HIGH TECH

Х. МАРМАНИС, Д. БАБЕНКО

АЛГОРИТМЫ ИНТЕЛЛЕКТУАЛЬНОГО ИНТЕРНЕТА

ПЕРЕДОВЫЕ МЕТОДИКИ СБОРА, АНАЛИЗА И ОБРАБОТКИ ДАННЫХ

СИМВОЛ®

Algorithms of the Intelligent Web

Haralambos Marmanis
Dmitry Babenko

MANNING

Алгоритмы интеллектуального Интернета

Хараламбос Марманис,
Дмитрий Бабенко

Санкт-Петербург — Москва
2011

Харалambос Марманис,
Дмитрий Бабенко

Алгоритмы интеллектуального Интернета

Передовые методики сбора, анализа и обработки данных

Перевод М. Низовец

Главный редактор	*А. Галунов*
Зав. редакцией	*Н. Макарова*
Выпускающий редактор	*П. Щеголев*
Научный редактор	*С. Щербак*
Редактор	*Т. Темкина*
Корректор	*С. Минин*
Верстка	*К. Чубаров*

Марманис Х., Бабенко Д.

Алгоритмы интеллектуального Интернета. Передовые методики сбора, анализа и обработки данных. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 480 с., ил.

ISBN 978-5-93286-186-8

Важный аспект современных коммерчески успешных приложений – применение методик, позволяющих осуществлять обработку информации и добавлять средства интеллектуальной поддержки. Многочисленные примеры успешных проектов, основанных на применении таких методик, включают такие широко известные бренды, как Google, Netflix и Amazon. Эта книга о том, как построить алгоритмы, формирующие интеллектуальное ядро таких веб-приложений.

В книге рассматриваются пять важных категорий алгоритмов: поиск, выработка рекомендаций, создание групп, классификация и ансамбли классификаторов. Исходный код написан на языке Java, тем не менее программистам, знающим другой объектно-ориентированный язык, вполне по силам разобраться в этом коде и использовать общие принципы с учетом своей специфики. Материал в равной степени применим к различным приложениям – от утилит мобильной связи до традиционных настольных приложений.

Издание в первую очередь адресовано программистам и веб-разработчикам, однако множество примеров и новых идей будут полезны и руководителям разного уровня, желающим лучше разобраться в соответствующих технологиях и предлагаемых возможностях с точки зрения бизнеса.

ISBN 978-5-93286-186-8
ISBN 978-1-933988-66-5 (англ)

Издательство «Символ-Плюс». 199034, Санкт-Петербург, 16 линия, 7,
тел. (812) 380-5007, www.symbol.ru. Лицензия ЛП N 000054 от 25.12.98.

Подписано в печать 29.04.2011. Формат 70×100 $^{1}/_{16}$. Печать офсетная.
Объем 30 печ. л. Тираж 1200 экз. Заказ №
Отпечатано с готовых диапозитивов в ГУП «Типография «Наука»
199034, Санкт-Петербург, 9 линия, 12.

Оглавление

Предисловие

Во время обучения в аспирантуре я познакомился с проблемами машинного обучения и, в частности, с распознаванием образов. Центральное место в моей работе занимало математическое и численное моделирование, но возможность распознавать образы в большом объеме данных нашла очевидные применения во многих областях. Последующие годы подвели меня к теме машинного обучения даже ближе, чем я себе это представлял.

В 1999 году я оставил академические занятия и приступил к работе в промышленном секторе. В одном из моих консалтинговых проектов мы пытались определять риск сердечной недостаточности на основании (главным образом) данных ЭКГ пациентов. В подобных задачах точная математическая формулировка либо невозможна, либо ее нельзя реализовать. Моделирование (наше программное обеспечение) должно было опираться на методы, способные повысить точность прогнозирования за счет имеющегося набора историй болезни пациентов, для которых риск сердечной недостаточности уже был диагностирован кардиологом. Другими словами, мы искали методы, которые могли бы «обучаться» на своих входных данных.

Тогда, в 1990-х, благодаря стечению обстоятельств новая индустрия развивалась быстро. Интернет стал вездесущим! По закону Мура, ЦП продолжали наращивать быстродействие и дешеветь. Модули ОЗУ, жесткие диски и другие компьютерные компоненты следовали тем же тенденциям совершенствования возможностей и снижения стоимости. Не отставая от них, продолжала возрастать пропускная способность типичного сетевого подключения, которое одновременно становилось все более доступным по цене. Более того, были востребованы и появились надежные технологии разработки веб-приложений, а быстрое распространение проектов с открытым исходным кодом, охватывающих все направления разработки программного обеспечения, ускоряло развитие соответствующих методов. Все эти факторы внесли свой вклад в построение всеобъемлющей цифровой экосистемы, которую мы сегодня называем Интернетом.

Естественно, первостепенной задачей, вставшей перед представителями нашей профессии – разработчиками программного обеспечения и веб-разработчиками всего мира, – стала разработка таких технологий, которые позволили бы создавать надежные, масштабируемые и эстетически

привлекательные веб-приложения. Таким образом, за последние десять лет было приложено много сил, чтобы достичь этих целей, и налицо значительный прогресс. Разумеется, совершенство – это конечная цель, а не состояние, так что нам еще есть куда расти. Но что касается надежности, масштабируемости и эстетической привлекательности, то здесь, по-видимому, наша эффективность уже не повышается. Эра «обустройства» интернет-приложений более или менее закончилась. Простое агрегирование данных и несложные модели типа «запрос пользователя/ответ системы», основанные на заданной логике, достигли состояния зрелости.

Сегодня в некоторых приложениях можно обнаружить другую волну инноваций, и эта волна довольно быстро распространяется благодаря повышению уровня просвещения. Именно такие приложения мы называем в этой книге *интеллектуальными приложениями*. В отличие от традиционных, интеллектуальные приложения настраивают свое функциональное поведение в соответствии с полученными ими входными данными, во многом подобно тому, как моя программа моделирования должна была диагностировать риск сердечной недостаточности по данным ЭКГ.

За последние пять лет мне стало ясно, что многие методики, применяемые в интеллектуальных приложениях, труднодоступны для подавляющего большинства профессионалов в области программного обеспечения. Думаю, главные причины здесь две. Во-первых, коммерческий потенциал инноваций в этих областях способен обеспечить гигантскую финансовую прибыль. Имеет смысл (с финансовой точки зрения) защищать право собственности на компоненты конкретных приложений и скрывать важнейшие детали реализаций. Вторая причина, по которой базовые методики так долго остаются в тени, заключается в том, что почти все они зародились как научные исследования и, следовательно, в значительной мере опираются на язык математики. С первой причиной ничего не поделаешь. Но объем общедоступных знаний так велик, что возникает вопрос: является ли непреодолимой вторая причина? Мой краткий ответ – громкое и решительное «Нет!». Чтобы получить развернутый ответ, вам придется прочесть эту книгу!

Я решил написать эту книгу с целью продемонстрировать, что некоторые из упомянутых методик можно представить в виде алгоритмов, не предполагая при этом наличия у читателя серьезной математической подготовки. Задача этой книги – дать вам методики, помогающие встроить в приложение интеллектуальное поведение с минимальным по возможности обращением к математическому аппарату. Все необходимые математические выкладки содержатся в программном коде – в виде алгоритма.

Сначала я хотел использовать для представления этих методик библиотеки с открытым исходным кодом. Но большинство таких библиотек разрабатывались конъюнктурно и нередко без малейшего намерения обучать положенным в их основу методикам. То есть, скорее всего, это

туманный код, который даже читать нелегко, не говоря уже о том, чтобы понимать! Я понял, что мой предполагаемый читатель больше выиграл бы от ясной, хорошо документированной программной базы. На этом распутье ко мне присоединился Дмитрий, и это он написал большую часть программного кода, который вы встретите в этой книге.

Количество книг, посвященных этой новой и захватывающей теме, будет медленно, но верно расти. Настоящая книга – только введение в область, которая уже обширна и продолжает быстро развиваться. Естественно, число рассматриваемых алгоритмов должно было быть ограниченным, а объяснения краткими. Моя задача сводилась к тому, чтобы отобрать несколько вопросов и дать им хорошее толкование, вместо того чтобы пытаться охватить как можно больше с риском запутать вас или просто создать рецептурный справочник.

Надеюсь, мы справились с поставленной задачей, придерживаясь следующих четыре правил:

- Работать с понятными примерами, сосредоточившись именно на них
- Использовать высокоуровневые сценарии, которые отражают такое применение алгоритмов, как если бы вы внедрили их в собственное приложение
- Помогать вам экспериментировать с программным кодом и осваивать его с помощью многочисленных заданий из разделов «сделать»
- Писать первоклассный удобочитаемый программный код

Итак, захватите свой любимый горячий напиток, сядьте поудобнее и опробуйте несколько умных приложений – они здесь, чтобы остаться!

