УДК 004.4’2

М.М. Глибовець, О.В. Конюшенко

РОЗРОБКА АГЕНТНОЇ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ ВІРТУАЛЬНОГО УНІВЕРСИТЕТУ ЗАСОБАМИ JADE

В роботі запропоновано модель агентної рекомендаційної системи (РС), що функціонуватиме в рамках віртуального університету та описано її мультиагенту реалізацію засобами JADE. Обгрунтовано вибір типу РС (item-based collaborative filtering systems), алгоритмів, що використовуються на різних кроках генерування рекомендацій. Розписані ролі та поведінка агентів.

Вступ

За останній час інтерес до алгоритмів, що лежать в основі рекомендаційних систем невпинно зростає. Відносно прості алгоритми та підходи, які донедавна справлялися з генеруванням рекомендацій, поступово вичерпують себе, вимагаючи від розробників постійного вдосконалення та пошуку альтернатив [1]. Зі зростанням кількості користувачів Інтернету та доступного контенту, сьогодні як ніколи потрібні якісні алгоритми генерування рекомендацій – адже ми живемо в еру інформаційного перевантаження.

«Інформаційне перевантаження» – термін, популяризований Алвіном Тоффлером у його книзі «Майбутній шок» [2]. У ній йдеться про труднощі, що виникають у процесі прийняття рішень і розуміння проблем людиною за умов надзвичайно великого обсягу інформації. Хоча сам термін виник ще до появи Інтернету, він якнайкраще описує сьогодення та найближче майбутнє.

Світ рухається до глобалізації і все більше людей отримують доступ до Інтернету, перетворюючись у його активних користувачів. Це призводить до збільшення різноманітного контенту в мережі Інтернет. Лоуренс Лессіг пояснює це «читай-пиши» («read-write») природою Інтернету [3]. Існує навіть думка, що ми надто залежні від подібного методу отримання інформації [4]. Вона ґрунтується на тому, що, маючи доступ до надзвичайно великого обсягу інформації, ми не завжди задумуємось про її правдивість, ризикуючи покластися на хибну інформацію.

Тому розробники програмного забезпечення мають прагнути надавати користувачам якомога кращий сервіс.

У 2009 році популярний американський інтернет-сервіс Netflix оголосив конкурс з призовим фондом в 1 мільйон доларів. Учасники повинні були підвищити точність рекомендацій клієнтам сервісу більш ніж на 10 відсотків. За оцінкою Джона Рідла, який був одним з перших дослідників колаборативної фільтрації, у таких гігантів, як Amazon та Netflix – наразі, найкращі рекомендаційні системи. Втім, «всі прості рішення вже реалізовані» і з часом стає все важче покращити точність рекомендацій (передбачень). Навіть незначне покрашення може призвести до відчутного зростання прибутків компаній, що й змусило Netflix піти на зазначений крок [5].

Тому в найближчі роки актуальність та інтерес до потужних рекомендаційних систем лише зростатиме.

Коли ми говоримо про зростання контенту в мережі, окрему увагу варто приділити навчальним матеріалам. У навчанні – як і інших аспектах свого розвитку – людина пройшла довгий шлях: від посиденьок у вогнища до обладнаних за останнім словом техніки лабораторій і аудиторій. Тоді як початкова освіта проникає навіть у найвіддаленіші куточки планети, на передовій цієї галузі – інтернет-портали найвідоміших університетів світу, що пропонують користувачам як анотації і окремі матеріали більшості пропонованих предметі, так і деякі з них – повністю.

Все починалося трохи більше десяти років тому з таких проектів, як *MIT OpenCourseWare, AllLearn, Harvard@Home.* Цікаво, що важливим поштовхом до подальшого розвитку онлайн-навчання стала поява YouTube – сервісу, що надав необхідний інструментарій не тільки провідним університетам, але й звичайним користувачам, які готові були ділитися своїми знаннями. Деякі з подібних проектів – *Khan Academy, The New Boston* – стали настільки успішними, що принесли своїм засновникам не тільки славу, але й статок. Нині ми маємо змогу спостерігати за наступним етапом еволюції онлайн-навчання – порталами на зразок *coursera.org, edxonline.org* та *udacity.com* [5].

У цій статті опишемо розроблену та реалізовану нами агентну модель подібного помічника студента – рекомендаційної системи віртуального університету.

Дана рекомендаційна система покликана допомагати студентам з вибором курсів вільного вибору; вона розроблялася з метою імплементації у Віртуальний університет Національного університету «Києво-Могилянська академія».

Ми опишемо побудову рекомендаційної системи, що належить до класу item-based collaborative filtering systems. Це – відносно новий та прогресивний підхід до створення рекомендацій. Зокрема, використання подібних систем дозволяє боротися з проблемою розрідженості матриць та зробити систему легко масштабованою. Ми також обґрунтовуємо вибір мультиагентної системи для реалізації рекомендаційної системи.

Онлайн-навчання

Онлайн-університети, швидше за все, назавжди змінять наші уявлення про освіту; нині вони надають повноцінні курси – з лекціями, завданнями, «дедлайнами», іспитами та сертифікатами про закінчення; попри труднощі періоду спроб та помилок, вони вже перетворилися у грізних конкурентів «традиційних» університетських відповідників. Сотні тисяч людей записуються на ці курси – і хоча більшість відсіюється ще до здачі іспиту, загальна кількість «студентів», що отримують сертифікати різних рівнів, вражає. Успіх та стрімкий розвиток онлайн-навчання змушує бачити в ньому майбутнє освіти.

