_W)$, а дальше порядок действий такой же, как и в предыдущем пункте.

Замечание. Отход от классического $tf \cdot idf$ преобразования произошёл в результате решения задачи оптимизации при различных нормировках в (3). Использовалось также деление на логарифм, на саму частоту, на её квадрат и т.д. При этом качество менялось на 2–5%.

Для получения решения алгоритм строит вектор

$$\gamma = 0.19 \cdot \sqrt{0.6 \cdot \gamma_{cat} + 5.6 \cdot \gamma_{auth}} + \sqrt{4.5 \cdot \gamma_{lang} + 5.8 \cdot \gamma_{dic} + 3.1 \cdot \gamma_{dic2}}.$$

Здесь корень извлекается поэлементно. В качестве рекомендации алгоритм даёт лекции, которые соответствуют наибольшим значениям координат в векторе $\gamma = (\gamma_1, \dots, \gamma_N)$. Вид решения определяется технологией решения прикладных задач «LENKOR», развиваемой автором, основные этапы применения которой:

1. Выделение различных видов информации, описание способов вычислений близости по каждому виду.
2. Формирование линейной комбинации функций близости, настройка коэффициентов (методом покоординатного спуска). При этом автоматически определяются «ненужные» виды информации (соответствующие слагаемые входят с нулевыми весами)
3. «Деформирование» комбинации (попытка построить нелинейную формулу решения путём перебора различных алгебраических выражений), настройка коэффициентов.

На третьем этапе приходится иметь дело с относительно небольшими выражениями (часть функций близости отсеивается на втором этапе), что обеспечивает возможность перебора нелинейных выражений различных типов. В данной задаче оптимальным выражением оказалась линейная комбинация корней, которая, впрочем, не сильно улучшает линейное решение (на 2%).

В решении, которое автор выложил на сайт [s2], внесено ещё одно изменение: вектор $\gamma = (\gamma_1, \dots, \gamma_N)$ преобразуется в вектор

$$\left(\gamma_1 \cdot \left(1 + \delta \frac{t_{\max} - t_1}{t_{\max} - t_{\min}} \right), \dots, \gamma_N \cdot \left(1 + \delta \frac{t_{\max} - t_N}{t_{\max} - t_{\min}} \right) \right),$$

где t_j – время выкладывания на сайт j -й новой лекции, $t_{\min}$ – минимальное среди всех этих времён новых лекций, $t_{\max}$ – максимальное (вычислялось в днях). Такая поправка учитывает условия соревнования: поскольку оценка происходит по имеющейся статистике (т.е. сколько пользователей посмотрели новую лекцию при условии, что посмотрели лекцию из контрольной выборки), то важно не только насколько популярна лекция, но и сколько она была доступна для просмотра. В табл. 1 представлена зависимость от δ .

δ	качество
0.05	36.24%
0.07 (выбрано)	37.28%
0.09	37.24%

Табл. 1. Качество алгоритма при различных δ .

Алгоритмы	Предфинальный результат	Итоговый результат
Наш алгоритм	37.281%	35.857%
Второе место (E. Spyromitros-Xioufis, E. Stachtiari [s8])	31.063%	30.743%
Третье место (M. Možina [s9])	30.661%	27.684%

Табл. 2. Результаты алгоритмов-победителей первого подконкурса.

Описанный алгоритм занял первое место среди 62 участников с результатом 35.857% и достаточно солидным отрывом от второго места (см. табл. 2). После загрузки и обработки данных (занимает около 1 часа, но для построения рекомендательной системы может быть выполнено один раз) время работы алгоритма для решения задачи соревнования составило 17.3 секунд на компьютере HP p6050ru Intel Core 2 Quad CPU Q8200 2.33GHz, RAM 3Gb, OS Windows Vista в среде MATLAB 7.10.0. При этом сделано 5704 рекомендации по 30 лекций в каждой. Словарь состоит из 35664 основных форм. Всего лекций и событий: 8624.

Ниже представим граф алгоритма (изображение вычислительного процесса, принятое в современных системах анализа данных, таких как RapidMiner и Weka), см. рис. 2.

