МОСКОВСКИЙ ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ
(ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ)
ФАКУЛЬТЕТ ОБЩЕЙ И ПРИКЛАДНОЙ ФИЗИКИ
КАФЕДРА СИСТЕМНОЙ ИНТЕГРАЦИИ И МЕНЕДЖМЕНТА

На правах рукописи

Соболев Михаил Сергеевич

МЕТОД ЛАТЕНТНОГО СЕМАНТИЧЕСКОГО АНАЛИЗА

магистерская диссертация

Руководитель доктор физ.-мат. наук,
профессор, член-корреспондент РАН Ю.А. Флеров

Рецензент кандидат филологических наук,
доцент В.В. Рыков

Москва 2007

Содержание

1 Введение

Интенсивное развитие систем электронного обучения (e-Learning), используемых как для дистанционного обучения, так и для поддержки очного учебного процесса ставит большое число сложных проблем, связанных с возможностью автоматической проверки ответов обучаемого. Одной из важных задач, стоящих перед системами, использующими электронные средства обучения, является организация автоматической или полуавтоматической проверки ответов, данных студентом в свободной форме (например, проверка соответствия содержания эссе заданной теме) [1]. Для реализации такой проверки необходим метод, позволяющий осуществлять смысловое сравнение отрывков текста.

В данной работе описан пример использования метода латентного семантического анализа (ЛСА) для выявления смысловых корреляций между отрывками текста.

1.1 Актуальность работы

Актуальность данной работы обуславливается следущим:

- В большинстве из широко распространенных компьютерных систем обучения при тестировании используются вопросы, основанные на прямом сравнении ответа с заранее заданным вариантом правильного ответа. Такие тесты подходят для проверки фактологических знаний и понимания концептуальных связей в предметной области, косвенной проверки практических навыков решения задач в определенной предметной области. При этом, однако, не доступны для оценивания аспекты знания, связанные со способностью тестируемого практически демонстрировать свои

знания и умения в рассуждениях, дискуссиях, ответах на вопросы собеседников [1].

- Практически невозможно проводить автоматическое тестирование творческих способностей студентов, например, в рамках таких специальностей как журналистика, литература и перевод. Однако существует потребность в системе полуавтоматической оценки, облегчающей труд проверяющего. Такая система может быть создана на основе метода ЛСА.

- На сегодняшний день не существует открытых решений, позволяющих использовать ЛСА для оценивания русскоязычных текстов.

1.2 Цели работы

Цели данной работы:

- Освоить математическую модель, лежащую в основе метода латентного семантического анализа;
- Провести анализ численных методов и выбрать алгоритм сингулярного разложения, наилучшим образом удовлетворяющий поставленной задаче;
- Спроектировать и разработать программное обеспечение (ПО), позволяющее использовать ЛСА;
- Изучить применимость разработанного ПО к англо- и русскоязычным текстам;
- Провести анализ существующих разработок в области метода латентного семантического анализа;

2 Метод ЛСА, обзор литературы, общее описание

Одним из перспективных методов, позволяющих получать данные о смысле приведенного текста, является метод латентного семантического анализа (ЛСА).

ЛСА позволяет выявлять значения слов с учетом контекста их использования путем обработки большого набора текстов. Принцип действия метода заключается в том, что сравнение множества всех контекстов, в которых слова или группы слов употребляются, и контекстов, в которых они не употребляется, позволяет сделать вывод о степени близости смысла этих слов или групп слов.

Впервые метод ЛСА был описан в работе [2] и затем развит в трудах Scott Deerwester, Susan Dumais, George Furnas и др. В настоящее время лидером в области применения ЛСА является компания Pearson Knowledge Technologies ([3]). Её коммерческие продукты позволяют убедиться в хорошей эффективности метода ЛСА. Однако конкретные алгоритмы реализации этого метода не опубликованы, поскольку представляют собой коммерческую тайну, поэтому представлялось важным на первом этапе работы разработать программное обеспечение, которое позволит на практике воспроизвести и проверить результаты работы алгоритма ЛСА.

Представления слова и абзаца с помощью метода ЛСА во многом моделируют восприятие текста человеком [4]. Например, с его помощью можно оценить эссе на соответствие теме или сопоставить смыслы отрывков текста.

Некоторые из этих результатов будут описаны ниже. Кроме того содержательные примеры работы метода ЛСА могут быть найдены в работах [2], [5].