Себастіан Тхурн, професор Стенфордського університету та Google Fellow, а також один із співзасновників онлайн-університету Udacity, каже, що через 50 років лише 10 інституцій у всьому світі пропонуватимуть освіту. В червні 2011 Тхурн став співзасновником KnowLabs, вклавши в компанію $300000 власних грошей – в сподіванні, що його дітище в перспективі стане однією з таких інституцій. Можливо, проект KnowLabs – Udacity, що в 2012 році запустив 6 курсів (від CS101 Building a Search Engine до «екзотичного» CS373 Programming a Robotic Car) – дійсно чекає така доля [6].

Зрештою, Тхурн як, можливо, ніхто розуміє, що ми стоїмо на порозі революційних змін у навчанні. Коли KnowLabs запускала свій перший курс, в компанії розраховували, що на нього запишеться максимум 2000 людей. Тхурн написав листа в Асоціацію розвитку штучного інтелекту (Association for the Advancement of Artificial Intelligence) в сподіванні зацікавити її членів. На ранок у них було 5000 користувачів. Через кілька днів – 10000. Два тижні потому – 58000. В середині серпня в The New York Times з'явилася стаття про курс; незабаром була взята планка в 100000 слухачів. Сайт, що був створений з розрахунку на цифру в десять разів меншу, не витримував навантаження. Один з високопосадовців компанії напівжартома сказав, що саме тоді він перестав спати [6].

Безумовно, остаточні цифри виглядають не так вражаюче на тлі того безумства, що мало місце на початку курсу, проте це лише омана. Так, 137000 слухачів не закінчило курс, «лише» 20000 дійшли до середини. Проте як для першого подібного експерименту, це був успіх.

Отож, так чи інакше, освіта поступово перебирається на простори Інтернету, що не тільки знищує географічні рамки, але й надає їй свій потужний інструментарій. Тому не дивно, що все більше і більше навчальних закладів по всьому

світу всерйоз переймаються своєю онлайн-присутністю – зокрема, головному та допоміжним сайтам. Якщо раніше сайт сприймався в першу чергу як обличчя університету та інформаційний портал для абітурієнтів і медіа, сьогодні акцент робиться на застосуванні різноманітних додатків з метою розширення можливостей безпосередньо студентів цього навчального закладу, звичайних та «віртуальних».

Україна все ще на відстані кількох років від онлайн-курсів у відкритому доступі, хоча ця ідея активно обговорюється – зокрема, і в закладі Києво-Могилянської академії. Втім, вже можна говорити про певний прогрес: у нас успішно функціонує система електронного навчання та близький до запуску віртуальний університет. Це значить, що Україна також близька до того, щоб стати частиною глобального освітнього простору.

Рекомендаційні системи

В англійській мові використовують таке поняття, як «educated guess»; так називають здогадку, припущення, зроблене на основі досвіду чи теоретичних знань. Цікаво, що подібні припущення можуть робити не тільки студенти, які навчаються засобами онлайн-університетів, але й самі університети. Для цього їм достатньо використовувати так звані рекомендаційні системи.

Рекомендаційні системи були створені для допомоги в подоланні інформаційного перевантаження. Правильне використання подібної системи позитивно впливає як на користувацький досвід, так і на роботу системи загалом.

Двома базовими елементами будь-якої рекомендаційної системи є користувач (інколи його називають «покупцем») та предмет інтересу («продукт»). Відповідно, рекомендаційна система отримує від користувача оцінку деяких предметів інтересу (в даному випадку – вибіркових курсів), а потім видає йому прогнозовану оцінку (educated guess!) інших курсів. Формат даних, які надає користувач системі, залежить від використаних алгоритмів, проте загалом буває трьох видів: рейтинги (наприклад, від 1 до 5 «зірочок» для фільмів або 1 чи 0 – залежно від того, купив користувач продукт чи ні); демографічна інформація (вік, стать користувача тощо); дані вподобаних користувачем продуктів. Висновком рекомендаційної системи може бути передбачення чи рекомендація. Передбачення – числове значення, що показує прогнозовану оцінку користувача продукту; рекомендація – список продуктів, які система розцінює як такі, що найбільше сподобаються користувачу [1].

Рекомендаційна система ставить на меті видати користувачу припущення стосовно нових чи неоцінених ним продуктів на основі введених ним даних. Нехай m – кількість користувачів: $U = \{u_1, u_2, \ldots, u_m\}$, n – кількість предметів інтересу: $I = \{i_1, i_2, \ldots, i_n\}$. Кожному користувачеві u_1 відповідає список L_{ui} продуктів, які користувач вже так чи інакше оцінив (список може бути порожнім, проте він не повинен містити в собі оцінки всіх продуктів). Всі наявні оцінки від усіх користувачів створюють матрицю $m * n$. Залежно від алгоритму, що використовується, відбуваються операції з рядками (окремий рядок становлять оцінки кожного користувача) чи стовпчиками (окремий стовпчик становлять оцінки кожного продукту) матриці; користувачеві повертається підмножина прогнозованих рейтингів неоцінених ним продуктів або відсортований список неоцінених продуктів для подальшого ознайомлення – рекомендація системи [7].

То куди ж рухатимуться рекомендаційні системи?

По-перше, попри постійне розширення Інтернету, рекомендаційні системи вийдуть за його рамки – наприклад, в телебачення (якщо воно не перебереться до того часу в Інтернет). Хороші рекомендаційні системи можуть змінити навіть прилади у вашому домі. Праска, що рекомендує режим прасування? Холодильник з рекомендаціями здорової їжі?

По-друге, як зазначається у [8], змінюються контекст рекомендацій та контекст для рекомендацій. Ми наперед не знаємо в якому місці шукатимемо які речі,

а системи порівнюватимуть нас з іншими людьми та друзями. Ми будемо шукати рекомендацій для груп людей.