Хараламбос Марманис

Благодарности

Мы хотели бы выразить признательность сотрудникам издательства Manning, предоставившим возможность опубликовать эту работу. Они не только помогли привести рукопись к ее окончательному виду, но и терпеливо дождались завершения работы над ней, для чего потребовалось много больше времени, чем мы планировали изначально. В частности, нам хотелось бы поблагодарить Марджана Бейса (Marjan Bace), Джеффа Блейела (Jeff Bleiel), Карен Тетмайер (Karen Tegtmeyer), Меган Йоки (Megan Yockey), Мери Пирджис (Mary Piergies), Морин Спенсер (Maureen Spencer), Стивена Хонга (Steven Hong), Рона Томича (Ron Tomich), Бенджамина Берга (Benjamin Berg), Элизабет Мартин (Elizabeth Martin), а также всех тех членов коллектива издательства Manning, кто работал над этой книгой, но чьих имен мы не знаем. Спасибо за ваш напряженный труд.

Нам также хотелось бы признать своевременными, действенными и ценными замечания рецензентов и посетителей нашего авторского форума Author Online. Ваши отклики помогли во многом улучшить эту книгу. Мы понимаем, как ограниченно и ценно «свободное» время каждого профессионала, поэтому очень благодарны за ваш вклад.

Мы выражаем особую благодарность за многократное прочтение нашей рукописи на разных стадиях ее написания и за предоставленные комментарии следующим рецензентам: Роберту Хэнсону (Robert Hanson), Самиту Пэлу (Sumit Pal), Карлтону Гибсону (Carlton Gibson), Дэвиду Хэнсону (David Hanson), Эрику Свенсону (Eric Swanson), Франку Вонгу (Frank Wang), Бобу Хатчисону (Bob Hutchison), Крейгу Уоллсу (Craig Walls), Николасу Хейнли (Nicholas C. Heinle), Владу Горски (Vlad Gorsky), Алессандро Галло (Alessandro Gallo), Крейгу Ланкастеру (Craig Lancaster), Джейсону Колтеру (Jason Kolter), Мартину Флетчеру (Martyn Fletcher) и Скотту Доусону (Scott Dawson). Последняя, но ничуть не меньшая благодарность в адрес Аджая Бхандари (Ajay Bhandari), который был корректором и последним вычитал главы и проверил программный код перед сдачей книги в печать.

Х. Марманис

Мне хотелось бы поблагодарить своих родителей, Еву и Александра. Это от них у меня неуемное любопытство и страсть к знанию, которые заставляют меня писать и заниматься исследованиями ночи напролет. Этот долг слишком велик, чтобы его можно было оплатить за одну жизнь.

Я от всего сердца благодарю мою дорогую жену Аврору и трех наших сыновей: Никоса, Лукаса и Альберта – величайшую гордость и радость моей жизни. Я всегда буду признателен за их любовь, терпение и понимание. Ежеминутное любопытство моих детей всегда вдохновляло меня на изыскания. Бесконечная благодарность родителям моей жены, Кучи и Хосе, моим сестрам Марии и Катерине, а также моим лучшим друзьям Майклу и Антонио за их постоянное ободрение и безоговорочную поддержку.

Не могу не отдать должное всесторонней поддержке докторов наук Амилькара Авенданьо (Amilcar Avendaco) и Марии Балерди (Maria Balerdi), которые многому научили меня в области кардиологии и на первых порах субсидировали мою исследовательскую работу. Я также выражаю свою благодарность профессору Леона Купера (Leon Cooper) и многим другим замечательным сотрудникам университета Брауна (Brown University), чей энтузиазм в изучении человеческого мозга вдохновил таких, как я, заняться интеллектуальными приложениями.

Моим коллегам в прошлом и в настоящем Аджаю Бхандари (Ajay Bhandari), Кавите Канеткар (Kavita Kanetkar), Александру Петрову (Alexander Petrov), Кишоре Кирдату (Kishore Kirdat) и многим другим, кто поощрял и поддерживал все мои начинания, связанные с искусственным интеллектом: эти скупые строки не могут выразить всю глубину моей благодарности вам.

Д. Бабенко

Прежде всего, хочу поблагодарить свою любимую жену Елену. На завершение работы над этой книгой ушло больше года, и моей жене пришлось смириться с тем, что муж все свое время отдавал службе или работе над книгой. Ее поддержка и ободрение создали идеальную среду для того, чтобы я мог написать эту книгу.

Мне хотелось бы поблагодарить всех моих бывших и нынешних коллег, которые повлияли на мою профессиональную жизнь, став ее вдохновителями: Константина Бобовича (Konstantin Bobovich), Пола А. Денниса (Paul A. Dennis), Кита Лоулесса (Keith Lawless) и Кевина Беделла (Kevin Bedell).

В заключение я также хотел бы поблагодарить своего соавтора доктора Марманиса за приглашение участвовать в этом проекте.

Об этой книге

Впечатление от современных веб-приложений в основном складывается за счет насыщенного возможностями пользовательского интерфейса (user interface, UI). Менее известный аспект современных приложений – применение методик, позволяющих осуществить интеллектуальную обработку информации и добавить полезные свойства, которые нельзя обеспечить другими средствами. Многочисленные примеры успешных проектов, основанных на применении таких методик, включают такие широко известные бренды, как Google, Netflix и Amazon. Эта книга о том, как построить алгоритмы, формирующие интеллектуальное ядро этих веб-приложений.

В книге рассматриваются пять важных категорий алгоритмов: поиск, выработка рекомендаций, создание групп, классификация и ансамбли классификаторов. По каждому из этих тематических разделов можно было бы написать отдельную книгу, и понятно, что целью данной книги не является исчерпывающее их рассмотрение. Эта книга – введение в основы пяти названных тематических разделов. Это попытка познакомить вас с базовыми алгоритмами интеллектуальных приложений, а не охватить сразу все алгоритмы вычислительного интеллекта. Книга адресована самой широкой аудитории и требует от читателя минимум предварительных знаний.

Отличительная особенность этой книги – специальный раздел в конце каждой главы. Мы назвали его «Сделать» (To Do), предназначив не только для дополнительного материала. Каждый из этих разделов позволит вам глубже вникнуть в тему соответствующей главы. Кроме того, разделы «Сделать» призваны посеять зерно любопытства, заставляющее вас задуматься о новых возможностях, равно как о связанных с ними проблемах, всплывающих в реальных приложениях.

В этой книге широко применяется библиотека сценариев BeanShell. Этот выбор служит достижению двух целей. Первая цель – обеспечить знакомство с алгоритмами на более легком для восприятия уровне, прежде чем углубляться в дебри подробностей. Вторая цель – наметить шаги, которые вам следует предпринять, чтобы включить эти алгоритмы в свое приложение. В большинстве случаев можно воспользоваться данной библиотекой, написав при этом лишь несколько строк кода! Более того, чтобы гарантировать сохранность и упростить сопровождение

исходного кода, мы создали на специальном сайте Google проект *http://code.google.com/p/yooreeka/*.

Структура книги

Книга состоит из семи глав. Глава 1 – введение. В главах 2–6 рассматриваются поиск, выработка рекомендаций, создание групп, классификация и ансамбли классификаторов соответственно. Глава 7 обобщает материал предыдущих глав, но в ней рассматривается новая тема в контексте одного приложения.

Хотя в главах встречаются ссылки на другие главы, материал организован так, что главы 1–5 можно прочесть независимо друг от друга. Глава 6 опирается на материал главы 5, так что ее трудно будет читать отдельно. Глава 7 тоже не является самостоятельной, поскольку в ней затрагивается материал всей книги.

В главе 1 приведен обзор интеллектуальных приложений и рассмотрены некоторые примеры их полезности. Дано практическое определение интеллектуальных веб-приложений, описан набор принципов проектирования. Перечислены шесть крупных классов веб-приложений, в которых можно задействовать интеллектуальные алгоритмы, рассматриваемые в этой книге. Также приводятся сведения о происхождении этих алгоритмов и рассматривается их связь с такими областями, как искусственный интеллект, машинное обучение, анализ данных и мягкие вычисления. В конце главы приведены восемь ошибок проектирования, часто встречающихся на практике.

Глава 2 начинается с описания процедуры поиска, основанного на традиционных методиках извлечения информации. Обобщив традиционный подход, мы перейдем к поиску вне индексов, в том числе к самому известному алгоритму оценки ссылок – PageRank. Один из разделов главы посвящен уточнению результатов поиска путем анализа экранных данных. Эта методика изучает предпочтения пользователя в отношении конкретного сайта или тематического раздела, и ее можно в значительной степени развить и дополнить, чтобы обеспечить новые возможности.

