А.М. Прокофьев

Метод оптимизации классификаторов на основе наблюдения за их использованием

Аннотация. В статье описывается способ оптимизации иерархических классификаторов в системах управления контентом. По сравнению с существующими подходами, он имеет более широкую область применения и может использоваться с первого дня работы системы, без необходимости предварительного обучения или настройки. Приводятся концептуальное и формальное описание метода, а также результаты его применения.

Ключевые слова: оптимизация, оценка, классификаторы, классификация, онтологии.

Введение

Современные системы управления контентом широко используют иерархическую классификацию хранимой в них информации. Наиболее распространенным примером классификатора является дерево папок, в которых находятся документы в файловой системе. Более сложные способы организации информации предлагают корпоративные базы знаний, в которых каждая статья соотносится сразу с несколькими категориями иерархического классификатора, описывающего область деятельность компании.

В этой статье под классификатором мы будем понимать систематизированный перечень объектов, каждому из которых присвоен определенный шифр, а под классификацией – процесс соотнесения объектов категориям классификатора.

Построение оптимального классификатора является нетривиальной задачей, которая решается несколькими способами. На начальном этапе классификаторы могут быть скопированы из предыдущих проектов в новую систему. Этот подход требует наличия предыдущего опыта, либо привлечения экспертов. При этом изучение большого классификатора потребует значительных усилий от конечных пользователей и затруднит внедрение и запуск в эксплуатацию системы управления знаниями.

Альтернативным подходом является предоставление пользователям системы возможности самостоятельно формировать классификаторы. В таком случае, классификация будет формироваться и изучаться постепенно, что упрощает начальный этап использования системы [6].

Однако группа пользователей с первого раза, скорее всего, не сможет сформировать оптимальный классификатор и потребуется его значительная переработка. Даже опытный сотрудник с аналитическими навыками может построить классификатор, который будет неудобен для других сотрудников, потому что они представляют предметную область иным образом.

В статье рассмотрены способы решения следующих задач: построение оптимального классификатора, выполнение оптимальной классификации и оценки результатов классификации. После чего предложим новый метод, решающий эти задачи совместно.

1. Обзор существующих подходов

Задача построения и развития оптимального классификатора решается как с помощью классических методов информатики, так и онтологического инжиниринга [4]. Рассмотрим основные подходы к формированию классификаторов. В первую очередь необходимо определиться с критериями оптимальности, среди которых:

1. Оценки *согласованности, полноты и корректности* [10] предлагают набор правил, которым должен соответствовать классификатор.

2. *Качество и скорость* выполнения набора задач с использованием онтологии [17] является одной из наиболее точных оценок оптимальности классификатора, так как измеряет показатели его фактического использования на реальных задачах конечными пользователями. Однако результаты этого метода сильно зависят от набора задач, который формулируется аналитиком.

3. *Оценки эргономичности* [1] предлагают набор формально-проверяемых критериев, которые характеризуют оптимальные классификаторы.

Разработано большое количество способов формирования и оптимизации классификаторов, которые применяются на различных этапах разработки и использования информационной системы. Мы рассмотрим их в трех группах: первоначальное формирование классификатора; выполнение классификации элементов; изменение и доработка классификатора.

Существует несколько способов для первоначального формирования классификатора:

1. Золотые стандарты [25], содержащие шаблонные классификаторы для различных предметных областей, которые можно взять за основу.

2. Автоматическая кластеризация [12] известна уже боле 50 лет и предлагает набор алгоритмов, группирующих элементы по определенным критериям. Сформированные поименованные группы становятся элементами классификатора.

3. Формирование на основе семантического анализа конвента [7], в том числе с помощью мульти-агентных систем [20].

4. Копирование классификатора из аналогичных систем и проектов.

5. Неформализованные способы на основе анализа предметной области.

После формирования основных классификаторов и на всех дальнейших этапах использования информационной системы выполняется задача классификации хранимых в системе элементов. Она может быть решена одним из трех способов.

1. Ручной. Сотрудники самостоятельно классифицируют элементы, опираясь на свои знания и опыт.

2. Полуавтоматический. Программа предлагает набор категорий для элемента, а пользователь его утверждает.

3. Автоматическая классификация:

- на основе правил и метаданных [13], при которой администратор системы заранее настраивает правила соответствия контента категориям классификатора;

- на основе содержимого, с обучением [11, 19], при которой пользователи первоначально классифицируют набор элементов, а система затем выполняет размещение новых элементов по аналогии. Распространенным примером такого способа классификации являются спам-фильтры в почтовых клиентах.