Зрештою, музика та інші розваги більше не виступатимуть єдиними «продуктами», що підлягають рекомендаціям – дещо несподівано, проте соціальні мережі на зразок LinkedIn зрештою зможуть стати повноцінними помічниками HR-спеціалістам: радити не продукти, а людей. В нещодавньому інтерв'ю Wired Рейд Хоффман, засновник LinkedIn, бідкався, що люди досі не використовують потужність цієї мережі на повну [9]. Все ще не розкритий до кінця потенціал соціальних мереж разом з потужними алгоритмами, що використовуються для генерації рекомендацій, які суттєво вплинуть на процес пошуку працівників.

Агентна рекомендаційна система

Нагадаємо зазначені дві основні задачі – забезпечення відповідності рекомендаційної системи вимогам якості (зокрема: якісні рекомендації, масштабованість, забезпечення транзитивності в околі користувачів, шляхи вирішення проблем розрідженості, «першого рейтингу» та «незвичного користувача»), а також – побудова відповідної моделі агентної системи.

Розглянемо основні вимоги до рекомендаційних систем детальніше згідно [7].

Перша з них – якісні рекомендації. Для забезпечення якісних рекомендацій, система має мінімізувати кількість хибнопозитивних результатів у вихідних даних. Хибнопозитивними вважатимемо такі результати, які система відносить до вартих рекомендації, але які не подобаються користувачеві: у нашому випадку – якийсь з рекомендованих вибіркових курсів. Очевидно, що велика кількість хибнопозитивних результатів неприйнятна для рекомендаційної системи [10].

Наступна вимога – масштабованість. Алгоритми, що лежать в основі рекомендаційної системи, мають ефективно опрацьовувати поточну кількість «покупців» та «продуктів», що схильна до зростання [11].

Рекомендаційні системи також мають забезпечувати транзитивність у певному околі користувачів (якщо два користувачі А та В мають високий коефіцієнт кореляції, то третій користувач С, який має високий коефіцієнт кореляції з А, також мусить мати високий коефіцієнт кореляції з В). Попри очевидність подібного підходу, ця вимога часто ігнорується рекомендаційними системами, адже алгоритми, якими вони керуються, враховують кількість «продуктів», відмічених відразу обома користувачами. Для забезпечення якісних результатів розробнику необхідно забезпечити транзитивність в околах користувачів [7].

Розробник також мусить вирішити кілька специфічних проблем. Серед них – розрідженість, «першого рейтингу» та «незвичний користувач». Більшість алгоритмів рекомендаційних систем працюють з матрицями «покупець-продукт» та околами користувачів у них. Розрідженість матриць виникає внаслідок великої різниці між загальною кількістю «продуктів» та тими з них, які були якось відмічені «покупцями». Розрідженість призводить до утруднення встановлення потрібних околів користувачів [12]. Проблема «першого рейтингу» – яку ще називають проблемою «холодного старту» – необхідність наявності хоча б одного рейтингу у «продукту» для його рекомендації системою [13]. Проблема «незвичного користувача» – також відома, як проблема «сірої овечки» – присутність окремої групи користувачів, чиї смаки не дозволяють однозначно віднести їх до певного околу, що (переважно негативно) впливає на те, наскільки якісними будуть для них отримані рекомендації системи [7].

Процес створення рекомендацій

Традиційно, для створення рекомендацій використовуються системи, що можна віднести до двох класів: memory-based collaborative filtering algo rithms та model-based collaborative filtering algorithms.

Важливою відмінністю memory-based collaborative filtering algorithms є те,

що для створення рекомендацій вони використовують всю матрицю «покупець-продукт» (МПП). За допомогою статистичних інструментів ці системи знаходять множину користувачів, що називається околом, смаки яких збігаються зі смаком обраного користувача (у випадку з предметами вільного вибору це значить, що обидва користувачі в минулому обирали однакові предмети). Після цього, на основі знайденого околу та за допомогою конкретних алгоритмів генеруються рекомендації. Перевагою memory-based collaborative filtering algorithms є відносно якісні рекомендації та відносна простота імплементації, а серед недоліків – відносно повільна швидкість роботи, що є наслідком використання всієї матриці для генерування рекомендацій [12].

На відміну від memory-based collaborative filtering algorithms, model-based collaborative filtering algorithms генерують рекомендації, виходячи з моделі виставлення користувачем оцінок, яку вони створюють у процесі. Алгоритми, що використовуються при такому підході, відносять до класу ймовірнісних; вони розглядають процес колаборативного фільтрування як підрахунок очікуваного рейтингу (оцінки) предмету, виходячи з попередніх оцінок користувача. За побудову моделі оцінювання користувача відповідають алгоритми машинного навчання, такі як Bayesian network (баєсівські мережі, створює ймовірнісну модель оцінювання), clustering (ділить користувачів на групи (кластери) за смаками, після чого використовує умовну ймовірність для генерування рекомендацій) тощо [12].

Подібні підходи були дуже успішними в минулому, проте з часом окреслилися певні труднощі у їх використанні.

По-перше, зі зростанням кількості контенту виникла проблема розрідженості МПП. Враховуючи вражаючі обсяги різноманітних продуктів (книжок, фільмів тощо) окремий користувач зазвичай міг надати оцінку зовсім малій їх частці. Це, в свою чергу, ускладнювало пошук найближчих сусідів і, як наслідок, негативно впливало на якісь рекомендацій.