ЛСА можно рассматривать в двух аспектах:

- как практический прием для получения примерных оценок контекстной связи слов в больших фрагментах по смыслу, либо оценок смысловых корреляций между словом и набором слов (в случае присутствия таких корреляций);
- как компьютерную модель получения и использования знаний человеком, читающим текст.

В качестве практического метода, характеризующего значение слова, ЛСА позволяет измерить корреляции типа «слово-слово», «слово-отрывок» и «отрывок-отрывок». Эти корреляции моделируют механизм мышления человека, сопоставляющего части текста по смыслу. Опыт показывает наличие связи между результатами работы метода ЛСА и человеческим восприятием. Важно отметить, что результаты, даваемые методом ЛСА, зависят не только от частотности использования слов в отрывках. Метод основывается на выявление более глубоких («латентных») связей и, таким образом, лучше моделирует человеческое восприятия текста чем простые методы, основанные на частотности употребления слов [6].

Следует отметить, что у метода ЛСА существуют некоторые ограничения. В нем не используется информация о порядке слов, и, следовательно, метод не учитывает синтаксические отношения, логику или морфологию. Несмотря на это, результаты метода достаточно достоверно отображают смысловые корреляции между словами и отрывками [2].

Существуют два основных отличия метода ЛСА от прочих статистических методов обработки текстов:

- в качестве исходных данных ЛСА использует частоту использования слов в отрывках текста, а не частоту совместного использования слов;

- метод собирает данные не о попарной совместной используемости слов, а об используемости множества слов в большом массиве отрывков.

Таким образом, метод рассматривает влияние выбора, а не порядка слов на смысл отрывка. Можно сказать, что ЛСА представляет значение слова как среднее значений отрывков, в которых оно встречается, а значение отрывка - как среднее значений всех слов, составляющих отрывок.

3 Математическая модель ЛСА

3.1 Матрица употребляемости

ЛСА использует матрицу, которая описывает употребляемость слов в отрывках. Пусть, для определенности, столбцы матрицы соответствуют документам, а строки – словам, встречающимся в документах. Элементы матрицы тогда представляют собой количество употреблений данного слова в данном отрывке. Такой подход стандартен для всех семантических моделей.

Итак, оригинальная матрица отражает связи между словами и отрывками. При анализе текстов на естественном языке возникает две основные проблемы – синонимия и полисемия:

- *Синонимия* подразумевает что одно и то же понятие может описываться в естественном языке разными комбинациями слов.
- При *полисемии* одно и то же слово или комбинация слов могут описывать разные понятия.

3.2 Понижение ранга

После получения матрицы употребляемости требуется понизить ее ранг (т.е аппроксимировать ее матрицей с меньшем рангом). Это необходимо делать по нескольким причинам:

- из-за высокого ранга исходной матрицы вычисления с ней часто невозможны;
- исходная матрица содержит «шумы» - например «случайное» попадание в отрывок слова, несоответствующего смыслу отрывка. Понижение ранга позволяет снизить значимость таких случаев – то есть отчасти избавиться от «шумов»;
- предполагается, что исходная матрица слишком «разрежена» - она учитывает только слова, появляющиеся в отрывке - но не слова, связанные с отрывком (через синонимию).

В результате преобразования размерность разложения отрывков по словам уменьшается, например:

{(машина), (грузовик), (растение)} → {(1.563*машина+0.661*грузовик),(растение)}

Операция уменьшает влияние синонимии, так как понижение ранга «сливает» размерности, связанные со словами, имеющими близкие значения. Также уменьшается влияние полисемии: в случае, если многозначное слово имеет «правильное значение», то его элемент «сливается» с матричными элементами слов с таким же значением. Если же слово употреблено в «неправильном» значении, то соответствующий элемент будет уменьшен или отброшен.