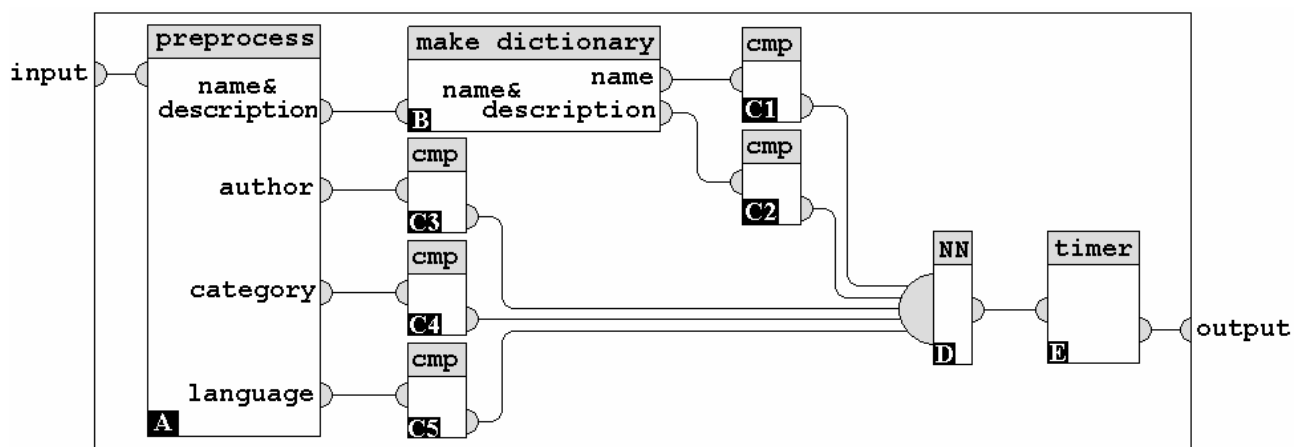


Рис. 2. Граф алгоритма.

Вторая задача «пост-троечные последовательности» (pooled sequences)

Во второй задаче соревнования была предоставлена статистика поведения пользователей, однако, не полная, как это принято в задачах такого типа (кто, что и когда смотрел). Причиной такой неполноты является одна из тенденций в развитии современной IT-индустрии: обеспечение анонимности действий пользователей в Интернете. В последнее время актуальнейшей задачей становится проблема правильного «огрубления» информации, при котором задачи анализа данных можно решать с приемлемым качеством, но нельзя восстановить исходные данные (подробные логи пользователей). Организаторы соревнования для такого огрубления применили метод пост-троечных последовательностей (это наш перевод термина «pooled sequences»).

Обучающая выборка T состоит из троек $\{a, b, c\}$ номеров лекций (всего 109044 троек). Для каждой тройки указано $n(\{a, b, c\})$ – число пользователей, которые посмотрели все три лекции. Кроме того, указана пост-троечная последовательность (polled sequence). Это список лекций, которые посмотрел хотя бы один просмотревший $\{a, b, c\}$ после всех лекций тройки, с указанием, сколько таких пользователей посмотрели эти лекции. Опишем кратко процедуру построения таких последовательностей [s2]. Пусть лог какого-то пользователя (последовательность лекций, которые он посмотрел)

$102 \rightarrow 33 \rightarrow 2 \rightarrow 34 \rightarrow 35 \rightarrow 2 \rightarrow 102 \rightarrow 17 \rightarrow 36,$

удаляем из неё повторы:

$102 \rightarrow 33 \rightarrow 2 \rightarrow 34 \rightarrow 35 \rightarrow 17 \rightarrow 36.$

Для тройки $\{2, 33, 35\}$ данный пользователь считается тем, кто посмотрел все три лекции тройки, а лекции с номерами из $\{17, 36\}$ будут входить в посттроечную последовательность.

Мы формализуем понятие «пост-троечная последовательность» с помощью вектора $v(\{a, b, c\}) \in \mathbf{Z}^L$, где L – число лекций,

$$v(\{a, b, c\}) = (v_1(\{a, b, c\}), \dots, v_L(\{a, b, c\})),$$

$v_j(\{a, b, c\})$ – сколько раз была просмотрена j -я лекция после тройки $\{a, b, c\}$ (неформально говоря, это популярность j -й лекции после просмотра лекций из $\{a, b, c\}$, наш пример «добавляет единицы» к 17-му и 36-му элементам вектора).