Также в этой главе описан поиск документов, не являющихся веб-страницами, с помощью нового алгоритма, который мы называем DocRank. Это довольно перспективный алгоритм, но важнее то, что он показал: математические принципы оценки ссылок несложно дополнить и, внеся необходимые изменения, применять в других контекстах. Также в этой главе говорится о некоторых проблемах, возникающих при работе в очень крупных сетях. В конце главы затрагивается проблема правдоподобия и контроля достоверности результатов поиска.

В главе 3 вводятся важнейшие понятия расстояния и сходства. В ней представлены две обширные категории методик выработки рекомен-

даций: коллаборативная фильтрация и подход на основе контента. В качестве контекста для выработки рекомендаций взят вымышленный онлайн-магазин музыкальных записей. Также здесь рассматриваются два примера более общего характера. Первый – гипотетический веб-сайт, который с помощью интерфейса Digg API извлекает контент пользователей, чтобы рекомендовать им статьи, которые они еще не видели. Второй пример, связанный с рекомендациями по кинофильмам, вводит понятие нормализации данных. В этой главе мы также оцениваем точность вырабатываемых рекомендаций исходя из среднеквадратической погрешности.

В главе 4 представлены алгоритмы кластеризации. Есть много прикладных областей, где можно применить кластеризацию. Теоретически, для кластеризации пригоден любой набор данных, объекты которого можно определить в терминах значений атрибутов. В этой главе мы рассматриваем создание групп сообщений на форуме и выявление пользователей веб-сайта, между которыми есть сходство. Здесь также предлагаются общий обзор типов кластеризации и полные реализации шести алгоритмов: одной связи, средней связи, минимального остовного дерева с одной связью, k-средних, ROCK и DBSCAN.

В главе 5 представлены алгоритмы классификации, которые являются важнейшими компонентами интеллектуальных приложений. В начале главы описаны онтологии, основанные на трех фундаментальных блоках – концептах, образцах и атрибутах. Классификация представлена как задача назначения «наилучшего» концепта данному образцу. Классификаторы отличаются друг от друга способом представления и способом оценки этого оптимального назначения. В этой главе дается обзор категорий классификации, который включает *двоичную* и *многоклассовую* классификации, *статистические* и *структурные* алгоритмы. Здесь также представлены три этапа жизненного цикла классификатора: обучение, тестирование и эксплуатация.

Изложение материала в главе 5 продолжает представлять на высоком уровне регрессионные алгоритмы, байесовские алгоритмы, алгоритмы на основе правил, функциональные алгоритмы, алгоритмы ближайших соседей и нейронные сети. Подробно обсуждаются три методики классификации. В основе первой методики – наивный байесовский алгоритм, применяемый к единственному строковому атрибуту. Вторая методика основана на процессоре правил Drools, который является объектно-ориентированной реализацией алгоритма Rete и позволяет объявлять и применять правила в целях классификации. Третья методика представляет и применяет *вычислительные нейронные сети*; в этой главе предлагается элементарная, но надежная реализация для построения нейронных сетей общего назначения. Глава 5 также предупреждает вас о проблемах, связанных с достоверностью и вычислительными требованиями классификации, которые необходимо решить, прежде чем внедрять возможности классификации в приложения.

В главе 6 рассматриваются ансамбли классификаторов – модифицированные методики, способные повысить точность классификации, обеспечиваемую одиночным классификатором. Основной пример, который рассматривается в этой главе, – оценка кредитоспособности для приложения, обслуживающего ипотечное кредитование. Подробно представлены метод формирования ансамблей классификаторов (bagging) и метод улучшения слабых моделей (boosting). Также в этой главе представлена реализация алгоритма улучшения Бреймана arc-x4.

В главе 7 демонстрируется применение интеллектуальных алгоритмов в контексте новостного портала. Мы обсуждаем технические проблемы наряду с новыми, ценными с точки зрения бизнеса возможностями, которые интеллектуальные алгоритмы могут добавить в приложение. Например, алгоритм кластеризации можно использовать для объединения в группы похожих новостных сообщений, а также для расширения области видимости релевантных новостных сообщений с помощью *перекрестных ссылок*. В этой главе мы в общих чертах описываем адаптацию различных интеллектуальных алгоритмов и их объединение для достижения конкретной цели.

Специальный раздел «Сделать»

В каждой главе, начиная со второй, есть раздел с несколькими заданиями для выполнения, которые послужат вам руководством при изучении различных тем. Нам как разработчикам программного обеспечения нравится название «Сделать» («ToDo»): подразумевая обязательность выполнения, оно менее формально, чем такие названия, как «Упражнения» («Exercises»).

Одни задания разделов «Сделать» нацелены на более глубокое освоение темы данной главы, другие предоставляют отправную точку для изучения вопросов, расширяющих эту тему. Выполнив эти задания полностью, вы получите более глубокое и полное представление об интеллектуальных алгоритмах.

Везде, где это уместно, наш программный код снабжен тегами `TODO`, которые должны быть видны в интегрированных средах разработки (Integrated Development Environment, IDE), например в среде Eclipse, если щелкнуть мышью на панели Tasks (Задачи). Если щелкнуть мышью на любом из этих заданий, соответствующая гиперссылка позволит увидеть фрагмент программного кода, связанный с этим заданием.

Кому адресована эта книга

Книга «Алгоритмы интеллектуального Интернета» написана для разработчиков программного обеспечения и веб-разработчиков, которым хотелось бы подробнее ознакомиться с новой разновидностью алгоритмов, обеспечивающих интеллектуальную поддержку многих коммерчески успешных приложений. Поскольку исходный программный код

написан на языке Java, работающим с этим языком данная книга может показаться более привлекательной, чем тем, кто использует другие языки программирования. Тем не менее пишущим на других языках вполне по силам изучить приведенный в этой книге программный код и, возможно, перевести его на предпочитаемый ими язык.

Книга полна примеров и идей, которые могут найти широкое применение, следовательно, она также может быть полезна техническим директорам, менеджерам по продукции и другим руководителям, желающим лучше разобраться в соответствующих технологиях и предлагаемых возможностях с точки зрения бизнеса.

Наконец, несмотря на слово *Интернет* в заглавии книги, ее материал в равной степени применим к самым разным программным приложениям, от утилит мобильной связи до традиционных настольных приложений, таких как текстовые редакторы и электронные таблицы.

Соглашения о примерах кода

Весь программный код в книге набран моноширинным шрифтом, выделяющим код из окружающего текста. Почти каждый пример программного кода снабжен комментариями, выделяющими ключевые понятия, а иногда и нумерованным списком в тексте, содержащим дополнительную информацию по этим коментариям. Те строки, которые не поместились по ширине, включают символ переноса на следующую строку.

Исходный программный код для этой книги доступен по адресу *http://code.google.com/p/yooreeka/downloads/list* или на сайте издательства *http://www.manning.com/marmanis/*.

Файл дистрибутива следует распаковать прямо на диск C:\. Мы предполагаем, что вы используете операционную систему Microsoft Windows, в противном случае вам придется модифицировать наши сценарии, чтобы заставить их работать в вашей системе. Каталог верхнего уровня архивного файла называется *iWeb2*: все ссылки на каталоги даются в книге с учетом этой корневой папки. Например, ссылка на каталог *data/ch02*, согласно нашей договоренности, означает абсолютный путь к каталогу *C:\iWeb2\data\ch02*.

Если указанный файл распакован, вы готовы выполнить сценарий создания сборки Ant. Просто перейдите в каталог сборки и выполните команду `ant`. Обратите внимание: сценарий Ant будет работать независимо от того, куда вы распаковали файл. Теперь вы готовы к тому, чтобы выполнить сценарий BeanShell, как описано в приложении A.

Авторы в Интернете

Издательством Manning Publications организован веб-форум Author Online, где можно оставить свой комментарий об этой книге, задать технические вопросы и получить помощь авторов и других пользователей. Чтобы подписаться на участие в нем, перейдите по ссылке *http://*

www.manning.com/marmanis/. На этой странице есть информация о том, какого рода помощь доступна, как зарегистрироваться, а также о правилах поведения на форуме. Кроме того, на этой странице представлены ссылки на исходный код рассматриваемых в книге примеров, список замеченных опечаток и другие материалы для скачивания.

Издательство Manning предоставляет читателям место для встреч, где может иметь место целевой диалог между отдельными читателями, а также между читателями и авторами. При этом оно не обязано обеспечивать определенный объем участия со стороны авторов, чей вклад в работу форума Author Online является добровольным (и бесплатным). Чтобы поддерживать заинтересованность авторов, советуем вам стараться задавать им разные стимулирующие вопросы!