3.3 Преобразование ([7])

Пусть A –матрица, гле элемент (i,j) отображает употребление слова i в отрывке j. Матрица A будет иметь следующий вид:

$$\begin{matrix} & & \mathbf{d}_j \\ & & \downarrow \\ \mathbf{t}_i^T \rightarrow & \begin{bmatrix} x_{1,1} & \dots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \dots & x_{m,n} \end{bmatrix} \end{matrix}$$

Элемент $x_{i,j}$ представляет собой количество употреблений i-го слова в j-том отрывке. Каждая строка в этой матрице – это вектор, соответствующий слову и отражающий его употребляемость в каждом документе.

$$\mathbf{t}_i^T = \begin{bmatrix} x_{i,1} & \dots & x_{i,n} \end{bmatrix}$$

Аналогично, каждый столбец представляет из себя вектор, соответствующий отрывку и отражающий употребляемость слов в этом отрывке:

$$\mathbf{d}_j = \begin{bmatrix} x_{1,j} \\ \vdots \\ x_{m,j} \end{bmatrix}$$

Пусть $A \in \mathbb{C}^{m \times n}$. Тогда $A^* A \in \mathbb{C}^{m \times n}$ - эрмитова неотрицательно определенная матрица:

$$(A^* A)^* = A^* (A^*)^* = A^* A; \qquad xA^* Ax = (Ax, Ax) = |Ax|^2 \geq 0 \quad \forall x \in \mathbb{C}^n.$$

Поэтому все ее собственные значения неотрицательны.

Неотрицательные квадратные корни из собственных значений матрицы A^*A называются синглярными числами матрицы A. Сингулярные числа $w_i = w_i(A)$ принято нумеровать по невозрастанию:

$$w_1 \geq w_2 \geq \ldots \geq w_r > w_{r+1} = \ldots = 0 .$$

Будем считать, что A имеет r ненулевых сингулярных чисел.

Пусть $u_1,\ldots,u_n$ - ортонормированный базис собственных векоров матрицы A^*A такой, что

$$A^*A = \begin{cases} w_i^2 u_i, & 1 \leq i \leq r, \\ 0, & r+1 \leq i \leq n. \end{cases}$$

Положим $v_i = Au_i / w_i$, $1 \leq i \leq r$. Тогда $(v_i, v_j) = 0$ при $i \neq j$ и $(v_i, v_i) = 1$. Дополним систему $v_1,\ldots,v_r$ векторами $v_{r+1},\ldots,v_m$ до ортонормированного базиса в $\mathbb{C}^m$. Заметим также, что при $j \geq r+1$

$$A^*Au_j = 0 \Rightarrow u_j^* A^*Au_j = 0 \Rightarrow (Au_j)^*(Au_j) = 0 \Rightarrow \left|Au_j\right| = 0 \Rightarrow Au_j = 0 .$$

В итоге получаем

$$A[u_1,\ldots,u_n] = [v_1,\ldots,v_m] \begin{bmatrix} w_1 & & \\ & \ddots & \\ & & w_r \\ & & \end{bmatrix} \Rightarrow AU = VW ,$$

где $U = [u_1,\ldots,u_n]$ и $V = [v_1,\ldots,v_m]$ - унитарные матрицы, а W - диагональная прямоугольная матрица тех же размеров, что и матрица A.

Столбцы матриц U и V образуют сингулярные базисы матрицы A. Столбцы U называются правыми сингулярными векторами, а столбцы V - левыми сингулярными векторами матрицы A. Связь между сингулярными векторами и ненулевыми сингулярными числами устанавливается соотношениями

$$Au_i = w_i v_i , \quad A^*v_i = w_i u_i , \quad 1 \leq i \leq r .$$

Кроме того,

$$Au_i = 0, \ \ r+1 \leq i \leq n , \qquad A^*v_i = 0, \ \ r+1 \leq i \leq m .$$

Итак, мы доказали, что для любой матрицы $A \in \mathbb{C}^{m\times n}$ имеет место равенство

$$AU = VW. \qquad (*)$$

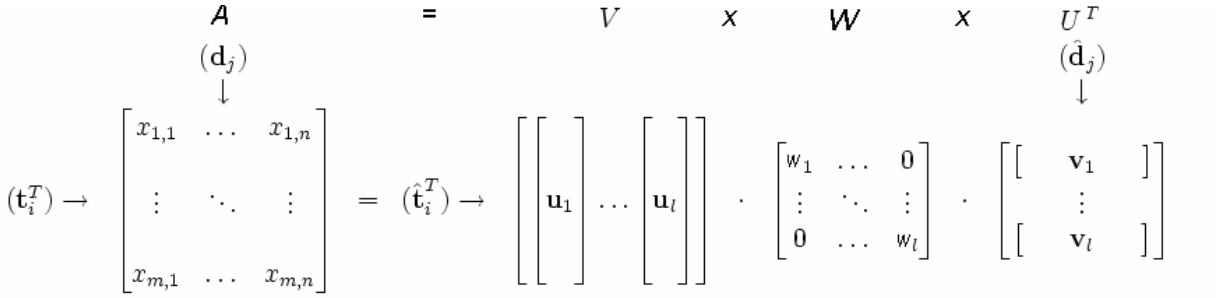
Для некоторых унитарных матриц $U \in \mathbb{C}^{n\times n}$, $V \in \mathbb{C}^{m\times m}$ и диагональной прямоугольной матрицы размеров $m\times n$ с числами $w_i \geq 0$ при $i = j$. Записав (*) в виде