Существующие классификаторы должны развиваться и изменяться, чтобы включать в себя новые понятия предметной области, описывать ее более детально, или предлагать более совершенное, с точки зрения практических задач, представление этой области.

Рассмотрим следующие способы эволюции и развития классификаторов.

1. Ручной. Целевую структуру определяет аналитик, и он же выполняет преобразование;

- с привлечением экспертов для выполнения изменений;

- с использованием методических указаний [21], предоставляющих инструкции по доработке классификатора.

2. Полуавтоматическая переработка:

- структура предлагается аналитиком, переход к новой структуре выполняется с помощью программного инструмента [24];

- изменения в структуре предлагаются программой на основе критериев оптимальности, которые указаны выше, но выполняются аналитиком [1].

3. Автоматическая переработка классификаторов:

- на основе явной обратной связи от пользователей [23];

- на основе переноса изменений из связанных систем [8].

Задача построения оптимального классификатора является одной из частных задач онтологического инжиниринга [2]. Обзор состояния исследований в этой области в целом приведен в [15], а способов эволюции онтологий - в [9].

В статье описан метод оптимизации и оценки иерархических классификаторов и результатов классификации. Для краткости мы будем

называть его методом оптимизации классификаторов. Научная новизна метода состоит в следующем:

1. Новый принцип оптимизации – формирование и исполнение рекомендаций на основе пассивного автоматизированного наблюдения за использованием классификатора.

2. Новый объект классификации – интерпретация контента пользователями, а не сам контент (раздел 4 –сравнение с существующими подходами).

3. Применимость в областях, в которых другие методы неэффективны:

- непрерывно изменяющиеся классификаторы;
- классификаторы с малым количеством элементов в каждой категории (до 10);
- классификация нетекстового контента (изображения, звуковые и видео файлы, схемы).

2. Метод оптимизации классификаторов

Работа метода основана на предположении, что пользователь во время поиска в классификаторе косвенным образом выполняет оптимальную, с его точки зрения, классификацию искомого элемента. Рассмотрим это на примере.

Категории классификатора изображены на Рис.1 прямоугольниками, кружки с цифрами отображают последовательность перехода пользователя по классификатору. Искомым элементом является X. Пользователь переходит из категории A в B, а затем обратно в A. На основании этого можно сделать вывод, что он искал элемент X в категории B и сформировать рекомендацию о перемещении элемента X в B, либо переименовании пары категорий (B,C).

2.1. Концептуальное описание метода

Работа метода начинается с формирования общего потока событий, описывающих действия пользователя в информационной системе, и последовательном анализе каждого атомарного события для выявления зависимостей и шаблонов, формирующих комплексные события. Общие методы выявления и обработки комплексных событий описаны в [18].

В рамках метода рассматриваются следующие атомарные события.

1. Просмотр элемента:

- простой - открытие карточки, скачивание файла;

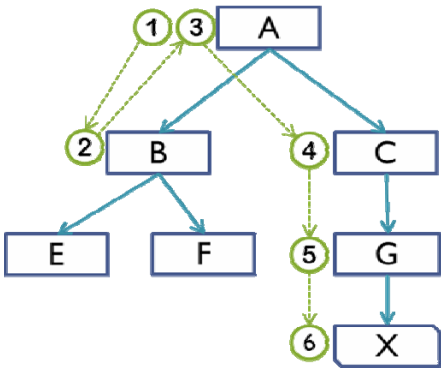

Рис. 1. Пример поиска в классификаторе

- с учетом действий и времени - просмотр карточки/файла в течении N секунд; пролистывание до конца содержимого;
- с обратной связью - пользователь поставил оценку элементу или внес в него изменения.

2. Создание/изменение/удаление/ нового элемента.

3. Классификация элемента:

- в древовидных хранилищах – размещение элемента в контейнер;
- в хранилищах на основе графа – создание связи между элементами.

4. Навигация – переход между элементами хранилища.

Первоочередной задачей является определение успешности поиска в иерархически структурированном хранилище. Для ее анализа была разработана модель поиска (Рис. 2).

На рисунке пунктиром показаны состояния, которые являются подразумеваемыми и не являются прямым результатом действий пользователя. На основе этой модели формируются следующие комплексные события:

Ошибочный шаг при поиске. Обнаружение (действия пользователя):

- находится на уровне A;
- перешел на уровень глубже (B);
- не просматривал никаких элементов;
- перешел на уровень выше(A) или в другую ветку (C).

Вывод: переход на уровень B был ошибочен. Для данного пользователя классификация на этом уровне оказалась неправильной.