По-друге, зростання контенту та кількості користувачів також вказало на проблеми з масштабованістю подібних систем. Пошук сусідів вимагає серйозних підрахунків, відповідно, більші МПП вимагали потужніших алгоритмів.

Тому є сенс у використанні альтернативних підходів. Одним з них є, зокрема, використання мультиагентних систем з інтелектуальними агентами (зокрема, фільтрботами). Втім, як зазначено в [12], такий підхід сам по собі не є достатнім для того, щоб нівелювати описані вище проблеми.

Рекомендаційна система, описана в цій роботі, відноситься до класу систем колаборaційного фільтрування з акцентом на «продукті» (Item-Based Collaborative Filtering Systems). Можна виділити три кроки процесу створення рекомендацій: репрезентація, формування околів та генерування рекомендацій [1,7].

На рис. 1 зображено процес створення рекомендацій (передбачень) [12].

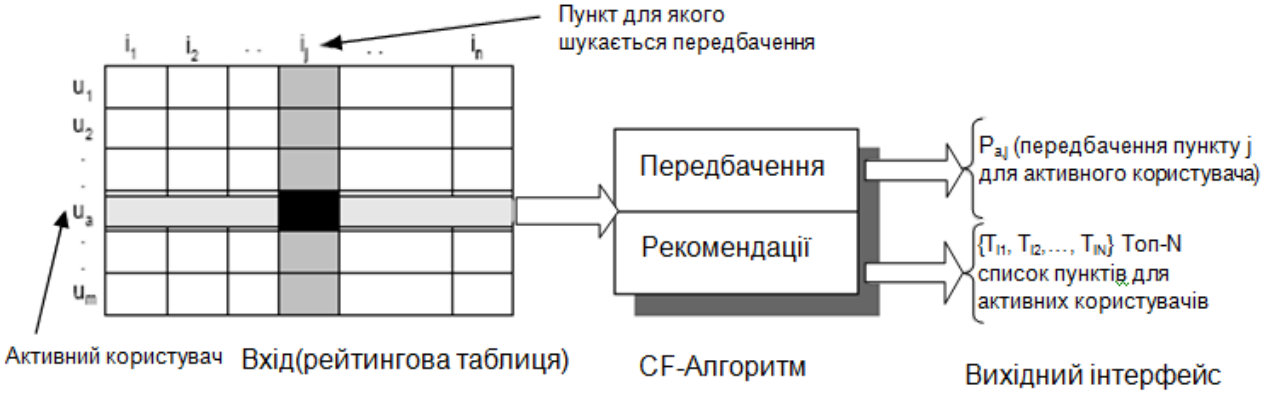

Рис.1. Процес колаборaційного фільтрування

Перший крок – репрезентація рекомендаційної системи. Ми працюємо з даними у вигляді МПП. Для *m* покупців та *n* продуктів ця матриця матиме розмірність *x*m*, при цьому ми не вимагаємо від жодного «покупця» оцінити всі можливі «продукти» (через що виникає розрідженість, про яку згадувалося вище). Під оцінкою ми розуміємо вибір користувачем певного курсу: «обрав» та «не обрав» (відповідно, 1 та 0).

Для боротьби з розрідженістю ми використаємо фільтрботів – агентів, задачею яких є автоматичне виставлення рейтингів у МПП. Рекомендаційна система сприймає фільтрботів як звичайних користувачів, втім, на відміну від останніх, фільтрботи лише оцінюють «продукти» та не потребують рекомендацій. Фільтроботи також є хорошим помічником у випадку так званого «холодного старту», коли недостатньо «покупців» оцінили різноманітні «продукти».

Для підвищення ефективності фільтрботів ми повинні активно їх використовувати при появі нових «продуктів» – у нашому випадку, нових предметів. Змушувати конкретного фільтрбота оцінити новий предмет ми будемо із врахуванням так званої ієрархії курсів.

Ієрархія курсів дозволяє формувати списки конкретних предметів, які обиратиме студент на кожному курсі. Якщо предмет А, що доступний для вибору на третьому курсі, вимагає від студента прослуханих предметів В та С на другому, ми можемо скласти відповідну ієрархію. Використання подібних ланцюжків курсів допомагає нам як при формуванні специфічних фільтрботів, так і на етапі формування околів. Незважаючи на те, що формування подібних ланцюжків вимагатиме безпосереднього втручання адміністратора системи, воно дає змогу покращити рекомендації (передбачення) системи.

Таким чином, використання ланцюжків предметів вільного вибору є частиною інтелектуальної потужності фільтрботів, що використовуються в нашій рекомендаційній системі. Ця ідея є розвиненням класу так званих «жанрових» фільтрботів, які використовуються у системах, що рекомендують користувачам фільми за їх смаками [7].

Другий крок – формування околів. Цей крок ще називають кроком *визначення схожості продуктів*. Ми виходимо зі схожості (відмінності) смаків пар користувачів.

У нашому випадку ми розрізняємо їх за обраними предметами. Якщо користувач обрав предмет, у відповідного елемента МПП буде значення 1, в протилежному випадку – 0; таким чином, отримуємо бінарну рейтингову схему. Щоб визначити схожість між двома предметами вільного вибору, i_j та i_k i_k, підрахуємо ймовірність вибору предмету i_j за умови, що предмет i_k був обраний раніше (тобто, так звану умовну ймовірність). Такий підхід називається умовною ймовірнісною схожістю [7]. Формула:

$$sim_{jk} = cond_{jk} = \frac{Freq(jk)}{Freq(k)}, \qquad (1)$$

де: $cond$ – умовна ймовірність, $Freq(jk)$ – кількість користувачів, що обрали обидва предмети, $Freq(k)$ – кількість користувачів, що обрали предмет k.