Форум Author Online и архивы предыдущих обсуждений будут доступны на веб-сайте издательства до тех пор, пока книга находится в продаже.

1

Что такое интеллектуальный Интернет?

- Преимущества интеллектуальных веб-приложений
- Применение веб-приложений в реальном мире
- Встраивание интеллекта в веб-приложения

Итак, о чем эта книга? Прежде всего, давайте скажем, о чем в ней не говорится. Эта книга не о построении безупречного пользовательского интерфейса и не об использовании формата обмена данными JSON или языка запросов XPath, и даже не об архитектуре служб RESTful. Есть хорошие книги о приложениях, поддерживающих стандарт Web 2.0, где описана реализация проектов на основе технологии AJAX и общая практика применения развитых пользовательских интерфейсов. Немало и книг, посвященных другим эффективным веб-технологиям, таким как язык преобразований XSL Transformations (XSLT) и язык запросов XPath, масштабируемая векторная графика (Scalable Vector Graphics, SVG), технология веб-форм XForms, декларативный язык описания интерфейсов XUL и формат обмена данными JSON (JavaScript Object Notation).

Отправной точкой для этой книги послужил факт «тупости» большинства традиционных веб-приложений в том смысле, что отклик системы не учитывает введенные ранее данные и прежнее поведение пользователя. Мы говорим не о проблемах неудобного пользовательского интерфейса, а о том, что на ввод конкретных данных система всегда реагирует одинаково. Наш главный интерес – это построение веб-приложений,

которые учитывают введенные каждым пользователем данные, его поведение в системе на протяжении некоторого периода времени, а также другую потенциально полезную информацию, которая может быть доступна.

Например, вы начинаете заказывать еду с помощью веб-приложения и каждую среду заказываете рыбу. Вам было бы гораздо приятнее, чтобы по средам это приложение спрашивало: «Не хотите ли сегодня рыбу?», а не интересовалось: «Что бы вы хотели заказать сегодня?» В первом случае приложение *каким-то образом смогло понять*, что по средам вы предпочитаете рыбу. Во втором случае приложение пребывает в неведении относительно этого факта. То есть данные, созданные в результате вашего взаимодействия с сайтом, не влияют на то, как приложение выбирает контент страницы, или на то, как этот контент отображается. Обращение с вопросом, сформулированным на основе ранее выбранных пользователем вариантов, представляет собой новый вид взаимодействия между веб-сайтом и его пользователями. И мы могли бы назвать подобное свойство веб-сайтов *способностью к обучению*.

Следующим шагом в этом направлении будет корректировка взаимодействия интеллектуального веб-приложения с пользователем в зависимости от входных данных, полученных от других пользователей, так или иначе связанных друг с другом. Если ваш обычный рацион во многом совпадает с рационом Джона, приложение может рекомендовать вам несколько пунктов меню, обычных для Джона, которые вы никогда не заказывали; выработка рекомендаций рассматривается в главе 3.

Другой пример – сайт социальной сети, такой как Facebook, который предлагал бы чат или электронную конференцию (форум) с проверкой фактов. Под *проверкой фактов* (fact checking) мы подразумеваем следующее: по мере ввода вами текста сообщения то, что вы написали, в фоновом режиме подвергается проверке, чтобы удостовериться в точности утверждаемых вами фактов и даже в том, что они не противоречат вашим предшествующим сообщениям. Такая функциональность сходна с проверкой орфографии, вероятно, уже знакомой вам, но вместо того чтобы проверять правила грамматики, в данном случае проверяется набор фактов, который мог бы включать общеизвестные истины («Япония вторглась в Манчжурию в 1931 году»), ваши собственные представления о конкретном предмете («снижение налогов – благо для экономики») или простые факты частной жизни («посещение врача 11/11/2008»). Веб-сайты с таким функциональным поведением способны *делать логические выводы*; организацию соответствующей функциональности мы описываем в главе 5.

Можно утверждать, что по-настоящему эра интеллектуальных веб-приложений началась с появлением в Интернете поисковых систем, таких как Google. Разумеется, вы можете спросить: почему именно Google? Как решаются задачи, связанные с извлечением информации (поиском), было известно задолго до появления Google на мировой сцене.

Но исключительно важно то, что поисковые системы вроде Google плодотворно используют взаимосвязанность веб-контента. У Google был следующий тезис: на веб-страницах гиперссылки формируют базовую структуру, которую можно анализировать, чтобы определить важность различных страниц. В главе 2 мы подробно рассматриваем алгоритм PageRank, который обеспечивает такую возможность.

Расширяя обсуждаемую тему, можно сказать, что интеллектуальные веб-приложения изначально разрабатываются с прицелом на взаимодействующий и взаимосвязанный мир. Эти приложения проектируются для автоматического обучения, так что могут понимать вводимые пользователем данные или его поведение, либо и то и другое, и соответственно настраивать свой отклик. Совместное использование пользовательских профилей коллегами, друзьями и членами семьи на сайтах социальных сетей, таких как MySpace или Facebook, как и общий доступ к контенту телеконференций и онлайн-форумов наряду с высказываемыми там мнениями, создает новые уровни связанности, которые являются ключевыми для интеллектуальных веб-приложений и выходят за рамки обычных гиперссылок.

1.1. Примеры интеллектуальных веб-приложений

Рассмотрим приложения, в которых на протяжении последнего десятилетия применяется такого рода интеллект. Как уже говорилось, поворотным моментом в истории Интернета стало появление поисковых систем. Все то, что предлагалось в Интернете, в основном не менялось вплоть до 1998 года, когда в контексте поиска появилась и штурмом захватила рынок методика *оценки ссылок* (см. главу 2). Меньше чем за 10 лет компания Google Inc. из начинающего выросла в доминирующего игрока в этом технологическом секторе, благодаря, во-первых, успеху используемой ею технологии поиска на основе ссылок и, во-вторых, набору других предоставляемых ею услуг, таких как Google News (Новости Google) и Google Finance (Финансы Google).

Однако область применения интеллектуальных веб-приложений – это далеко не одни только поисковые системы. Розничный онлайн-торговец Amazon стал одним из первых интернет-магазинов, предлагающих своим пользователям рекомендации на основе образцов сделанных ими покупок. Вам, скорее всего, знакома эта функциональная возможность. Предположим, вы покупаете книгу о JavaServer Faces и книгу о языке программирования Python. Как только товары добавлены в корзину, на сайте Amazon вам порекомендуют дополнительные товары, так или иначе связанные с тем, что вы только что выбрали, например книги, посвященные технологиям AJAX или Ruby on Rails. Кроме того, при следующем вашем посещении веб-сайта Amazon вам могут порекомендовать эти же или другие связанные товары.

Еще одно интеллектуальное веб-приложение – Netflix[1], крупнейшая в мире интернет-служба видеопроката, где подписчикам (а их больше 7 000 000) предлагается 90 000 наименований DVD-дисков и пополняющаяся библиотека, которая предоставляет для онлайн-просмотра на пользовательских компьютерах больше 5000 полнометражных фильмов и телепрограмм. Пять лет подряд, с 2005 по 2007 годы, веб-сайт Netflix возглавлял рейтинг согласно опросу об удовлетворенности клиентов, проводимому раз в полгода исследовательскими компаниями ForeSee Results и FGI Research.

Отчасти успех этого сайта в Интернете – следствие того, что он предлагает пользователям легкий способ отбора фильмов из громадного списка. Ядро этой функциональной возможности – система выработки рекомендаций Cinematch. Ее задача – предсказать, понравится ли данный фильм конкретному пользователю, исходя из того, насколько ему нравятся или нет другие фильмы. Эта система – еще один замечательный пример интеллектуального веб-приложения. Предсказательные способности системы Cinematch настолько важны для Netflix, что в конце концов в октябре 2006 года был объявлен приз[2] в миллион долларов за усовершенствование возможностей данной системы. К октябрю 2007 года насчитывалось 28 845 конкурентов из 165 стран. В главе 3 мы предлагаем всестороннее рассмотрение алгоритмов, необходимых для создания системы рекомендаций, подобной Cinematch.

Интеллектуальные предсказания на основе мнений членов коллектива не ограничиваются рекомендациями по книгам или фильмам. Фирма PredictWallStreet собирает прогнозы своих пользователей относительно конкретных акций или индексов, чтобы выявить тенденции в оценках трейдеров и предсказать объем базовых активов. Мы не советуем вам снять со счета сбережения и заняться торговлей на бирже, полагаясь на их предсказания, но это – еще один пример творчески примененных методик, которые рассматриваются в этой книге, в сценарии из реальной жизни.