Завершение поиска:

- успешное - просмотрел элемент с учетом действий и времени или обратной связи;

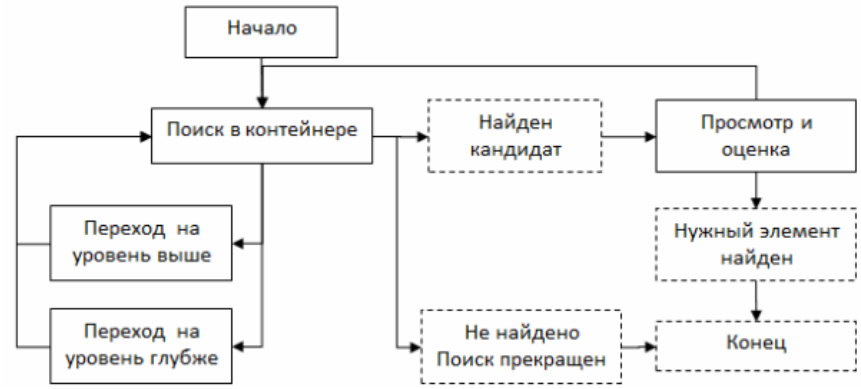

Рис. 2. Модель поиска

- неуспешное, одно из двух: пользователь в течение 5 минут не выполняет поисковых действий; пользователь завершил работу с модулем или системой.

Успешный поиск с первого прохода. Обнаружение (действия пользователя):

- выполняет один или несколько переходов на уровень глубже, при этом не переходит на вышестоящие и соседние уровни;
- успешно завершает поиск.

Успешный поиск:

- пользователь выполняет один или несколько переходов, при этом он может просматривать некоторые элементы;
- успешно завершает поиск.

Успешный поиск, но в процессе были ошибочные шаги:

- пользователь выполняет один или несколько переходов; некоторые шаги являются ошибочными;
- успешно завершает поиск.

Полученные комплексные события подвергаются дальнейшему статистическому анализу. При фрагментарном подходе рассматриваются отдельные ошибочные шаги при поиске и выявляются наиболее часто встречающиеся, а при интегральном – выявляются совпадающие пути и их части, которые включают в себя ошибочные шаги. В результате анализа выявляются конкретные проблемы классификации в иерархическом хранилище:

- классификация на уровне, на котором произошел ошибочный шаг, была некорректной и требует доработки;
- для отдельного элемента можно сказать, что его поиск выполнялся в определенном наборе контейнеров, однако найден он был в другом.

По результатам анализа формируется отчет по реструктуризации информационного хранилища, который включает в себя рекомендации по доработке отдельного контейнера, в котором происходят ошибочные шаги или переработки области хранилища в целом, если проблемы наблюдаются в нескольких ее контейнерах.

Для компенсации выявленных проблем в контейнерах, в которых часто происходит ошибочный поисковый шаг, добавляются подсказки, указывающие на корректное расположение искомого элемента. При наличии большого количества рекомендаций по реструктуризации возможно автоматическое построение параллельных классификаторов, в которых эти рекомендации будут применены.

2.2. Формальное описание метода

Перейдем к формальному описанию предложенного метода для общего случая. Рассмотрим направленный связный ациклический граф G:

G = (N, L), где N – множество узлов, а L – множество связей между узлами;

$N = C \cup D$, где C – множество категорий, D – множество информационных элементов (например, документов или форм).

Метод оптимизации классификаторов M_{dyn} состоит из следующих шагов.

Шаг 1. Обнаружение событий в информационной системе, формирование из них множества E_{all}. Существует несколько путей обнаружения событий навигации в информационной системе:

- использование специализированных API ИС;
- обработка журналов аудита ИС;
- периодическая проверка текущего состояния ИС;

- создание приложения-прослойки между клиентской и серверной частью ИС.

Более подробно эти способы будут рассмотрены в разделе 4.

Шаг 2. Фильтрация и группировка событий по сессиям в потоки $\boldsymbol{E_{all} \xrightarrow{filter} \{E_s\}}$. Рассматриваемые информационные системы могут генерировать тысячи и десятки тысяч событий в секунду, но для дальнейшего анализа нужна лишь определенная их часть. На этом шаге отфильтровываются лишние события, а также выполняется их группировка по пользователям, сессиям пользователя и упорядочивание по времени.

Шаг 3. Выделение поисковых путей из потока $\boldsymbol{E_s \xrightarrow{split} \{Sp\}}$. Навигационные события, собранные для пользовательской сессии, содержат информацию о работе пользователя с системой в течение длительного времени и могут включать в себя множество поисковых путей, а также случайные действия. Задача этого шага – выделение конкретных поисковых путей из E_s.