Варто зазначити, що зазвичай

$$cond(j|k) != cond(k|j),$$

тому ми ризикуємо отримати асиметричні реляції. Крім того, кожен предмет матиме високу умовну ймовірність щодо популярних предметів: тобто, наш коефіцієнт буде високим лише тому, що один з предметів вибирають часто, а не тому, що користувачі обирають їх разом [14]. Для вирішення цієї проблеми скористаємось запропонованими у [15] методами.

Спершу зазначимо, що один з способів вирішення цих проблем – ділення: ми просто ділимо коефіцієнт схожості $sim(j|k)$ на $Freq(j)$, де $Freq(j)$ – кількість користувачів, що обрали предмет j. Ділення також можна замінити множенням на коефіцієнт (подібний підхід «виріс» з інформаційного пошуку) – $\log_2(P(j))$.

Продуктивність рекомендаційних систем різко знижується з масштабуван-

ням системи, тому у [15] пропонується видозмінена формула для підрахунку коефіцієнта:

$$sim_{jk} = \frac{Freq(jk)}{Freq(k) * (Freq(j))^{\alpha}}$$

при цьому, у випадку $\alpha = 0$ формула набуває вигляду (1), а при $\alpha = 1$: $(1)/Freq(j)$.

Єдиний недолік останнього варіанту – те, що вона не розрізняє користувачів, які обрали багато предметів, та тих, що обрали всього кілька; проте для створення рекомендацій вибір користувачів з невеликим набором обраних предметів часто може бути важливішим. Тому варто нормалізувати кожен рядок МПП та вираховувати коефіцієнт як

$$sim_{jk} = \frac{\sum_{\forall i:\, r_{i,j}>0} r_{i,k}}{Freq(j) * (Freq(k))^{\alpha}}. \qquad (2)$$

Єдина відмінність між покращеним алгоритмом (2) та оригіналом – те, що ми використовуємо суму відповідних ненульових значень k-ої колонки МПП, а не кількість користувачів, що обрали певний предмет. Завдяки нормалізації ми отримуємо рядки «одиничної довжини», і користувачі, що обрали багато предметів, матимуть менший вплив на рекомендації [15].

Після підрахунку коефіцієнтів схожості для всіх предметів, мусимо обрати n предметів, коефіцієнти схожості яких з предметом i_j, для якого ми формуємо передбачення, найвищі. Цей набір утворить *окіл предмету* i_j.

Третій крок – генерування рекомендацій (передбачень). Для цього скористаємось стандартним підходом – зваженою сумою. Система генерує передбачення щодо певного предмету i_j підсумовуючи оцінки (у нашому випадку – одиниці та нулі) користувача u_a предметів у околі i_j; ці рейтинги зважуються коефіцієнтами схожості:

$$pr_{aj} = \frac{\sum_{k=1}^{l} sim_{jk} * r_{ak}}{\sum_{k=1}^{l} \left| sim_{ak} \right|},$$

де l – кількість предметів в околі.

Алгоритми, описані вище та застосовані в нашій рекомендаційній системі, належать до класу item-based collaborative filtering. Перевагами цього алгоритму є відносно хороша якість рекомендацій (передбачень). Алгоритм його відносно легко реалізувати, а використання всієї бази даних у процесі генерування рекомендації жодним чином не обмежує оновлення самої бази даних. Разом з цим, використання всієї бази даних дещо негативно впливає на швидкість роботи; інші недоліки цього алгоритму нівельовані нашими нововведеннями, які були описані вище.

Використання мультиагентної системи для рекомендаційної системи виправдано з точки зору ефективного розподілу задач та синтезу їх рішень (важливі проблеми з точки зору керування інформацією). Тут стає у нагоді розподіленість, природна для агентів. Інтереси студентів теж змінюються в часі, навчальні матеріали і методики також. Але ж і агентам притаманно навчатися та адаптуватися до навколишнього середовища. По-третє, студенти мають різні рівні підготовки, кожен з них особистість, тому методологія навчання мусить адаптуватися до індивідуальності студента. Спілкування, обговорення спільних тем й інтересів, може бути підтримано здатністю агентів до переговорів і наявністю спільної агентної мови, базованої на онтологіях. Нарешті, студенти можуть реєструватися в кількох курсах одночасно. Тому так важлива координація занять, теж природна для агентів.

В роботі [16] серед переваг поєднання інтелектуальних агентів з навчальним простором вказуються: швидка інтеграція нових користувачів та їх залучення до спільноти; ефективне використання контенту та спільних знань; поліпшення якості супроводу, асинхронної комунікації між користувачами та доступу до контенту.

Зрештою, варто також врахувати «традиційні» переваги мультиагентних систем. При розподілі обов'язків контроль та відповідальність за дії розподілені між різними агентами; відповідно, якщо

один з них вийде з ладу, система не припинить функціонувати. Завдяки модульності мультиагентну систему можна легко нарощувати та видозмінювати, адже для цього не потрібно змінювати код всієї програми. Системи зі змінними у часі параметрами можуть бути представлені сукупністю агентів.

Реалізація рекомендаційної системи засобами JADE

Сучасні мультиагентні системи використовують розподілений тип взаємодії. В колаборативному середовищі кожен агент володіє певними знаннями та спроможний ділитися ними з іншими агентами. В контексті об'єктно-орієнтованого програмування агент представляє собою поєднання набору функцій та інтерфейсу; останній використовується для комунікації між агентами. Втім, агенти також мають зовсім інший рівень автономії порівняно з програмами, запущеними на віддалених вузлах чи в мережі – завдяки механізму генерування та послідовного виконання власної мети (агенти, що мають цей механізм, називаються інтелектуальними агентами).