1.2. Базовые элементы интеллектуальных приложений

Давайте пристальнее вглядимся в то, какие отличительные особенности делают интеллектуальными приложения, рассмотренные нами в предыдущем разделе, и в частности подчеркнем различие между координацией совместной деятельности и интеллектом. Рассмотрим случай веб-сайта, где пользователи могут коллективно составлять документ. Такой веб-сайт вполне можно было бы квалифицировать как развитое веб-приложение согласно набору определений термина «раз-

[1] Источник: Netflix, Inc., *http://www.netflix.com/MediaCenter?id=5379*.

[2] Источник: *http://www.netflixprize.com//rules*.

витой» (advanced). Это приложение, разумеется, способствовало бы совместной деятельности многих пользователей в режиме онлайн, и оно могло бы предложить изобилующий возможностями, не вызывающий затруднений пользовательский интерфейс, гладкий технологический процесс и так далее. Но следует ли рассматривать такое приложение как интеллектуальное веб-приложение?

Документ, создаваемый на рассматриваемом веб-сайте, будет иметь бóльший объем, он будет глубже проработан и, наверное, более точен, чем другие документы, написанные каждым участником в отдельности. В этом отношении такой документ фиксирует не только знания каждого отдельного члена коллектива, но также результаты влияния взаимодействующих пользователей на конечный продукт. Следовательно, созданный таким способом документ фиксирует коллективное знание его создателей.

Идея не нова. Процессом определения стандарта в любой области науки или инженерии обычно управляет технический комитет. Этот комитет создает первый рабочий вариант документа, соединяющий знания экспертов и мнения нескольких заинтересованных групп и отвечающий потребностям коллектива, а не конкретного индивидуума или производителя. Затем первый рабочий вариант становится общедоступным для комментирования. В результате этого процесса окончательный вариант документа должен представить всю совокупность знаний сообщества и сформулировать рекомендации по отдельным требованиям, выявленным в этом сообществе.

Вернемся к нашему приложению. Как было определено до сих пор, это приложение позволяет зафиксировать коллективное знание как результат коллективных усилий, но пока еще не является интеллектуальным. Коллективный интеллект – термин весьма популярный, но часто понимаемый неверно, – требует коллективного знания и создается в результате коллективных воздействий, однако эти условия являются хоть и необходимыми, но не достаточными для того, чтобы характеризовать базовую программную систему как интеллектуальную.

Чтобы понять, из каких обязательных составляющих складывается то, что мы понимаем под *интеллектом* (intelligence), допустим еще, что у нашего воображаемого веб-сайта есть и другие возможности: по мере того как пользователь вводит свою статью документа, система идентифицирует другие документы, которые могут оказаться релевантными вводимому контенту, извлекает из них цитаты и помещает эти цитаты на боковую панель. Это могут быть документы из личной коллекции пользователя, а также совместно используемые участниками текущей работы или просто открытые, свободно доступные документы.

Пользователь может отметить фрагмент разрабатываемого документа и предписать системе уведомлять его, когда она обнаружит в сети документы, имеющие отношение к содержимому данного отрывка или, что, пожалуй, интереснее, когда консенсус сообщества, достигнутый

относительно этого содержимого, изменится в соответствии с некими критериями, заданными пользователем.

Для создания приложения с такими возможностями требуется много больше, чем привлекательный пользовательский интерфейс и платформа для совместных действий. Для этого требуется понимать произвольный текст. Нужна способность, позволяющая распознавать смысл сказанного в некотором контексте. Необходимо иметь возможность автоматически обрабатывать и группировать документы или фрагменты документов, которые содержат произвольный текст на естественном (человеческом) языке, по принципу их «похожести». Требуется некое структурированное знание о мире или, по меньшей мере, о предметной области рассуждения, к которой относится данный документ. Необходима способность сосредоточиться на конкретных документах, удовлетворяющих определенным правилам (пользовательским критериям), и сделать это быстро.

Таким образом, мы приходим к заключению, что приложения вроде Википедии и других общественных порталов отличаются от приложений вроде поисковой системы Google, рекламных программ Google Ads, программы выработки рекомендаций Netflix Cinematch и так далее. Приложения первого рода – это платформы для совместных действий, которые способствуют накоплению и обслуживанию коллективного знания. Приложения второго рода выделяют абстрактные структуры из всей совокупности коллективного знания и, следовательно, генерируют новый уровень возможностей и ценности. Мы завершаем этот раздел, суммируя элементы, необходимые для создания интеллектуального веб-приложения:

- *Агрегированный контент* (*aggregated content*) – другими словами, большой объем данных, относящихся к конкретному приложению. Агрегированный контент является скорее динамическим, чем статическим, и его источники, как и места его хранения, могут быть рассредоточены географически. Каждый фрагмент информации обычно связан или ссылается на множество других фрагментов информации.
- *Ссылочные структуры* (*reference structures*) – эти структуры обеспечивают одну или несколько структурных и семантических интерпретаций контента. Например, это относится к тому, что называют *фолксономией* (*folksonomy*), то есть к использованию тегов для динамического аннотирования контента и постоянного обновления представления коллективного знания для пользователей. Ссылочные структуры о мире или о конкретной предметной области знаний принадлежат одному из трех крупных классов: словари, базы знаний и онтологии (см. ссылки по теме в конце главы).
- *Алгоритмы* (*algorithms*) – этот элемент относится к слою модулей, позволяющих приложению задействовать информацию, скрытую

в данных, и использовать ее в целях абстракции (обобщения), предсказания и (со временем) улучшенного взаимодействия с их пользователями. Алгоритмы применяются к агрегированному контенту и иногда требуют присутствия ссылочных структур.

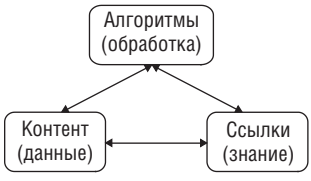

Рис. 1.1. *Треугольник интеллекта: три неотъемлемые составные части интеллектуальных приложений*

Наличие этих составляющих, сведенных вместе на рис. 1.1, обязательно для того, чтобы можно было определить приложение как интеллектуальное веб-приложение, и мы будем ссылаться на них на протяжении всей этой книги как на *треугольник интеллекта* (triangle of intelligence).

Целесообразно не смешивать эти три компонента друг с другом и построить такую модель их взаимодействия, которая в наибольшей степени отвечает вашим потребностям. Более подробно о разработке архитектуры мы поговорим в других главах, особенно в главе 7.

1.3. Что могут выиграть приложения от интеллектуальности?

Составляющие интеллекта, как показано в предыдущем разделе, можно обнаружить в широком диапазоне приложений: от сайтов социальных сетей до специализированных приложений для борьбы с терроризмом. В этом разделе мы рассмотрим примеры приложений из каждой категории. Наш список, разумеется, не полон, но он продемонстрирует, что рассматриваемые в этой книге методики во многих областях могут быть полезными, а в некоторых случаях и незаменимыми.

1.3.1. Сайты социальных сетей

За последние несколько лет наиболее заметное влияние на Интернет оказали веб-сайты социальных сетей. Такие сайты являются веб-приложениями, предоставляющими своим пользователям возможность установить факт своего присутствия в сети, не пользуясь для этого никакими иными средствами, кроме броузера и подключения к Интернету. Пользователи могут обмениваться друг с другом файлами (презентациями, видеофайлами, аудиофайлами), комментировать текущие события или страницы других пользователей, создать собственную со-

циальную сеть или присоединиться к уже имеющейся исходя из своих интересов. Два наиболее посещаемых[1] сайта социальных сетей – это MySpace и Facebook с сотнями миллионов и десятками миллионов зарегистрированных пользователей, соответственно.

По своему устройству эти сайты являются агрегаторами контента, так что первая составляющая для создания интеллекта легко доступна. Вторая составляющая также присутствует на этих сайтах. Например, на сайте MySpace контент распределяется по категориям с помощью основных ярлыков, таких как Books (Книги), Movies (Фильмы), Schools (Школы), Jobs (Вакансии) и так далее, которые отчетливо видны на странице сайта (рис. 1.2).