Одним из подходов к решению этой задачи является группировка событий по временной шкале, чтобы разница во времени возникновения последовательных событий в группе t_{gr_int} была значительно меньше временного интервала между группами t_{gr_ext}:

$$E_s \rightarrow \{Sp_i = \{e_{i,j} | e_{i,j+1}.t - e_{i,j}.t < t_{gr\_int}\} | e_{i+1,1}.t - e_{i,\mathrm{len}(Sp_i)}.t > t_{gr\_ext}\}$$

Количество возвращаемых групп будет зависеть от выбора параметров t_{gr}. Таким образом, разбиение можно выполнить несколькими способами. Эти параметры зависят от множества факторов – количества элементов, из которых нужно сделать выбор на следующем шаге поискового пути и их схожести, индивидуальных особенностей пользователя, меры его ознакомления с классификатором и других.

Для исключения из потока обособленных событий и тривиальных групп, на основе которых не удастся сформировать рекомендации в дальнейшем, зададим минимальный размер группы:

$$split_{time}(E_s) := \{Sp_i \mid \mathrm{len}(Sp_i) > K_{gr\_msize}\}$$

Шаг 4. Определение проблемных шагов в поисковом пути $\boldsymbol{Sp \xrightarrow{detect} \{Bp\}}$. Оптимальным поисковым путем будем считать последовательность переходов по графу G, не содержащую в себе повторных переходов в один и тот же узел, и приводящих к искомому узлу.

$$Sp_{opt} = \{N_{start}, N_i \ldots N_j, N_{target} | N_i \neq N_j, \forall i \neq j\}$$

Для ациклического графа и заданного искомого узла таких путей может быть несколько. В рамках данного метода мы будем считать их равно оптимальными, так как, несмотря на различную их длину, они также могут иметь и различное семантическое значение для пользователя. В результате, пользователь может пройти длинный, но понятный ему путь, быстрее, чем короткий.

В рамках данного метода рассматриваются следующие ситуации, которые считаются проблемными с точки зрения скорости нахождения искомого элемента в иерархическом классификаторе:

- отличие поискового пути от множества оптимальных путей;
- превышение времени перехода между узлами поискового пути порогового значения $t_{choice\_max.}$

Для определения первой ситуации, вычисляется разность между рассматриваемым поисковым путем и множеством оптимальных путей $\Delta Sp_i = Sp \setminus Sp_{opt_i}$. Разность с наименьшей мощностью обозначим ΔSp_{min}. Множество проблемных шагов определяется как:

$$\{Bp\} = \Delta Sp_{min} \cup \{N_i | N_{i+1}.t - N_i.t > t_{choice\_max}\}$$

Шаг 5. Формирование локальной рекомендации $\boldsymbol{\{Sp, \{Bp\}\} \xrightarrow{recommend} \{Rl\}}$. На основе обнаруженных проблем формируются рекомендации по их устранению:

$$R = \{C, A, E, W, T\},$$

где C- контекст, описывающий проблемную область, A – рекомендуемое к выполнению действие, E – объяснение необходимости действия, W – вес (степень уверенности в рекомендации), T - тип рекомендации.

Вид параметров рекомендации зависит от ее типа. При обнаружении поисковых шагов, отличающихся от оптимального пути, формируется следующий набор локальных рекомендаций:

- переместить искомый элемент во все контейнеры, в которых он искался, но не был найден;

- для каждого ошибочного шага, переименовать пару контейнеров $\{N_{rigth}, N_{wrong}\}$, где N_{rigth} является следующим шагом оптимального пути, а N_{wrong} - поискового.

При превышении времени перехода между узлами поискового пути порогового значения t_{choice_max} формируется рекомендация на переименование контейнеров на этом уровне. Вес локальной рекомендации равняется точности обнаружения решаемой проблемы.