$$A = VWU^*, \qquad (**)$$

получаем представление матрицы, называемое ее сингулярным разложением.

Если каким-то способом получено разложение (**) с унитарными матрицами U и V, то $A^*A = U(W^*W)U^*$. Поэтому если W - диагональная прямоугольная матрица с неотрицательными элементами, то ее ненулевые элементы определены однозначно.

Таким образом, разложение имеет следующий вид:



Числа $w_1,...,w_l$ называются сингулярными числами, а $u_1,...,u_l$ и $v_1,...,v_l$ правыми и левыми сингулярными векторами соответственно. Следует заметить, что единственная часть матрицы V, которая влияет на элементы вектора t_i - это её i-я строка. Обозначим этот вектор $\hat{t}_i$. Аналогично, на d_i влияет только j-ый столбец U^T, $\hat{d}_i$.

Если из всех сингулярных значений отобрать k наибольших, то мы получим аппроксимацию исходной матрицы матрицей ранга k.

Покажем, что из всех матриц полученная матрица будет наилучшей аппроксимацией.

Теорема 1. *Пусть матрица* $A \in \mathbb{C}^{m\times n}$ *задана сингулярным разложением вида*

$$A = \sum_{l=1}^{r} w_l v_l u_l^*$$

и условимся считать, что $w_{r+1} = 0$. *Пусть задано целое* $1 \le k \le r$. *Тогда*

$$\min_{\substack{rankB\le k \\ B\in\mathbb{C}^{m\times n}}} \|A-B\|_2 = w_{k+1} = \|A-A_k\|_2 \text{ , где } A_k = \sum_{l=1}^{k} w_l v_l u_l^*$$

Доказательство. Пусть $rankB \le k$. Тогда $\dim\ker B \ge n-k$. Рассмотрим линейную оболочку $L = L(u_1,...,u_{k+1})$, натянутую на старшие сингулярные векторы. По теореме Грассмана,

$$\dim(\ker B \cap L) = \dim\ker B + \dim L - \dim(\ker B + L) \ge (n-k)+(k+1)-n = 1.$$

Поэтому существует ненулевой вектор $z \in \ker B \cap L$. Будем считать что $\|z\|_2 = 1$.

Учитывая, что

$$z = \sum_{l=1}^{k+1} \alpha_l u_l, \quad \sum_{l=1}^{k+1} |\alpha_l|^2 = 1$$

находим

$$\|A-B\|_2 \geq \|(A-B)z\|_2 = \|Az\|_2 = \sqrt{\sum_{l=1}^{k+1} |\alpha_l|^2 w_l} \geq w_{k+1}$$

В то же время,

$$A - A_k = \sum_{l=k+1}^{r} w_l v_l u_l^* \Rightarrow \|A - A_k\|_2 = w_{k+1}.$$

Такая аппроксимация исходной матрицы матрицей меньшего ранга называется понижением ранга. Эта операция является основной в методе ЛСА, так как при ее проведении отбрасывается «избыточная информация», что, как будет показано ниже, позволяет достигнуть лучших результатов работы метода и сократить объем вычислений. Автоматический выбор нового ранга матрицы – открытая исследовательская проблема. В данной реализации метода ЛСА ранг матрицы после преобразования задается пользователем.

4 Численные методы сингулярного разложения

Один из первых алгоритмов, осуществляющих сингулярное разложение, был алгоритм Якоби, приводящий прямоугольную матрицу к диагональному виду при помощи последовательности элементарных вращений [8]. Хотя этот метод обладает интересным свойством - при правильной его реализации он позволяет очень точно находить все сингулярные значения матрицы, даже очень малые - низкая скорость его работы привела к тому, что он уступил место семейству алгоритмов, основанных на QR-итерации.