Контрольная выборка состоит из 60274 троек (не входящих в обучение). Требуется для троек контрольной выборки определить пост-троечные последовательности, а точнее 10 первых членов последовательностей (10 координат вектора $v(\{a, b, c\})$ наибольшими значениями). Это соответствует рекомендации пользователю списка из 10 лекций после просмотра им тройки лекций. Отметим, что также известна контентная информация (как и в первой задаче), но она не была использована участниками, поскольку не улучшала качества алгоритмов. Функционал качества для оценки алгоритмов во второй задаче использовался такой же, как и в первой, но при $Z = \{5, 10\}$.

Алгоритм решения второй задачи

Сначала производятся две нормировки векторов обучающей выборки, первая –

$$v'(\{a, b, c\}) = \left(\frac{v_1(\{a, b, c\})}{\log(|\{\tilde{t} \in T \mid v_1(\tilde{t}) > 0\}| + 2)} \quad \dots \quad \frac{v_L(\{a, b, c\})}{\log(|\{\tilde{t} \in T \mid v_L(\tilde{t}) > 0\}| + 2)} \right),$$

ясно что $|\{\tilde{t} \in T \mid v_j(\tilde{t}) > 0\}|$ – число троек из обучения, в пост-троечные последовательности которых входит j -я лекция. Вторая нормировка –

$$v''(\{a, b, c\}) = \left(\frac{v'_1(\{a, b, c\}) \cdot \log\left(\sum_{j=1}^L v'_j(\{a, b, c\}) + 1\right)}{\sqrt{3 \cdot n(\{a, b, c\}) + \varepsilon}} \dots \right. \\ \left. \dots \frac{v'_L(\{a, b, c\}) \cdot \log\left(\sum_{j=1}^L v'_j(\{a, b, c\}) + 1\right)}{\sqrt{3 \cdot n(\{a, b, c\}) + \varepsilon}} \right)$$

Замечание. Первая нормировка учитывает, что если лекция часто входит в пост-троечные последовательности, то факт такого вхождения не является информативным. Были опробованы варианты деления на $|\{\tilde{t} \in T \mid v_j(\tilde{t}) > 0\}|$, квадрат и корень этого числа. Вторая нормировка также получилась в результате перебора разных вариантов и увеличивает качество на 1%.

Пусть

$$s(\tilde{d}) = \sum_{\tilde{t} \in T: \tilde{d} \subseteq \tilde{t}} v''(\tilde{t}),$$

а $n(\tilde{d}) = |\{\tilde{t} \in T: \tilde{d} \subseteq \tilde{t}\}|$ – число слагаемых в этой сумме. Например, $s(\{a, b\})$ – сумма векторов $v''(\{a, b, d\})$ для всех d таких, что $\{a, b, d\} \in T$. Пусть операция ω удаляет из вектора все нулевые координаты, кроме одной, а если нулевых координат не было, то добавляет одну нулевую. Например, $\omega(1, 0, 0, 2, 0) = (1, 0, 2)$, $\omega(1, 2) = (1, 2, 0)$. Пусть

$$\text{std}(x_1, \dots, x_n) = \sqrt{\frac{1}{n-1} \sum_{t=1}^n \left(x_t - \frac{1}{n} \sum_{i=1}^n x_i \right)^2}$$

(стандартное отклонение [s10]).

Алгоритм действует очень просто: если среди чисел $n(\{a, b\})$, $n(\{a, c\})$, $n(\{b, c\})$ есть как минимум два ненулевых числа (это соответствует достаточному количеству статистики), то полагаем

$$\gamma = \frac{\log(s(\{a,b\}) + 0.02)}{\text{std}(\omega(s(\{a,b\}))) + 0.5} + \frac{\log(s(\{b,c\}) + 0.02)}{\text{std}(\omega(s(\{b,c\}))) + 0.5} + \frac{\log(s(\{a,c\}) + 0.02)}{\text{std}(\omega(s(\{a,c\}))) + 0.5}.$$

В противном случае добавляем к этой сумме слагаемые

$$\frac{\log(s(\{a\}) + 0.02)}{\text{std}(\omega(s(\{a\}))) + 0.5} + \frac{\log(s(\{b\}) + 0.02)}{\text{std}(\omega(s(\{b\}))) + 0.5} + \frac{\log(s(\{c\}) + 0.02)}{\text{std}(\omega(s(\{c\}))) + 0.5}.$$

Здесь логарифм $\log$ берётся поэлементно от вектора, константы получились в результате оптимизации качества алгоритма. Элементы полученного вектора γ трактуем как оценки популярности лекции в пост-троечной последовательности (чем выше оценка – тем более популярна). Рекомендуем лекции с наибольшими оценками.