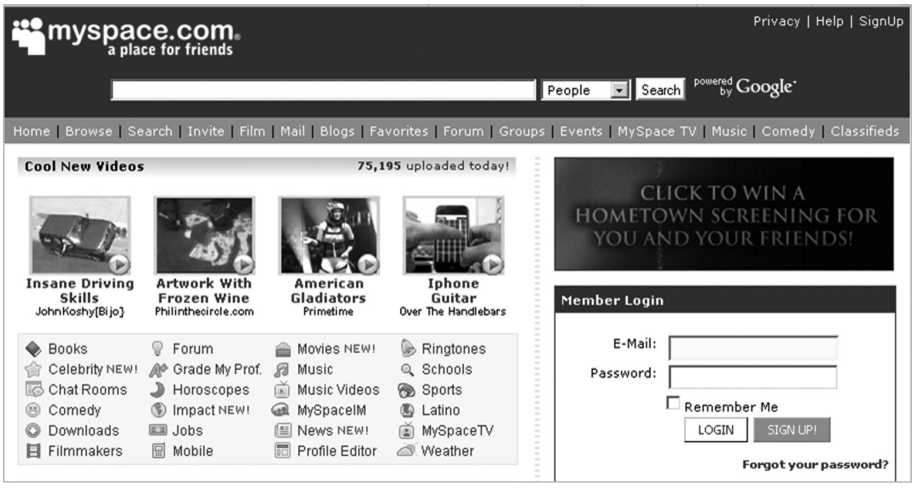

Рис. 1.2. *Категории, применяемые на веб-сайте MySpace*

Кроме того, эти категории верхнего уровня подвергаются дальнейшей детализации с помощью структур нижнего уровня, которые отделяют контент, относящийся к группе Classifieds (Объявления), от контента, относящегося к группам Polls (Опросы) или Weather (Погода). Наконец, большинство сайтов социальных сетей в состоянии порекомендовать своим пользователям новых друзей и новые сообщения (постинги, postings), которые могли бы представлять для них интерес. В этом они опираются на сложные алгоритмы создания предсказаний и абстракции накопленных данных и, следовательно, содержат все три составные части интеллекта.

[1] На основании данных о трафике, собранных службой Alexa.com в декабре 2007 года.

1.3.2. Гибридные веб–приложения (мэшапы)

На сайте DeveloperWorks фирмы IBM (*http://www.ibm.com/developerworks/spaces/mashups*) есть целый раздел, посвященный *гибридным веб-приложениям, или мэшапам* (mashups), с чрезвычайно подходящим определением: «Гибридные веб-приложения – это впечатляющий класс интерактивных приложений, которые извлекают контент из внешних источников данных, чтобы создать совершенно новые и инновационные службы». Другими словами, вы создаете сайт, используя контент и элементы пользовательского интерфейса, «заимствованные» у других. Еще один интересный сайт в контексте таких веб-приложений – это сайт ProgrammableWeb (*http://www.programmableweb.com*). Подходящее место для того, чтобы начать исследование мира гибридных приложений (рис. 1.3).

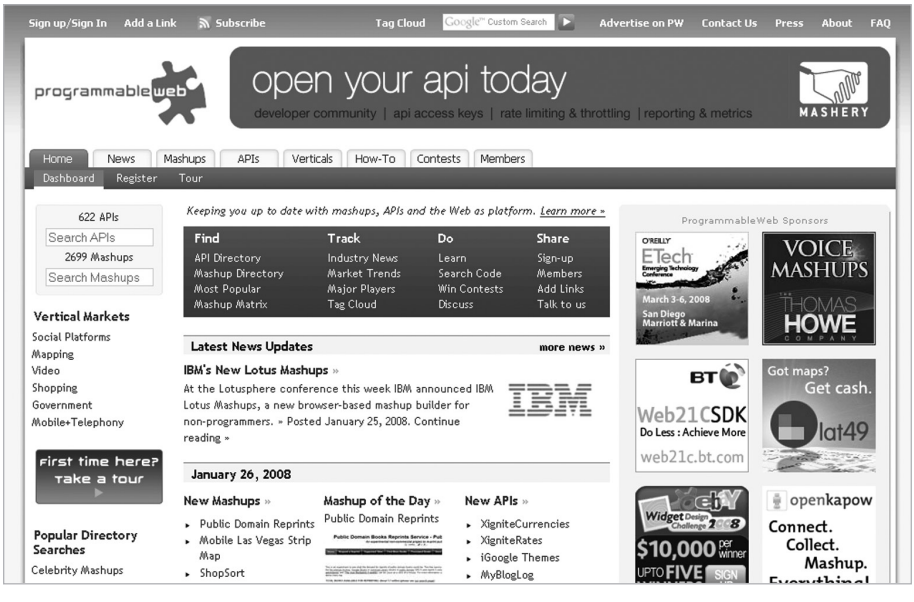

Рис. 1.3. *Чтобы подробнее изучить гибридные веб-приложения, посетите такие сайты, как ProgrammableWeb*

В нашем контексте гибридные веб-приложения важны, потому что основой для них является агрегированный контент; но, в отличие от сайтов социальных сетей, эти приложения не владеют тем контентом, который демонстрируют на экране, – по крайней мере, большая его часть принадлежит не им. Этот контент физически хранится в географически рассредоточенных пунктах и собирается воедино из разнообразных источников для создания уникальной презентации на основе вашего взаимодействия с приложением.

Однако не все гибридные веб-приложения являются интеллектуальными. Чтобы создать интеллектуальные гибридные веб-приложения, необходимо иметь возможность устранить разногласия или выявить сходные элементы контента, который мы пытаемся объединить.

В свою очередь, чтобы согласовать и классифицировать контент, нужно иметь одну или несколько ссылочных структур для интерпретации его смысла, а также ряд алгоритмов, которые позволяют установить, какие элементы этих ссылочных структур содержатся в различных фрагментах, или помогают понять, к какой категории следует отнести извлеченный из разных сайтов контент, чтобы обеспечить возможность его просмотра.

1.3.3. Порталы

Порталы (portal) и, в частности, новостные порталы – это еще один класс веб-приложений, для которых рассматриваемые в этой книге методики могут быть очень важны. По определению, такие приложения являются шлюзами к контенту, который разбросан по Интернету или, если речь идет о корпоративной локальной сети, по интранету. Это еще один случай, когда агрегированный контент рассредоточен, но доступен.

Лучший пример из данной категории – новостной портал Google News (*http://news.google.com*). Этот сайт собирает новостные сообщения из тысяч источников и автоматически группирует похожие сообщения под общим заголовком. Более того, каждая группа новостных сообщений отнесена к одной из новостных категорий, доступных по умолчанию, например Business (Бизнес), Health (Здоровье), World (Мир), Sci/Tech (Наука и техника) и так далее (рис. 1.4).

Вы можете создавать и собственные категории, определив, какого рода новости представляют для вас интерес. Опять-таки, мы видим, что подоплекой является агрегированный контент в связке со ссылочной структурой и рядом алгоритмов, которые могут выполнить требуемые задачи автоматически или, по меньшей мере, в полуавтоматическом режиме.

Многообещающим, с точки зрения встраивания интеллекта в ваш собственный портал, особенно из разряда социальных приложений, является проект OpenSocial (*http://code.go7ogle.com/apis/opensocial/*), а также несколько проектов, развивающихся вокруг него, например проект контейнера Apache Shindig. Исходная предпосылка проекта OpenSocial – создание общей базы интерфейса API, которая позволит разрабатывать приложения, взаимодействующие с большим и постоянно растущим числом веб-сайтов, таких как Engage, Friendster, hi5, Hyves, imeem, LinkedIn, MySpace, Ning, Oracle, orkut, Plaxo, Salesforce, Six Apart, Tianji, Viadeo и XING.

Рис. 1.4. *Веб-сайт Google News – интеллектуальное приложение портала*

1.3.4. Вики-сайты

Википедия (Wikipedia) не требует особого представления: вы, вероятно, уже посещали этот сайт или, по крайней мере, слышали о нем. Это вики-сайт, который не покидает десятку наиболее посещаемых ресурсов. *Вики* (wiki) – это доступное в компьютерной сети хранилище знаний. Вики-сайты используются социальными сообществами в Интернете и внутри корпораций в целях обеспечения общего доступа к знаниям.

Эти сайты, безусловно, являются агрегаторами контента. Кроме того, множество таких ресурсов, благодаря технологическому процессу создания страниц, имеют встроенную структуру, которая аннотирует контент. В Википедии можно отнести статью к какой-то *категории* (category) и связать с ней статьи, относящиеся к этой же тематике. Вики – многообещающая область для применения методик, рассматриваемых в этой книге. Например, вы могли бы создать или модифицировать свой вики-сайт таким образом, чтобы он автоматически распределял по категориям написанные вами страницы. Вики-страницы могли бы иметь панель-врезку (inlet) или другую панель с рекомендованными терминами, к которым вы можете осуществить привязку, – предполагается, что страницы на вики-сайте всегда связаны друг с другом, когда связь предоставляет пояснение или дополнительную информацию для некоторого термина или по какой-то теме. Наконец, естественная связь

страниц обеспечивает плодородную почву для поиска с применением усложненных запросов (глава 2), кластеризации (глава 4) и других аналитических методик.