Шаг 6. Слияние локальной и глобальных рекомендаций $\{\boldsymbol{R_{all}}, \boldsymbol{Rl}\} \overset{merge}{\rightarrow} \boldsymbol{R_{all}}$. После формирования локальной рекомендации, она объединяется с множеством всех рекомендаций. Отдельная рекомендация имеет малый вес, но если она возникает многократно, ее вес возрастает при операции слияния:

$$R_{all_i}.W := R_{all_i}.W + Rl.W \ , i \,|\, R_{all_i}.A = Rl.A$$

При первом возникновении локальная рекомендация добавляется к множеству глобальных:

$$R_{all} := R_{all} \cup Rl$$

Шаг 7. Выполнение первоочередных рекомендаций в ручном или автоматическом режиме. Рекомендации определенных типов (перемещение, добавление связи) содержат конкретные указания по изменению классификатора, которые могут быть выполнены в автоматическом режиме. Другие рекомендации (переименование, реструктуризация области) только указывают на проблему, но способ ее решения предлагается найти пользователям. Во всех случаях, рекомендация отображается пользователю только после того как она достигнет заданного веса W_{min}, т.е. после многократного повторения проблемы:

$$R_{top} := \{R \,|\, R \in R_{all}, R.W > W_{min}\}$$

Шаг 8. Переоценка рекомендаций. После выполнения конкретной рекомендации она исключается из множества R_{all}. Для остальных рекомендаций выполняется переоценка их веса. Чтобы понять, привело ли изменение классификатора к его улучшению с точки зрения данной рекомендации, рассмотрим динамику изменения ее веса.

Обозначим среднюю скорость увеличения веса рекомендации на временном интервале $[t_1, t_2]$ как $V_{[t_1,t_2]}(W)$ и рассмотрим ее значение до и после выполнения рекомендации:

$$\Delta V = V_{[t_r, t_r + t_w]}(W) - V_{[t_r - t_w, t_r]}(W),$$

где t_r – время выполнения рекомендации, t_w – размер временного интервала.

Отрицательные значения ΔV означают, что проблема, описываемая в рекомендации, стала менее значимой. Положительные значения указывают на то, что проблема только возросла после выполнения изменений классификатора. После выполнения оценки скорости, выполняется переоценка веса рекомендации:

$$R.W := R.W + \Delta V \cdot t_r$$

3. Сбор показателей и анализ метода

Для оценки данной технологии были собраны экспериментальные данные из системы управления знаниями, основные компоненты которой были разработаны в рамках предыдущих исследований. В ее рамках возможно выполнение всего спектра необходимых тестов, потому что имеется возможность глубокой интеграции и обнаружения любых событий, которые наблюдаются методом.

Рассматриваемая система содержит 46061 элемент – иерархически структурированные документы проектов, презентации и другие обучающие материалы. Она используется в консалтинговом агентстве для систематизации и интеграции внутренних знаний компании.

Для тестирования нагрузки и скорости обработки при большом потоке событий, использовались синтетические тесты, которые были построены на основе параметров, выделенных в реальных тестах. Они будут более подробно описаны в следующих разделах.

3.1. Методика тестирования

Для оценки метода оптимизации классификаторов применялась совокупность показателей, характеризующая эффективность и практическую применимость метода.

Для оценки эффективности измерялись следующие показатели:

1. Точность выделения комплексных событий из потока атомарных событий хранилища. Этот показатель характеризует способность технологии к интерпретации поступающих данных из существующих систем. Если точность будет низкой, то все дальнейшие показатели работоспособности не будут валидными.

2. Процент ошибок в навигации. Рассчитывается, как отношение путей, содержащих ошибочные шаги, к общему проценту путей. Показывает целесообразность применимости технологии и дает конкретную оценку объема решаемых проблем.

Валидизация измеренных значений выполняется пользователем. Например, для первого показателя, точности выделения комплексных событий, для случайной части событий пользователю будет задан вопрос – правильно ли оно было выделено. Задав такой вопрос многократно, можно сформировать статистическую оценку данного показателя.

Для оценки практической применимости был измерен ряд показателей, который характеризует, насколько данный метод масштабируется для реальных задач:

- время на определение комплексного действия в зависимости от количества одновременных пользователей;
- производительность механизма анализа в зависимости от объема хранилища, количества пользователей, объема накапливаемой истории;
- степень параллельности обработки и распределения нагрузки на все доступные вычислительные ресурсы.

Для оценки этих показателей будут использоваться синтетические тесты, симулирующие одновременную работу большого количества пользователей, также возможно использование записанных навигационных сессий реальных пользователей.

3.2. Оценка результатов применения

Для оценки эффекта от применения технологии был проведен эксперимент на основе разработанной системы управления знаниями. Модуль сбора событий выполнял наблюдение за навигацией, выполняемой пятью сотрудниками консалтинговой фирмы в течение пяти дней. После этого, полученные данные были переданы для системы анализа атомарных событий, которая сформировала рекомендации по улучшению структуры хранения.

За счет использования собственной системы управления знаниями удалось избежать проблемы с интерпретацией событий, формируемых информационной системой.