В розробці використовується широкий спектр мов програмування: від універсальних (Java) до мов програмування агентів (Goal); також варто враховувати окремі платформи (Jason, CLAIM, JACK тощо). Проте, можливо, найпопулярнішою агентною платформою є JADE – Java Agent DEvelopment Framework.

JADE – фреймворк з відкритим кодом, написаний на Java. Він спрощує розробку мультиагентної системи, виступаючи технологією проміжного рівня, що відповідає всім вимогам FIPA, та графічним інструментарієм, що підтримує «дебаггінг» та різні фази розробки. Завдяки тому, що агентів можна передавати з однієї машини на іншу прямо в процесі роботи, доступна зміна конфігурації «на льоту» через графічний інтерфейс. Використання агентних систем на JADE також дозволяє не турбуватися про відмінності між операційними системами на різних машинах [17]. Завдяки комунікаційній архітектурі JADE, процес обміну повідомленнями між агентами – гнучкий та ефективний: JADE створює чергу та керує потоком ACL-повідомлень (з налаштуваннями приватності). Доступ агентів до черги відбувається шляхом поєднання кількох доступних режимів роботи: блокування, голосування, перерви та співставлення (з еталоном).

Для створення агента в середовищі JADE ми повинні створити клас, що розширює базовий клас *jade.core.agent.* Метод setup() цього класу ініціалізує агента. При цьому всі дії агента (так звана поведінка агента) описуються в об'єктах класів, що називаються behaviours та розширюють клас *jade.core.behaviours.Behaviour* [18]. Графічний інтерфейс платформи, що імплементує RMA агента, дозволяє моніторити та адмініструвати платформу, а також тестувати її. Дозволяється запускати кілька RMA-агентів одночасно на одній платформі. RMA-агенти, по суті, є екземплярами класу *jade.tools.rma.rma.*

В інструментарії JADE є чимало корисних інструментів. *Dummy Agent* дозволяє відслідковувати обмін повідомленнями між агентами (відправка, отримання повідомлень, збереження у файл). Інтегроване застосування *Sniffer Agent* корисне для виправлення помилок в поведінці агентів (агент перехоплює повідомлення інших). *SocketProxyAgent* призначений для встановлення зв'язку з віддаленими клієнтами і підтримує до 50 паралельних підключень. *DF (Directory Facilitator) agent* відповідає за сервіс так званих "жовтих сторінок" ("yellow pages"), через який інші агенти роблять запити на послуги або відповідають на них.

Популярність JADE пояснюється потужністю, гнучкістю та відносною простотою цієї платформи. Саме тому вона була обрана для реалізації рекомендаційної системи.

Агентна модель системи

Цей розділ присвячено опису моделі агентної системи. Спершу, визначимось з різноманітними агентами, що існуватимуть в рамках рекомендаційної системи, вказавши їх назви та поведінку (набір дій, які вони здатні виконувати) (таблиця).

Таблиця

Назва агента	*Поведінка*
Student	Агент-студент. Може давати оцінку (1 чи 0) предметам, виходячи з того, прослухав він їх чи ні
Filterbot	Агент-фільтрбот. Активний на кроці репрезентації та (за певних умов) формування околів. Задачею фільтрботів є автоматичне виставлення рейтингів у МПП. Для рекомендаційної системи вони – аналог звичайних користувачів, втім, на відміну від останніх, не потребують рекомендацій, вони лише оцінюють за певними алгоритмами. «Побічний», але важливий ефект від дії фільтрботів – пониження розрідженості МПП; з цією ж метою ми змушуємо фільтрботів моніторити появу нових курсів та оцінювати їх. Фільтрботи також використовують ланцюжки предметів (ієрархію) – що перетворює їх у варіацію так званих «жанрових» фільтр ботів
Mayor	Агент-мер. Активний на кроці формування околів. Найцікавіший з агентів, адже в його обов'язки входить найнасиченіший етап створення рекомендацій. Щоб визначити схожість між двома предметами вільного вибору, i_j та i_k, підрахуємо ймовірність вибору предмету i_j за умови, що предмет i_k був обраний раніше (тобто, так звану умовну ймовірність). Крім того, завдяки удосконаленням наш коефіцієнт буде високим лише тому, що один з предметів вибирають часто, а не тому, що користувачі обирають їх разом; ми враховуємо те, скільки предметів користувач вже обрав (часто думка користувачів з меншим набором предметів важить більше), а масштабування не повинне стати проблемою. Код цього агента приведений далі
Committee	Агент-член комітету. Активний на етапах репрезентації та генерування рекомендацій. Генерує передбачення стосовно певного предмету i_j підсумовуючи оцінки (у нашому випадку – одиниці та нулі) користувача u_a предметів у околі i_j; ці рейтинги зважуються коефіцієнтами схожості. Відповідає за ієрархію предметів (саме до нього звертаються фільтрботи) та підрахунок кредитів

Складемо діаграму послідовності взаємодії агентів всередині системи. Діаграма послідовності відображає взаємодії об'єктів, впорядковані за часом: зокрема, послідовність відправлених повідомлень (рис. 2).