1.3.5. Сайты общего доступа к медиафайлам

YouTube – фирменная марка веб-сайтов, обеспечивающих общий доступ к медиафайлам, но и другие веб-сайты, такие как RapidShare (*http://www.rapidshare.com*) и MegaUpload (*http://www.megaupload.com/*), не страдают от недостатка посетителей. Уникальной характеристикой этих сайтов является то, что большая часть их контента представлена в двоичном формате, – это видео- или аудиофайлы. В большинстве случаев размер наименьшей информационной единицы на этих сайтах превышает размеры, характерные для сайтов-агрегаторов, обрабатывающих текстовые данные; только лишь объем данных, которые необходимо обработать, составляет одну из самых больших проблем в контексте накопления интеллекта.

Кроме того, две из наиболее трудных (а также наиболее интересных с точки зрения бизнеса) задач интеллектуальных приложений тесно связаны с обработкой двоичных данных. Эти две задачи – распознавание речи и распознавание образов. Такие фирмы, как Clearspring (*http://www.clearspring.com/*) и ScanScout (*http://www.scanscout.com/*), работая совместно, позволяют рекламодателю шире распространить свой бренд и информировать о нем более широкую аудиторию. Фирма ScanScout предоставляет рекламодателям сведения о распределении по сайтам их виджетов и о взаимодействии с ними; список таких сайтов включает MySpace, Facebook, Google и Yahoo, а всего их насчитывается больше 25!

Ту же модель, которую мы описали в предыдущих разделах, можно обнаружить и на этих сайтах. У нас есть агрегированный контент; как правило, мы хотим иметь контент, распределенный по категориям; и нам нужны алгоритмы, которые могут способствовать извлечению смысла из этого контента. Хотелось бы, чтобы наши двоичные файлы были распределены по категориям исходя из определенных нами тем – Autos & Vehicles (Автомобили и транспортные средства), Education (Образование), Entertainment (Развлечения), Politics (Политика) и так далее (рис. 1.5).

Аналогично другим случаям интеллектуальных приложений, эти категории могут иметь иерархическую структуру. Например, категория Autos & Vehicles может быть далее разбита на подкатегории, такие как Sedan (Седан), Trucks (Грузовики), Luxury (Люкс), SUV (Внедорожники) и так далее.

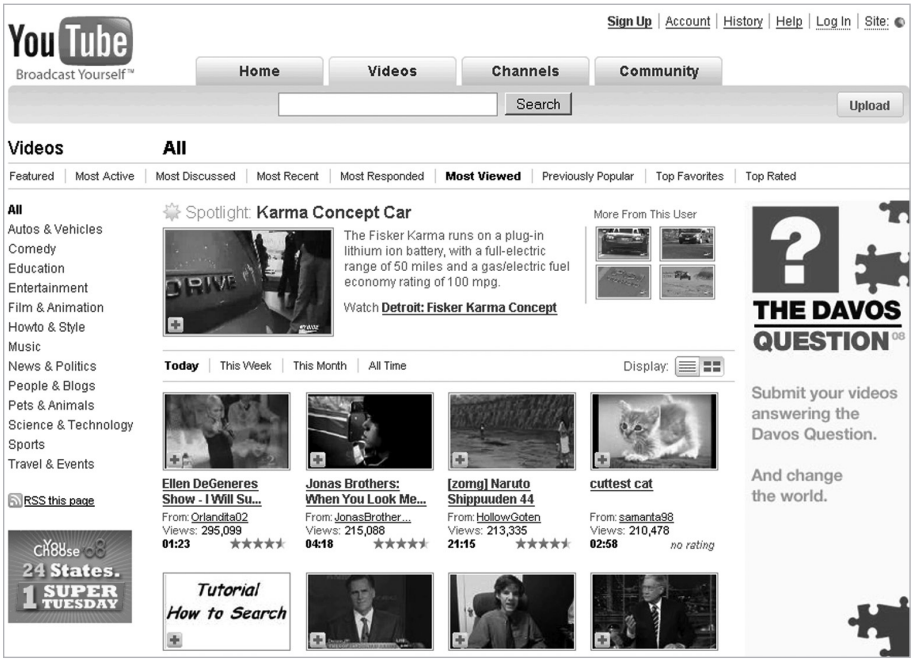

Рис. 1.5. *Категории видеозаписей на веб-сайте YouTube. Структура ссылок для классификации контента показана на панели слева*

1.3.6. Онлайн-игры

Захватывающие онлайн-игры с множеством игроков обладают всеми составляющими, необходимыми для того, чтобы встроить в игру интеллект. У них хватает агрегированного контента и ссылочных структур, которые отражают правила, и они, естественно, могут использовать алгоритмы, рассматриваемые нами в этой книге, чтобы ввести в игру новые уровни сложности. Персонажи, за которых играет компьютер, могут усваивать данные, вводимые игроками-людьми, так что игра станет более занимательной для человека.

Игры в сети – очень интересная область для применения интеллектуальных методик, и она может стать ключевым элементом для дифференциации конкурентов, поскольку с учетом сложности игр и инноваций возрастает уровень доступных для игры вычислительных мощностей и ожиданий игроков-людей. Методики, которые мы рассматриваем в главах 4, 5 и 6, как и большая часть материала приложений, находят прямое применение в онлайн-играх.

1.4. Как встроить интеллект в мое приложение?

Мы привели много причин для встраивания интеллекта в ваше приложение. Мы также рассмотрели ряд областей, где интеллектуальное поведение вашего программного обеспечения может радикально улучшить практические и стоимостные результаты, которые ваше приложение обеспечивает пользователям. На этой стадии естественным становится вопрос: «Как встроить интеллект в мое приложение?»

Вся эта книга – введение в проектирование и реализацию интеллектуальных компонентов, но чтобы использовать ее с наибольшей пользой, вам также следует обеспечить две предпосылки создания интеллектуального приложения.

Первая предпосылка – это ревизия функциональности вашего приложения. Что с ним делают пользователи? Как ваше приложение повышает потребительскую ценность или ценность бизнеса? Мы приводим несколько конкретных вопросов, относящихся в основном к алгоритмам, которые будут прорабатываться в оставшейся части этой книги. Важность вопросов зависит от того, что делает ваше приложение. Тем не менее эти конкретные вопросы должны помочь вам выявить те области, где интеллектуальный компонент добавил бы больше всего ценности в ваше приложение.

Вторая предпосылка касается данных. Для каждого приложения данные являются или внутренними (доступны непосредственно в приложении), или внешними, по отношению к этому приложению. Прежде всего, проанализируйте внутренние данные. Возможно, у вас есть все, что нужно, в таком случае вы готовы к действиям. И наоборот, возможно, вам необходимо организовать технологический процесс или добавить другие средства для получения от пользователей каких-то дополнительных данных. Например, вы хотите добавить на свои страницы элемент пользовательского интерфейса для классификации по принципу «пять звезд», с тем чтобы можно было создать механизм выработки рекомендаций, основываясь на классификации, которую обеспечат пользователи.

Или вам, к примеру, захотелось или понадобилось получать дополнительные данные из внешних источников. Есть много вариантов достижения этой цели. Мы не можем рассмотреть здесь все возможности, но определим четыре крупные категории, достаточно надежные технологически и популярные. Чтобы получить необходимые дополнительные сведения о конкретном выбранном методе, вам придется заглянуть в соответствующую литературу.

1.4.1. Анализ функциональности и данных

Вам следовало бы начать с выявления ряда вариантов использования (use cases), которые выиграли бы от интеллектуального поведения. Эти

варианты, разумеется, отличались бы от приложения к приложению, но вы можете выявить такие случаи, ответив на несколько очень простых вопросов:

- Обслуживает ли приложение контент, собранный из разных источников?
- Есть ли в приложении технологические процессы, реализованные с помощью «мастеров» (wizard)?
- Имеет ли приложение дело с текстом на естественном языке?
- Включает ли приложение создание отчетов какого-либо рода?
- Имеет ли приложение дело с географическими объектами, такими как карты?
- Предоставляет ли приложение функциональные возможности поиска?
- Используют ли ваши пользователи контент совместно?
- Важно ли для приложения обнаружение мошенничества?
- Важна ли для приложения верификация идентичности?
- Принимает ли приложение решения автоматически, основываясь на правилах?

Этот список, разумеется, не полон, но он указывает на имеющиеся возможности. Если на любой из этих вопросов дается ответ «да», ваше приложение может существенно выиграть от методик, которые мы будем рассматривать в оставшейся части этой книги.

Рассмотрим стандартный вариант использования поиска по данным вымышленного приложения. Почти все приложения предоставляют своим пользователям возможность осуществить поиск по сайту. Пусть, скажем, наше воображаемое приложение позволяет своим пользователям приобретать разные товары по каталогу. Пользователи могут выполнять поиск товаров, которые они хотят купить. Обычно такая функциональность реализуется с помощью прямого запроса на языке SQL, который обеспечит извлечение всех товарных позиций, соответствующих описанию товара. Это хорошо, но наш сервер базы данных не учитывает тот факт, что запрос может быть выполнен конкретным пользователем, о котором нам, скорее всего, многое известно в контексте его поиска. Мы наверняка сможем повысить эффективность взаимодействия пользователя с приложением, если реализуем методы ранжирования, описанные в главе 2, или методы выработки рекомендаций, описанные в главе 3.