Было собрано 3673 атомарных события, в результате анализа которых были получены следующие данные:

- количество пройденных путей при навигации – 348;
- общее время навигации - 01:49:57 = 6597 с.

Достоверно определить цель поиска в рамках текущей методики удалось только для определенной части путей, которая в дальнейшем и анализировалась. Остальные пути либо соответствовали обзорным переходам по хранилищу без цели найти определенный элемент, либо были целенаправленными, но элемент так и не был найден, из-за его неправильной классификации или отсутствия в хранилище системы:

- количество завершенных навигационных путей - 176 (51%);
- общее время завершенных навигационных путей - 01:03:11 = 3791 с.

После автоматического анализа путей навигации, было выявлено:

- количество путей, содержащих проблемные шаги - 66 (38%);
- общее время путей, содержащих проблемные шаги - 00:34:03 = 2043 с.

На основе выявленных проблем, были сформированы рекомендации:

- количество рекомендаций - 21;
- сокращение времени навигации после применения рекомендаций – 465 с;
- рекомендация с самым большим весом сокращает время навигации на 64 с, с самым малым – на 3 с.

Таким образом, в данном случае разработанная технология обеспечивает сокращение времени навигации на 12% .

Следует отметить, что рассматриваемое хранилище целенаправленно структурировалось пользователем, поэтому в хранилищах с менее качественной структуризацией результат от выполнения рекомендаций будет более значимым.

3.3. Тестирование производительности

Для нагрузочного тестирования был создан механизм, симулирующий действия пользователей по навигации в иерархическом хранилище. Для него были выбраны параметры, на основе реального тестирования в предыдущем разделе. В результате было сгенерировано 356 631 событий, на основе которых было выделено:

- количество пройденных путей при навигации - 37 598;
- общее время навигации - 8 дней 14:56:27 = 744 987 с.

Конфигурация оборудования, на котором производилось тестирование: четырехядерный процессор Intel Q6600, 8 Гб памяти, 64-битная ОС.

Для обработки и анализа событий потребовалось 669 с, таким образом, скорость обработки составила 533 событий/с. Обработка выполнялась в рамках одного потока, с задействованием только одного вычислительного ядра. Время анализа событий составило 2 м/с, что соответствует требованию к обработке в реальном времени.

При многопоточной обработке с использованием всех вычислительных ядер средняя производительность составила 1886 событий/с, что соответствует масштабируемости с коэффициентом 1,8.

Следует отметить, что скорость обработки в значительной степени зависит от характера событий, их последовательности и частоты возникновения. Для определения ряда комплексных событий требуется в течение значительного времени (до 10 мин) дожидаться возникновения другого события, или факта отсутствия новых событий. При этом все новые события проверяются на возможность дополнения комплексных событий уже находящихся в ожидании.

В целом, производительность предложенного метода достаточна для обеспечения наблюдения и анализа в реальном времени за действиями 900 пользователей на одном выделенном сервере с многоядерным процессором (при расчете два события от каждого пользователя в секунду).

4. Сравнение с существующими подходами и области применения

Большинство перечисленных подходов предназначено для оптимального построения или дополнения классификатора. Предлагаемый метод предназначен для устранения ошибок в уже существующем классификаторе, поэтому он может применяться совместно с другими методами и повышать их точность.

Наиболее близким аналогом является метод автоматической оптимизации классификаторов на основе явной обратной связи [23], однако он еще находится в стадии разработки и экспериментальные данные по нему не были получены. Поэтому мы проведем сравнение с одной из наиболее распространенных групп методов – автоматической классификацией контента. Рассмотрим ее особенности.

Точность и уровень автоматизации

На сегодняшний день, технологии автоматической классификации и кластеризации документов имеют лишь ограниченное практическое применение. Ряд эмпирических исследований показывает, что точность классификации доходит до 70-80% [5, 16, 22]. Однако это означает, что 20-30% документов классифицируются неправильно и не будут найдены пользователем при использовании сформированного набора категорий. Такой объем недоступной для нахождения информации является неприемлемым для большинства практических задач. Поэтому, организации вынуждены вводить ручной контроль классифицируемых документов, значительно снижая общую пропускную способность системы. Фактически, все документы просматриваются вручную для снижения процента ошибок классификации, и экономический эффект от ее автоматизации сводится к сокращению временных затрат на отнесения документа к той или иной категории. Однако на просмотр и поверхностное осмысление документа человеком затрачивается значительно больше времени, чем на размещение его в одной из известных ему папок. И, так как это действие выполняется в

любом случае, общая экономия трудозатрат при использовании технологии автоматической классификации оказывается значительно меньше желаемой.