Попытаемся обосновать алгоритм. Представляется очень логичным действовать просто по правилу

$$\begin{aligned} \gamma &= \log(s(\{a,b\})) + \log(s(\{b,c\})) + \log(s(\{a,c\})) = \\ &= \log(s(\{a,b\}) \cdot s(\{b,c\}) \cdot s(\{a,c\})), \end{aligned}$$

где « $\cdot$ » – поэлементное умножение векторов. Действительно, если нет информации о тройке $\{a,b,c\}$, то мы смотрим информацию о тройках $\{a,b,d\}$ для всех d , при которых они входят в обучающую выборку. При этом суммируем все $v(\{a,b,d\})$, что соответствует объединению мультимножеств [s11] – получаем вектор $s(\{a,b\})$. Аналогично поступаем для троек вида $\{a,c,d\}$ и $\{b,c,d\}$ – получаем векторы $s(\{a,c\})$, $s(\{b,c\})$. Теперь логично пересечь полученные мультимножества. В теории мультимножеств это обычно выполняется операцией минимум, но экспериментально было установлено, что вместо минимума лучше использовать произведение (как принято в теории нечётких множеств [s12]):

$$s(\{a,b\}) \cdot s(\{b,c\}) \cdot s(\{a,c\}).$$

Переход к выражению

$$(s(\{a,b\}) + \varepsilon) \cdot (s(\{b,c\}) + \varepsilon) \cdot (s(\{a,c\}) + \varepsilon)$$

связан с тем, что здесь не происходит зануления большинства элементов вектора (и потери информации), см. также табл. 3. Затем были произведены эксперименты по нормировкам векторов, а также по учёту $s(\{;\cdot\}) + \varepsilon$ с разными весами. Последние привели к делению на $\text{std}(\omega(s(\{;\cdot\}))) + 0.5$ (это увеличивает качество примерно на 1%).

Замечание. Участники соревнования, занявшие второе место [s13], использовали правило «энтропийного» типа:

$$\gamma = s(\{a,b\}) \cdot \log(s(\{a,b\})) + s(\{b,c\}) \cdot \log(s(\{b,c\})) + s(\{a,c\}) \cdot \log(s(\{a,c\})),$$

и простейшую нормировку:

$$v''(\{a,b,c\}) = \left(\frac{v_1(\{a,b,c\})}{n(\{a,b,c\})} \dots \frac{v_L(\{a,b,c\})}{n(\{a,b,c\})} \right)$$

(для интерпретации получаемых величин как вероятностей). Возможно, применение энтропийного правила в описанном методе увеличивает качество.

γ (вид выражения)	качество
$(s(\{a,b\}) + \varepsilon) \cdot (s(\{b,c\}) + \varepsilon) \cdot (s(\{a,c\}) + \varepsilon), \varepsilon = 0$	57.27%
$=*=, \varepsilon = 0.01$	62.11%
$=*=, \varepsilon = 0.1$	61.60%
$=*=, \varepsilon = 1$	58.84%
$(s(\{a,b\}) + s(\{b,c\}) + \varepsilon) \cdot (s(\{b,c\}) + s(\{a,c\}) + \varepsilon) \cdot (s(\{a,c\}) + s(\{a,b\}) + \varepsilon), \varepsilon = 0$	58.63%
$=*=, \varepsilon = 0.001$	59.87%

Табл. 3. Качество алгоритма при различных γ .

Замечание. При решении этой задачи также была использована идеология «LENKOR»: поиск подходящей линейной комбинации (именно поэтому произведение было записано как сумма логарифмов) и её последующая «деформация» (учитывая специфику задачи, больше внимания уделено нормировкам данных). Здесь не пришлось выбирать типы информации и оценку близости по ним, поскольку использование контентной информации не увеличивало качество (получались нулевые коэффициенты в линейной комбинации), зато был произведён обширный перебор различных нелинейных выражений (в турнирной таблице до последнего дня два лучших участника показывали близкие результаты, поэтому приходилось бороться за каждый процент качества).