1.4.2. Получение дополнительных данных из Интернета

Часто вам будет достаточно собственных данных для построения интеллекта, это важно и ценно для вашего приложения. Но в некоторых

случаях обеспечение интеллекта в приложении может потребовать доступа к внешней информации. На рис. 1.6 показана страница гибридного веб-сайта HousingMaps (*http://www.housingmaps.com*), который позволяет пользователям просматривать дома, доступные в некоторой географической точке, получая список домов с сайта объявлений Craigslist (*http://www.craigslist.com*) и карты от службы Google maps (*http://code.google.com/apis/maps/index.html*).

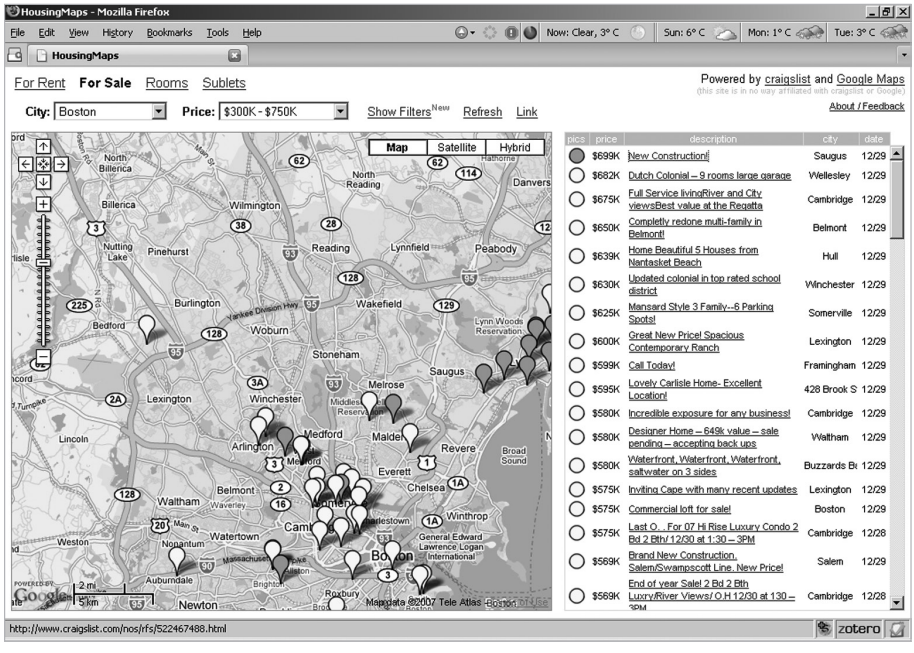

Рис. 1.6. *Список доступных домов, полученный с сайта Craigslist, в сочетании с картами местности, предоставленными службой Google maps (источник: http://www.housingmaps.com)*

Аналогичным образом, новостной сайт мог бы связать новостное сообщение с картой той местности, к которой относится данная новость. Возможность получить карту местности уже является усовершенствованием для любого приложения. Конечно, от этого ваше приложение не станет интеллектуальным, если только вы не сделаете что-то интеллектуальное с той информацией, которую получаете в результате обработки карты.

Географические карты – хороший пример получения внешней информации, но куда больший объем информации доступен в части Интернета, не связанной с картами. Давайте рассмотрим технологии, которые предоставляют такую информацию.

Обход контента и анализ экранных данных

Поисковые роботы (crawlers), их еще называют *пауками* (spiders), – это компьютерные программы, которые могут перемещаться по Интернету и скачивать общедоступный контент. Обычно поисковый робот будет посещать URL-адреса из некоторого списка, пытаясь пройти дальше по ссылкам и дойти до каждого пункта назначения. Этот процесс, который может повторяться многократно, обычно называют *глубиной обхода* (depth of crawling). Посетив страницу, поисковый робот локально сохраняет ее контент для последующей обработки. Вы можете собрать массу данных таким образом, но быстро столкнетесь с проблемами их хранения или с нарушением авторских прав. Будьте осторожны и дисциплинированны, осуществляя обход контента. В главе 2 мы представляем нашу собственную реализацию поискового веб-робота. Мы также включили в эту книгу приложение, где дан общий обзор обхода контента Интернета, кратко описан наш собственный поисковый веб-робот, а также несколько реализаций с открытым исходным кодом.

Анализ экранных данных (screen scraping) относится к извлечению информации, которую содержат HTML-страницы. Это незамысловатое, но трудоемкое занятие. Допустим, вы хотите создать поисковую систему исключительно для поиска места, где можно поесть (такую как *http://www.foodiebytes.com*). Извлечение информации о меню с веб-страницы каждого ресторана – одна из ваших первоочередных задач. Применение рассматриваемых в этой книге методик может оказаться полезным для анализа экранных данных. В случае системы поиска ресторанов вы можете оценить, насколько хорош какой-то ресторан, опираясь на оценки тех, кто там отобедали. В некоторых случаях может быть доступна шкала оценок, но в большинстве случаев эти отзывы – обычный текст на естественном языке. Чтение отзывов, одного за другим, и соответствующая классификация ресторанов, очевидно, не являются масштабируемым бизнес-решением. В процессе анализа экранных данных можно использовать интеллектуальные методики, способные помочь вам автоматически рассортировать отзывы по категориям, а также оценить класс ресторанов. Примером является система поиска ресторанов Boorah (*http://www.boorah.com*).

RSS-каналы

Синдикация (syndication) веб-сайтов – еще один способ получения внешних данных, избавляющий вашего поискового робота от повторного посещения веб-сайтов. Как правило, синдицированный контент в большей степени приспособлен для машинной обработки, нежели обычные веб-страницы, потому что эта информация хорошо структурирована. Есть три распространенных формата таких каналов: RSS 1.0, RSS 2.0 и Atom.

Формат RDF Site Summary, RSS 1.0, как подсказывает его название, является потомком модели представления данных Resource Description Framework[1] (RDF) и основан на том представлении, что информация в Интернете может использоваться как человеком, так и компьютером. Однако если человек, как правило, способен понимать семантику контента (смысл слов или фраз в контексте), то компьютеру сделать это нелегко. Модель RDF была разработана для того, чтобы способствовать семантической интерпретации Интернета. Вы можете воспользоваться ею для извлечения полезных данных и метаданных в ваших целях. Спецификацию RSS 1.0 можно найти по адресу *http://web.resource.org/rss/1.0/*.

Формат Really Simple Syndication, RSS 2.0 базируется на технологии Rich Site Summary 0.91 фирмы Netscape, – здесь, мягко говоря, имеется существенная перегрузка значениями сокращения RSS, – и исходно этот формат предназначался для снижения уровня сложности форматов, основанных на модели RDF. Этот формат использует специальный язык синдикации с представлением его в виде простого XML-формата, который не требует использования пространства имен XML или обращения непосредственно к модели RDF. Сегодня почти все главные сайты обеспечивают каналы RSS 2.0; обычно эти каналы бесплатны для некоммерческого использования – для частных лиц и некоммерческих организаций. На посвященном RSS-каналам сайте фирмы Yahoo (*http://developer.yahoo.com/rss*) есть множество ресурсов, позволяющих постепенно ознакомиться с этой темой. Доступ к спецификации RSS 2.0 и другой сопутствующей информации можно получить по адресу *http://cyber.law.harvard.edu/rss*.

Наконец, вы можете использовать синдикацию на базе формата Atom. Наличие ряда проблем, связанных с применением формата RSS 2.0, привело к разработке стандарта Рабочей группы инженеров Интернета (Internet Engineering Task Force, IETF), изложенного в документе RFC 4287 (*http://tools.ietf.org/html/rfc4287*). Формат Atom не базируется на модели RDF; он не так гибок, как формат RSS 1.0, и не так легок, как формат RSS 2.0. По сути, это был компромисс между возможностями существующих стандартов и ограничением, связанным с необходимостью обеспечить максимальную обратную совместимость с другими форматами синдикации. Тем не менее формат Atom, подобно формату RSS 2.0, получил широкое распространение. Большинство крупных веб-агрегаторов (таких как Yahoo! и Google) предлагают новостные каналы этих двух форматов. Подробнее о формате синдикации Atom можно прочесть на веб-сайте IBM Developer Works (*http://www.ibm.com/developerworks/xml/standards/x-atomspec.html*).

[1] *http://www.w3.org/RDF*