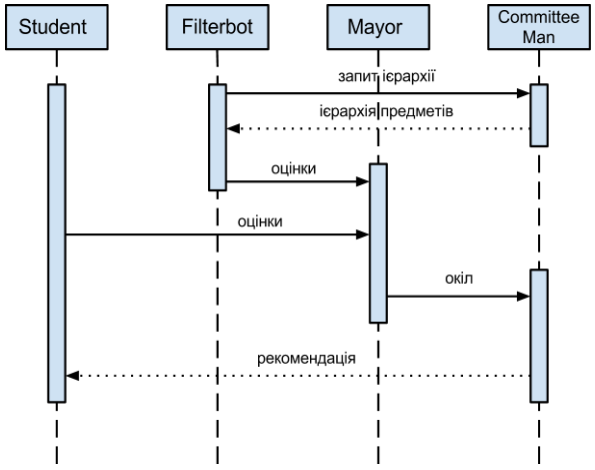

Рис. 2. Діаграма послідовності, що відображає взаємодії об'єктів

Як видно з діаграми, спочатку фільтрбот посилає запит агенту Commitee Man і отримує назад ієрархію предметів. Після цього оцінки фільтрботів та агентів Student формують МПП. На основі цієї матриці агент Mayor формує окіл для користувача та передає його агенту Commitee Man, який відповідає за генерацію рекомендацій. Після цього агент Commitee Man передає згенеровану рекомендацію (передбачення) агенту Student (це, звісно, передбачає звернення спудея до рекомендаційної системи на самому початку).

Приклад коду агента: MayorAgent

В цьому розділі наведений лістинг (з коментарями) одного з агентів – *MayorAgent* . Це – один з найцікавіших у реалізації агентів.

```
public class MayorAgent extends Agent {
protected void setup(){
  addBehaviour(new CyclicBehaviour(this) {
    @Override
    public void action() {
      ACLMessage msg = receive();
      if(msg != null) {
        // TODO Agent logic
        Entry entry = null;
        try {
             entry = (Entry) msg.getContentObject();
            if (entry == null)
               throw new RuntimeException("Internal error");
                // TODO internal exception class
             if (entry.getContent() != null) {
              ACLMessage reply = msg.createReply();
             String subject = msg.getContent();
             String[] recommendedCouses = Recommendations.getRecommendations(subject);

reply.setContentObject(recommendedCourses);
              send(reply);
           } else {
             // TODO implement
           }
           //
        } catch (UnreadableException e) {
          // TODO Auto-generated catch block
        } catch (IOException e) {
          // TODO Auto-generated catch block
        }
      }
      else {
          block();
      }
    }
  });
 }
}

package recommendations;

class ValueComparator implements Comparator {
  Map base;
  public ValueComparator(Map base) {
        this.base = base;
  }
  public int compare(Object a, Object b) {
      if((Double)base.get(a) > (Double)base.get(b)) {
              return 1;
      }       else       if((Double)base.get(a)       == (Double)base.get(b)) {
              return 0;
     } else {
              return -1;
      }
   }
}

public class Recommendations {
// Альфа – в знаменнику формули (2)
const double alpha = 0.5;
// Скільки предметів має бути в "околі"
const int subjectsToRecommend = 5;
// Клас для роботи з базою даних
DB db = new DB();
// Визначає рівень схожості між двома предметами - формула (2)
static double getSimilarity(Subject s1, Subject s2) {
        double sim = 0;
        // Для кожного студента, який слухає другий курс
        for (Student student : s2.students) {
        // Якщо він слухає і перший також
          if (s1.students.Contains(student)) sim++;
        }
        // Ділимо на кількість слухачів кожного з цих предметів
        return sim / ((double)s1.students.size() * Math.pow((double)s2.students.size(), alpha));
}
// Повертає "окіл" для даного предмета
public static String[] getRecommendations(String subject) {
        // Беремо з бази всі предмети
        ArrayList<Subject> subjects = db.getAllSubjects();
        // Створюємо мапу, в якій ключами будуть предмети, а значеннями -
        // рівні схожості (sim) з даним предметом (subject)
        Map<Subject,double> map = new HashMap<Subject,double>();
        // Проходимо усі предмети з бази
        for (Subject s: subjects) {
          // Якщо це наш предмет - пропускаємо
          if (s.name.equals(subject.name)) continue;
              // Визначаємо схожість
              double sim = getSimilarity(subject, s);
              // Передаємо в мапу
              map.put(s, sim);
        }
// Сортування
ValueComparator bvc =  new ValueComparator(map);
TreeMap<String,Double> sortedMap = new TreeMap(bvc);
sortedMap.putAll(map);
// Тепер в sortedMap - усі наші предмети відсортовані
// за спаданням схожості з даним
// Ініціалізуємо масив з 5-ти предметів для рекомендації (окіл)
  String[] result = new String[subjectsToRecommend];
  int i = 0;
  for (String sName : sortedMap.keySet()) {
        // Вихід, якщо достатньо
```

```
            if (i >= subjectsToRecommend) break;
            result[i++] = sName;
        }
        return result;
    }
}
```

Кінець лістингу.

Висновки

У цій роботі запропоновано модель агентної рекомендаційної системи віртуального університету, що покликана допомагати спудеям у виборі предметів вільного вибору. Пояснено вибір алгоритмів, використаних на різних кроках генерування рекомендацій (передбачень), та мультиагентного підходу. Розписані ролі та поведінка різних агентів. Внесені та пояснені зміни, що можуть якісно вплинути на роботу рекомендаційної системи.

Запропонована модель готова до імплементації у віртуальному університеті – проекті Національного університету "Києво-Могилянська академія".

Існують шляхи подальшого покращення рекомендацій (передбачень) системи. Приміром, можна ефективніше використовувати фільтрботів: порівнювати між собою їх оцінки; використовувати інші типи фільтрботів (не лише варіацію «жанрових»); використовувати ваги та генерувати рекомендації із застосуванням лінійних комбінації рекомендацій фільтрботів. Також можна застосувати принцип природнього відбору, за якого неефективні фільтрботи будуть знищуватися, а на зміну їм приходитимуть варіації їх ефективніших відповідників.