Объем обучающей выборки

Рассматриваемые исследования эффективности классификации использовали большие объемы обучающих выборок и сравнительно небольшое количество категорий (например, 5000 документов и 5 категорий в [5]). В этих исследованиях было показано, что точность классификации возрастает с ростом объема обучающей выборки. Однако при разработке системы управления знаниями, организация столкнется с ситуацией, когда количество категорий будет большим, а количество документов в них – малым. Эта ситуация особенно характерна на ранних этапах использования системы и при появлении новых категорий. В результате алгоритмы классификации покажут значительно меньшую точность, чем на больших выборках. В таких ситуациях использование автоматических методов классификации будет нерациональным, и следует, скорее, рассматривать технологии улучшения ручной классификации.

Неявные правила классификации

В ряде случаев, даже полное семантическое понимание документа является недостаточным для его корректной классификации. Пользователи могут использовать и используют внешние знания о содержании документа для определения категорий, к которым он относится. Эти знания, обычно, не существуют в виде формализованных правил и могут меняться с течением времени [19]. В данном случае, алгоритмы автоматической классификации наталкиваются на неустранимую проблему – необходимость обладания внешним контекстом для корректной интерпретации документа и его положения в общей структуре категорий. Отчасти, эта проблема может быть смягчена использованием баз знаний “здравого смысла”, содержащих в себе множество общепринятых контекстов. Однако на сегодняшний день такие базы находятся на ранних этапах развития и трудоемкость их создания колоссальна. Например, на формирование базы проекта CYC, по оценкам его руководителя, требуется не менее 40 лет [14]. Но даже при наличии таких общедоступных баз, остается значительный “организационный контекст”, специфичный для каждой компании и содержащийся в головах сотрудников.

Частным случаем неявных правил классификации является отбор важных или полезных документов, их группировка по состоянию работы над документом (в проекте, согласован, архивный). Также группировка документов может выполняться не по схожести их содержимого, а по порядку, в котором их следует читать (например, главы электронной книги или презентации обучающего курса). Другой способ - использовать признак принадлежности к одной тематике, не выраженной в самих документах (например, графическая схема устройства, текстовое руководство пользователя и таблицы с параметрами и характеристиками).

Преимущества нового метода состоят в следующем:

- не требует обучающей выборки и может применяться с первого дня использования информационной системы;
- учитывает неявные правила классификации, рассматривая интерпретацию контента пользователем, а не сам контент;
- в отличие от методов автоматической классификации, новый метод будет эффективно работать при непрерывном изменении классификатора, в то время как существующие подходы будут требовать переобучения и перенастройки.

Ограничения нового метода напрямую следуют из его принципа работы:

- оптимизирует только те области классификатора, которые активно используются. Для формирования рекомендаций, необходимо получить статистику использования одних и тех же областей несколькими пользователями. Существующие подходы могут строить классификаторы без участия пользователей;
- устраняет проблемы классификации после их возникновения, вместо их предотвращения.

Описанный метод может применяться в аналитической работе для сбора подробной статистики о качестве классификатора и эффективности его использования сотрудниками, так и для практического исправления выявленных недостатков классификатора в автоматическом и ручном режимах. Метод подразумевает применение в уже существующих и разрабатываемых системах, использующих иерархическую классификацию хранимой информации, таких как: файловые системы, структурированные базы знаний, ECM и DMS системы.

Механизм интеграции должен решать следующие задачи: (1) Извлечение событий, описывающих действия пользователя, и (2) извлечение структуры хранилища. Вторая задача решается стандартными средствами системы, с которой выполняется интеграция. В тоже время, задача извлечения событий о действиях пользователя является нетривиальной во многих системах и подробнее описана в [3].

Заключение

В результате исследования, был разработан и апробирован метод оптимизации классификатора на основе наблюдения за его использованием. Была показана необходимость и актуальность такой задачи, а также принципиальные недостатки альтернативных автоматизированных подходов. Для анализа действий пользователя разработана модель навигации в иерархическом хранилище, которая позволяет определить выполняемые пользователем поисковые задачи и оценить их успешность, основываясь только на элементарных событиях информационной системы. На основе модели был предложен способ выявления проблем классификации и способов их устранения.

Метод был реализован в рамках программного прототипа, который использовался для оценки его эффективности. Проведенные эксперименты показали, что 38% поисковых путей содержали в себе один или несколько проблемных шагов, и что применение метода сокращает время поиска в иерархическом хранилище на 12%. Данный эксперимент проводился на основе относительно качественно структурированной базы знаний консалтинговой фирмы.