В основе наиболее популярных современных алгоритмов сингулярного разложения лежит приведение матрицы ортогональным

преобразованием к двухдиагональной форме (достаточно простая задача, решающаяся за конечное число операций) и последующая её диагонализация итеративным QR-алгоритмом. Разные алгоритмы обычно отличаются тем, как они осуществляют итерации QR-алгоритма. Первоначально широкое распространение получило семейство алгоритмов, основанных на алгоритме Голуба-Кахана-Рейнча [8]. Достоинствами этого метода являются его простота и компактность реализации. Однако этот алгоритм имеет ряд недостатков. Хотя обычно алгоритм успешно справляется с задачей, в некоторых сложных случаях его сходимость и точность оставляют желать лучшего.

В данной работе был использован алгоритм сингулярного разложения, исправляющей недостатки алгоритма Голуба-Кахана-Рейнча. Следует отметить такую важную особенность этого алгоритма, как повышенную точность нахождения малых сингулярных значений двухдиагональной матрицы. Использованный вариант QR-итерации находит все сингулярные значения с одинаковой абсолютной погрешностью, при этом самое большое сингулярное значение находится с точностью, близкой к машинной.

3.1 Описание алгоритма ([8])

Таким образом, разложение $A = VWU^*$ производится в два этапа. Сначала матрица A посредством двух конечных последовательностей преобразований Хаусхолдера где $x_k^T x_k = y_k^T y_k = 1$, приводится к верхней двухдиагональной форме следующего вида:

$$
{}_1AQ_1\ldots Q_{n-2} = \begin{pmatrix} q_1 & e_1 & & .. & \\ & q_2 & e_2 & .. & \\ .. & .. & .. & .. & .. \\ & & & .. & e_n \\ & & & .. & q_n \\ - & - & - & - & - \\ & & 0 & & \end{pmatrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \} (m-n)\times n \end{matrix} \equiv J_0
$$

Далее реализуется итерационный процесс приведения двухдиагональной матрицы J_0 к диагональной форме, так что имеет место следующая последовательность: $J_0 \to J_1 \to \ldots \to \Sigma$, где $J_{i+1} = S_i{}^T J_i T_i$, а S_i и T_i – диагональные матрицы.

Матрицы T_i выбираются так, чтобы последовательность матриц $M_i = J_i{}^T J_i$ сходилась к двухдиагональной матрице. Матрицы же S_i выбирают так, чтобы все J_i сохраняли двухдиагональную форму. Переход $J_i \to J_{i+1}$ осуществляется с помощью плоских вращений – преобразований Гивенса. Отсюда, $J_{i+1} = S_i T^{(n)^T} S_i{}^{(n-1)^T} \ldots S_2{}^T J_i T_i{}^{(2)} T_i{}^{(3)} \ldots T_i{}^{(n)}$, где

$$
S_i{}^{(k)} = \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \end{matrix} \begin{matrix} & & & & k-1 & k & & \\ & & & & \downarrow & \downarrow & & \\ \end{matrix} \begin{pmatrix} 1 & .. & 0 & 0 & 0 & .. & 0 \\ .. & .. & .. & .. & .. & .. & .. \\ 0 & .. & 1 & 0 & 0 & .. & 0 \\ 0 & .. & 0 & \cos\theta_k & -\sin\theta_k & .. & 0 \\ 0 & .. & 0 & \sin\theta_k & \cos\theta_k & .. & 0 \\ .. & .. & .. & .. & .. & .. & .. \\ 0 & .. & 0 & 0 & 0 & .. & 1 \end{pmatrix} \begin{matrix} \\ \\ \\ \leftarrow k-1 \\ \leftarrow k, \\ \\ \\ \end{matrix}
$$

а матрица $T_i{}^{(k)}$ вычисляется аналогично с заменой θ_k на φ_k.

Пусть начальный угол φ_2 произволен, однако следующие значения угла необходимо выбирать так, чтобы матрица J_{i+1} имела ту же форму, что и J_i.

Таким образом $T_i^{(2)}$ не аннулирует ни одного элемента матрицы, но добавляет элемент $\{J_i\}_{21}$; ${S_i}^{(2)^T}$ аннулирует $\{J_i\}_{21}$ но добавляет $\{J_i\}_{13}$; $T_i^{(3)}$ аннулирует $\{J_i\}_{13}$ но добавляет $\{J_i\}_{32}$ и т.д., наконец, ${S_i}^{(n)^T}$ аннулирует $\{J_i\}_{n,n-1}$ и ничего не добавляет.