Варто зазначити, що, попри (виправдане) використання нами умовної ймовірнісної схожості на етапі формування околів, існують також інші підходи: CVS (Cosine/Vector Similarity), Adjusted Cosine Similarity тощо. Їх використання буде виправданим у випадку появи повноцінних рейтингів (користувачів конкретним предметам).

1. *Глибовець А.М.* Агенти для рекомендацій в колаборативних середовищах // Наукові праці Миколаївського державного гуманітарного університету ім. Петра Могили комплексу "Києво-Могилянська академія". – 2010. – Вип. 121, Т. 134: Комп'ютерні технології. – С. 142–151.
2. *Тоффлер Э.* Шок будущего: Пер. с англ. / Э. Тоффлер. – М.: ООО "Издательство АСТ", 2002. – 557 с.
3. Lawrence Lessig The Read-Write Internet – Режим доступу: www.youtube.com/watch?v=7nBL8eClYuU – 2006.
4. Михаил Грунин. "$1 000 000 за новый рекомендательный алгоритм". – Режим доступу: http://artpragmatica.ru/rs/?uid=1229 – Назва з екрана.
5. Harvard Magazine Online-Education Venture – Режим доступу: http://harvardmagazine.com/2012/05/harvard-mit-launch-edx-online-learning-venture – Назва з єкрана.
6. Steven Leckart. The Stanford Education Experiment Could Change Higher Learning Forever – Режим доступу: http://www.wired.com/wiredscience/2012/03/ff_aiclass// – Назва з екрана.
7. *Emmanouil Vozalis, Konstantinos G.* Margaritis Analysis of Recommender Systems. – Режим доступу: http://lsa-svd-application-for-analysis.googlecode.com/svn-histry/r72/trunk/LSA/Other/LsaToRead/hercma2003.pdf – 2003.
8. *MobBlog.* A Pitch on Future Recommender Systems – Режим доступу: http://mobblog.cs.ucl.ac.uk/2008/11/27/a-pitch-on-future-recommender-systems/ – Назва з екрана.
9. *David Rowan*. For LinkedIn Founder Reid Hoffman, Relationships Rule the World / David Rowan – Режим доступу: http://www.wired.com/epicenter/2012/03/ff_hoffman// – Назва з екрана.
10. *Nathaniel Good, J. Ben Schafer, Joseph A. Konstan, Al Borchers, Bardul M. Sarwar, Jon Herlocker and John T.* Riedl Combining collaborative filtering with personal agents for better recommendations AAAI '99/IAAI // Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence. – 1999. P. 439–446.
11. *Bardul M.* Sparsity, Scalability, and Distribution in Recommender Systems // Ph.D. thesis, University of Minnesota – Режим доступу: http://ssc.bibalex.org/viewer/detail.jsf;jsessionid=EF26CF691B01A7BE6DB04C97F585C52D?lid=E17DA62C3219B2B2CA69FF679AB36CF4&aterm=Search&page=1785&tid=ED69E7D0C1D173E0A63DD495DA59D02F&atype=&apage=1&id=. – 2001.

12. Badrul Sarwar, George Karypis, Joseph Konstan , John Riedl Item-based Collaborative Filtering Recommendation Algorithms // Proceedings of the 10-th international conference on World Wide Web, 1–5 May 2001, Hong Kong: ACM Press. – New York. – 2001. – P. 285–295.
13. *Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, and David M. Pennock.* Methods and metrics for cold-start recommendations // Proceedings of the 25-th annual international ACM SIGIR conference on Research and development in information retrieval, 11–15 August 2002, Finland: ACM Press. – New York. – 2002. – P. 253–260.
14. *Brendan Kitts, David Freed, and Martin Vrieze.* Cross-sell: A fast promotion-tunable customer-item recommendation method based on conditional independent probabilities // Proceedings of ACM SIGKDD International Conference, – 2000. – P. 437–446.
15. *George Karypis.* Evaluation of item-based top-n recommendation algorithms // Proceedings of the tenth international conference on Information and knowledge management, – 2001. – P. 247–254.
16. *Глибовець М.М., Сергієнко І.В., Гороховський С.С., Глибовець А.М.* Програмні засоби створення і супроводу розподіленого навчального середовища // Національний університет "Києво-Могилянської академії" – К.: НаУКМА: Аграр Медіа Груп, 2012. – 710 с.
17. *Глибовец Н.Н.* Использование JADE (Java Agent Development Environment) для разработки компьютерных систем поддержки дистанционного обучения агентного типа // Образовательные технологии и общество. – 2005. – Т. 8, № 3. – С. 325–345.
18. *JADE* Tutorial – Режим доступу: http://jade.tilab.com/doc/tutorials/JADEProgramming-Tutorial-for-beginners.pdf – Назва з екрана.

Одержано 04.07.2014

Про авторів:

Глибовець Микола Миколайович,

доктор фізико-математичних наук,
професор,
декан факультету інформатики
Національного університету
«Києво-Могилянська Академія»,

Конюшенко Олексій Володимирович,

аспірант.

Місце роботи авторів:

Національний університет
«Києво-Могилянська академія»
(НаУКМА),
04655, м. Київ,
вул. Г. Сковороди, 2.
Тел.: (044) 425 60 59.
E-mail: glib@ukma.kiev.ua.

Київський національний
університет імені Тараса Шевченка,
03680, м. Київ,
проспект Глушкова 4д.
Тел.: (095) 576 60 69.
E-mail: okonyushenko@gmail.com.