Описанный алгоритм занял первое место среди 22 участников с результатом 62.415% (у автора есть уверенность, что показанный результат близок к оптимальному для данной задачи). Алгоритм настолько простой, что MATLAB-код занимает несколько строчек. Время работы алгоритма на компьютере HP p6050ru Intel Core 2 Quad CPU Q8200 2.33GHz, RAM 3Gb, OS Windows Vista в среде MATLAB 7.10.0 для прогноза по одной лекции составило 0.0383 секунды, для вычисления ответа второй задачи соревнования (60274 прогнозов) – 38.33 минуты.

Алгоритмы	Предфинальный результат	Итоговый результат
Наш алгоритм	62.102%	62.415%
Второе место (J. Kreiner [s13])	60.791%	61.172%
Третье место (V. Nikulin [s14])	58.727%	59.063%

Табл. 4. Результаты алгоритмов-победителей второго подконкурса.

Выводы

Описанные алгоритмы достаточно просты, универсальны, могут использоваться для других постановок задач, допускают возможности распараллеливания. Решение получается в удобном виде: как вектор оценок.

Для рекомендации некоторого количества лекций достаточно отобрать столько наибольших элементов вектора, но, в принципе, параллельно получаются оценки популярности каждой лекции. Кроме того, алгоритмы такого типа могут быть использованы в рамках алгебраического подхода [s15] для формирования выражений над алгоритмами. По сути, в основе технологии «LENKOR» лежат идеи алгебраического подхода: выбирается «правильная» база пространства векторов оценок, а затем настраивается алгебраическое выражение.

Литература и ссылки

- [a1] Takacs G., Pilaszy I., Nemeth B. and Tikk D. (2009) Scalable Collaborative Filtering Approaches for Large Recommender Systems. *Journal of Machine Learning Research*, 10, 623–656.
- [a2] Michael J. Pazzani, Daniel Billsus: Content-Based Recommendation Systems. *The Adaptive Web 2007*: 325–341.
- [a3] Jannach, D., Zanker, M., Felfernig, A., Friedrich, G.: *Recommender Systems: An Introduction*. Cambridge University Press (2010)
- [s1] Репозиторий научных и учебных лекций, <http://www.videolectures.net/>
- [s2] N. Antulov-Fantulin, M. Bošnjak, T. Šmuc, M. Jermol, M. Žnidaršič, M. Grčar, P. Keše, N. Lavrač, ECML/PKDD 2011 - Discovery challenge: «VideoLectures.Net Recommender System Challenge», <http://tunedit.org/challenge/VLNetChallenge/>
- [s3] Европейская конференция по машинному обучению, www.ecmlpkdd2011.org
- [s4] Платформа для проведения соревнований, <http://tunedit.org>
- [s5] Маннинг К., Рагхаван П., Шютце Х. Введение в информационный поиск — Вильямс, 2011.
- [s6] Porter, 1980, An algorithm for suffix stripping, *Program*, Vol. 14, № 3, pp. 130–137.
- [s7] Код стеммера Портера, <http://tartarus.org/~martin/PorterStemmer/matlab.txt>
- [s8] E. Spyromitros-Xioufis, E. Stachtiari, G. Tsoumakas, and I. Vlahavas: A Hybrid Approach for Cold-start Recommendations of Videolectures // *Proc. of ECML-PKDD 2011 Discovery Challenge Workshop*, pp 29-39, (2011).
- [s9] M. Možina, A. Sadikov, and I. Bratko: Recommending VideoLectures with Linear Regression // *Proc. of ECML-PKDD 2011 Discovery Challenge Workshop*, pp 41-49, (2011).
- [s10] http://en.wikipedia.org/wiki/Standard_deviation
- [s11] Blizard, Wayne D. (1989) "Multiset theory," *Notre Dame Journal of Formal Logic*, Volume 30, Number 1, Winter: pp. 36–66.
- [s12] http://en.wikipedia.org/wiki/Fuzzy_mathematics
- [s13] J.A. Kreiner, E. Abraham Recommender System Based on Purely Probabilistic Model from Pooled Sequence Statistics // *Proc. of ECML-PKDD 2011 Discovery Challenge Workshop*, pp 51-57 (2011)
- [s14] V. Nikulin: OpenStudy: Recommendations of the Following Ten Lectures After Viewing a Set of Three Given Lectures // *Proc. of ECML-PKDD 2011 Discovery Challenge Workshop*, pp 59-69, (2011).

[s15] Журавлёв Ю.И. Об алгебраических методах в задачах распознавания и классификации // Распознавание, классификация, прогноз. – 1988. – Т.1. – С.9–16.