Этот процесс часто называют процессом преследования. Так как $J_{i+1} = {S_i}^T J_i T_i,,$ то $M_{i+1} = {J_i}^T J_i = {T_i}^T M_i T_i,$ и M_{i+1} – трехдиагональная матрица, точно так же, как и M_i. Начальный угол φ_2 можно выбрать так, чтобы преобразование $M_i \to M_{i+1}$ было *QR*–преобразованием со сдвигом, равным *s*.

Обычный *QR*–алгоритм со сдвигом можно записать в следующем виде:

$M_i - sI = {T_i}^s R^s;$

$R^s {T_i}^s + sI = {M_{i+1}}^s,$

где $(T_i^s)^T T_i^s = I; R^s$ – верхняя треугольная матрица. Следовательно, ${M_{i+1}}^s = (T_i^s)^T M_i T_i^s$. Параметр сдвига *s* определяется собственным значением нижнего минора (размерности 2×2) матрицы M_i. При таком выборе параметра *s* метод обладает глобальной и почти всегда кубичной сходимостью.

5 Программная реализация метода ЛСА

Для программной реализаци ЛСА был выбран язык Java. Java был разработан в 1995 году компанией Sun Microsystems и с самого начала проектировался как объектно-ориентированный язык. В этом он выгодно отличается от C++, который, из соображений совместимости, сохраняет многие элементы процедурного языка C.

В качестве исходных данных используется файл, содержащий отрывки текста и ключевые слова. Данные в файле представлены в формате XML, что позволяет использовать при работе с файлом стандартные библиотеки. После обработки данного файла программа формирует иерархическую структуру из узлов(nodes), из которой, в свою очередь, извлекаются исходные данные для работы алгоритма LSA: набор отрывков текста, набор ключевых слов или словоформ и.т.д.

Таким образом, XML-файл служит промежуточной формой хранения обрабатываемых текстов, удобной для решения поставленной задачи. Например, в такой форме может быть представлена часть толкового словаря, для последующего измерения смысловых корреляций между словарными статьями.

Следующим шагом является формирование матрицы употребляемости: происходит подсчет количества употреблений словоформ в отрывках текста и отбрасываются строки и столбцы, содержащие только нулевые значения. При выполнении сингулярного разложения используется библиотека JAMA. Эта свободно распространяемая библиотека позволяет производить основные операции с матрицами (сложение, умножение, транспонирование, сингулярное разложение и.т.д.).После выполнения сингулярного разложения и понижения ранга (определяемый пользователем ранг матрицы задается во входном xml-файле) матрица анализируется на предмет корреляций.

В отдельную подпрограмму вынесен алгоритм расчета коэффициента корреляции Спирмена, использующегося при измерении корреляции между отрывками. Этот алгоритм приведен в Приложении.

Таким образом, результатом работы ЛСА в программе является корреляционная матрица, отражающая смысловую связь между отрывками. Возможно сохранение этой матрицы в файле, либо ее дальнейший анализ.

Таким анализом, например, может быть поиск отрывков, имеющих наибольшее значение корреляции с заданным отрывком. Листинг части программы, выполняющей обработку файла и ЛСА полученной матрицы, приведен в приложении.

Программная реализация метода ЛСА подтвердила его эффективность, однако, подтвердилась и основная проблема при практическом использованиии метода – большой объем вычислений при выполнении сингулярного разложения ([2]). Отчасти она может быть решена путем оптимизации самого метода, а отчасти – путем предварительных вычислений над исходными данными.

6 Пример работы метода ЛСА

Проиллюстрируем работу метода ЛСА. В качестве исходных данных используются несколько заглавий научных работ, причем часть из этих работ (c1-c5) посвящена информационным технологиям, а часть (m1 – m4) – математике. Покажем, что метод ЛСА позволяет установить смысловую корреляции между отрывками из группы.

c1	**Human** machine **interface** for ABC **computer** applications
c2	A **survey** of **user** opinion of **computer system response time**
c3	The **EPS user interface** management system
c4	**System** and **human system** engineering testing of EPS
c5	Relation of **user** perceived **response time** to error management

m1	The generation of random binary ordered **trees**
m2	The intersection **graph** of paths in **trees**
m3	**Graph minors**: Widths of **trees** and well-quasi-ordering
m4	**Graph minors**: A **survey**

Таблица 1. Исходные данные.