Темой дальнейших исследований может стать повышение точности выделения комплексных событий в информационных системах и отделение действий пользователя от действий самой автоматизированной системы и ее модулей. Дополнительно, можно разработать интегрированные методы, объединяющие рассмотренный подход с уже существующими.

Литература

1. Гаврилова Т.А., Горовой В.А., Болотникова Е.С. Оценка когнитивной эргономичности онтологии на основе анализа графа. // Искусственный интеллект и принятие решений. 2009. С. 33–41.
2. Гаврилова Т.А., Кудрявцев Д.В., Горовой В.А. Модели и методы формирования онтологий. // Научно-технические ведомости Санкт-Петербургского государственного политехнического университета. 2006. С. 21–28.
3. Прокофьев А.М. Комплексная обработка событий в иерархических хранилищах // Информатика, моделирование, автоматизация проектирования. УлГТУ. 2010.
4. Apostolou D., Mentzas G., Abecker A. Managing knowledge at multiple organizational levels using faceted ontologies. // The Journal of Computer Information Systems. 2008. № 49. С. 32–49.
5. Barnett T. и др. Machine learning classification for document review // ICAIL 2009 Workshop on Supporting Search and Sensemaking for Electronically Stored Information in Discovery. Retrieved July. , 2009. С. 2009.
6. den Berg J.E. и др. Effects of the ISIS Recommender System for navigation support in self-organised Learning Networks. // Proceedings of Special Track on Technology Support for Self-Organised Learners. 2008.
7. Buitelaar P., Cimiano P., Magnini B. Ontology learning from text: methods, evaluation and applications. // Computational Linguistics. 2005. № 32.
8. Choi N., Song I.Y., Han H. A survey on ontology mapping. // ACM Sigmod Record. 2006. № 35. С. 34–41.
9. Flouris G. и др. Ontology change: classification and survey. // The Knowledge Engineering Review. 2008. № 23. С. 117–152.
10. Gómez-Pérez A. Evaluation of taxonomic knowledge in ontologies and knowledge bases. // 1999.
11. Harish B.S., Guru D.S., Manjunath S. Representation and classification of text documents: A brief review. // International Journal of Computer Applications IJCA. 2010. С. 110–119.
12. Jain A.K. Data clustering: 50 years beyond K-means. // Pattern Recognition Letters. 2010. № 31. С. 651–666.
13. Lei Y. и др. An infrastructure for acquiring high quality semantic metadata. // The Semantic Web: Research and Applications. 2006. С. 230–244.
14. Matuszek C. и др. An introduction to the syntax and content of Cyc // Proceedings of the 2006 AAAI Spring Symposium on Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering. 2006. С. 44–49.
15. Nguyen V. Ontologies and Information Systems: A Literature Survey. // 2011.
16. Pong J.Y. и др. A comparative study of two automatic document classification methods in a library setting. // Journal of Information Science. 2008. № 34. С. 213.
17. Porzel R., Malaka R. A task-based approach for ontology evaluation // ECAI Workshop on Ontology Learning and Population. Valencia, Spain. 2004.
18. Robins D.B. Complex Event Processing. Redmond, WA: University of Washington. 2010.
19. Roitblat H.L., Kershaw A., Oot P. Document categorization in legal electronic discovery: computer classification vs. manual review. // Journal of the American Society for Information Science and Technology. 2010. № 61. С. 70–80.
20. Sellami Z. и др. Ontology Co-construction with an Adaptive Multi-Agent System: Principles and Case-Study. // Knowledge Discovery, Knowlege Engineering and Knowledge Management. 2011. С. 237–248.

21. Tallis M., Gil Y. Designing scripts to guide users in modifying knowledge-based systems // Proceedings of the Sixteenth National Conference on Artificial Intelligence. 1999. С. 242–249.
22. Trieschnigg D. и др. MeSH Up: effective MeSH text classification for improved document retrieval. // Bioinformatics. 2009. № 25. С. 1412.
23. Wach E.P. Automated Ontology Evolution as a Basis for Adaptive Interactive Systems. // 2010.
24. Zablith F. Evolva: Towards Automatic Ontology Evolution. // 2008.
25. Zavitsanos E., Paliouras G., Vouros G.A. A distributional approach to evaluating ontology learning methods using a gold standard // Proc. of 3rd Workshop on Ontology Learning and Population (OLP3) at ECAI. 2008.

Прокофьев Александр Михайлович. Аспирант Санкт-Петербургского государственного политехнического университета. Окончил СПб ГПУ в 2008 году. Автор 13 печатных работ. E-mail: alex@8km.ru