	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	1	0	0	1	0	0	0	0	0
interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
user	0	1	1	0	1	0	0	0	0
system	0	1	1	2	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
survey	0	1	0	0	0	0	0	0	1
trees	0	0	0	0	0	1	1	1	0
graph	0	0	0	0	0	0	1	1	1
minor	0	0	0	0	0	0	0	1	1

Таблица 2. Матрица употребляемости

Составим матрицу употребляемости слов в отрывках. Её элементы – количество употреблений данного слова в данном отрывке текста.

В качестве критерия корреляции отрывков мы будем использовать ранговую корреляцию Спирмена (способ ее вычисления описан в Приложении 1) между столбцами матрицы. Будем считать корреляцию превышающую 0.4 (зеленый цвет) индикатором того, что отрывки

находятся в одногй предметной области. Аналогично, корреляцию меньше 0.4 (красный цвет) будем считать показателем смысловой противоположности отрывков.

	c1	c2	c3	c4	c5	m1	m2	m3	m4
c1	**1**								
c2	**0.19**	**1**							
c3	**0.00**	**0.00**	**1**						
c4	**0.00**	**0.00**	**0.47**	**1**					
c5	**-0.33**	**0.58**	**0.00**	**-0.31**	**1**				
m1	**-0.17**	**-0.30**	**-0.21**	**-0.16**	**-0.17**	**1**			
m2	**-0.26**	**-0.45**	**-0.32**	**-0.24**	**-0.26**	**0.67**	**1**		
m3	**-0.33**	**-0.58**	**-0.41**	**-0.31**	**-0.33**	**0.52**	**0.77**	**1**	
m4	**-0.33**	**-0.19**	**-0.41**	**-0.31**	**-0.33**	**-0.17**	**0.26**	**0.56**	1

Таблица 3. Корреляции в исходной матрице.

Видны недостатки простого частотного метода – плохая корреляция между отрывками из одной предметной области. Для улучшения результатов требуется отбросить «несущественную информацию». Для этого подвергнем исходную матрицу сингулярному разложению и понизим её ранг.

	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	0.16	0.40	0.38	0.47	0.18	-0.05	-0.12	-0.16	-0.09
interface	0.14	0.37	0.33	0.40	0.16	-0.03	-0.07	-0.10	-0.04
computer	0.15	0.51	0.36	0.41	0.24	0.02	0.06	0.09	0.12
user	0.26	0.84	0.61	0.70	0.39	0.03	0.08	0.12	0.19
system	0.45	1.23	1.05	1.27	0.56	-0.07	-0.15	-0.21	-0.05
response	0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
time	0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
EPS	0.22	0.55	0.51	0.63	0.24	-0.07	-0.14	-0.20	-0.11
survey	0.10	0.53	0.23	0.21	0.27	0.14	0.31	0.44	0.42
trees	-0.06	0.23	-0.14	-0.27	0.14	0.24	0.55	0.77	0.66
graph	-0.06	0.34	-0.15	-0.30	0.20	0.31	0.69	0.98	0.85
minor	-0.04	0.25	-0.10	-0.21	0.15	0.22	0.50	0.71	0.62

Таблица 4. Преобразованная матрица употребляемости.

В полученной матрице оказался учтен контекст употребления. Вес слова survey в отрывке m4 уменьшен с 1 до 0.42, так как в текстах группы m оно употребляется только один раз. Вес слова trees увеличен с 0 до 0.66, так как оно употребляется во всех остальных отрывках группы m.

	c1	c2	c3	c4	c5	m1	m2	m3	m4
c1	1								
c2	0.91	1							
c3	1.00	0.91	1						
c4	1.00	0.88	1.00	1					
c5	0.85	0.99	0.85	0.81	1				
m1	-0.85	-0.56	-0.85	-0.88	-0.45	1			
m2	-0.85	-0.56	-0.85	-0.88	-0.44	1.00	1		
m3	-0.85	-0.56	-0.85	-0.88	-0.44	1.00	1.00	1	
m4	-0.81	-0.50	-0.81	-0.84	-0.37	1.00	1.00	1.00	1

Таблица 5. Корреляции в преобразованной матрице.

Как и ожидалось, в преобразованной матрице повысились коэффициенты корреляции между отрывками из одной группы, и понизились коэффициенты корреляции между отрывками из разных групп. Появилась четкая положительная корреляция между отрывками из одной предметной группы и отрецательная корреляция между отрывками из разных предметных групп. Это произошло из-за учета контекста употребления слов в тексте.

Работа метода ЛСА была также проверена на примерах русских тестов. Рассмотрим шесть отрывков текста (первые три отрывка относятся к астрономии, а остальные – к физике):

P1 Эклиптика — большой круг небесной сферы, по которому осуществляется видимое годовое движение Солнца по небесной сфере. Плоскость эклиптики пересекается с плоскостью небесного экватора под углом 23. Плоскость эклиптики является основной в эклиптической системе небесных координат.
P2 Полюсы мира. Ось мира пересекается с поверхностью небесной сферы в двух точках — северном полюсе мира и южном полюсе мира. Северным полюсом называется тот, со стороны которого движение небесной сферы происходит по часовой стрелке, если смотреть на сферу извне.
P3 Точки равноденствия. Эклиптика пересекается с небесным экватором в двух точках — точке весеннего равноденствия и точке осеннего равноденствия. Точкой весеннего равноденствия называется та, в которой Солнце, в результате своего годового движения, переходит из южного полушария небесной сферы в

северное. В точке осеннего равноденствия Солнце переходит из северного полушария небесной сферы в южное.
P4 Физический маятник — твёрдое тело, совершающее колебания в поле каких-либо сил относительно точки, не являющейся центром масс этого тела, или оси, не проходящей через центр масс этого тела.Одной из характеристик физического маятника является приведённая длина. Движение физического маятника описывается соответствующим уравнением.
P5 Степени свободы - Подавляющее большинство физических систем может находиться не в одном, а во многих состояниях, описываемых как непрерывными (например, координаты тела), так и дискретными (например, квантовые числа электрона в атоме) переменными. При математическом описании, N степеням свободы отвечают N независимых переменных, называемых обобщёнными координатами этого тела.
P6 Инерциа́льная систе́ма отсчёта (ИСО) — система отсчёта в которой справедлив закон инерции: тело, на которое не действуют внешние силы, находится в состоянии покоя или равномерного прямолинейного движения. Всякая система отсчёта, движущаяся относительно ИСО равномерно и прямолинейно, также является ИСО. Согласно принципу относительности, все ИСО равноправны, и все законы физики в них действуют одинаково.

Таблица 6. Исходные данные для русскоязычного примера..

В результате, как и в предыдущем примере, информативность корреляционной матрицы значительно повысилась:

	p1	p2	p3	p4	p5	p6
p1	1					
p2	0.221	1				
p3	0.231	0.470	1			
p4	-0.355	-0.233	-0.229	1		
p5	-0.039	-0.404	-0.443	0.069	1	
p6	-0.038	-0.262	-0.322	0.372	0.320	1

Корреляции до LSA

	p1	p2	p3	p4	p5	p6
p1	1					
p2	0.853	1				
p3	0.872	0.995	1			
p4	-0.329	-0.622	-0.590	1		
p5	-0.506	-0.756	-0.725	0.947	1	
p6	-0.478	-0.737	-0.706	0.964	0.997	1

Корреляции после LSA

Таблица 7.Результаты работы ЛСА.

7 Существующие разработки, использующие метод ЛСА

7.1 Кластеризация текста

Латентный семантический анализ применяется при решении часто встречающейся задачи структурирования и анализа отрывков текста. Для решения этой задачи требуется произвести разбиение текстовых массивов на систему (возможно иерархических) подмножеств, помеченных какими-то их смысловыми описателями (автоматическую кластеризацию).

В методах кластеризации, основывающихся на метрике близости, документ представляется в виде многомерного вектора. Подходы к формированию вектора, представляющего документ, могут существенно различаться. В простейшем случае каждый элемент вектора соответствует наличию в тексте одной из словоформ, встречающейся в рассматриваемом

наборе текстов. Основной проблемой методов, основанных на такой матрице, оказывается слишком большая размерность пространства слов, большая часть которых является избыточными и даже вредными. Например, при кластеризации текстов в данной предметной области термины, не относящиеся к этой области, могут маскировать сходство между документами. Сингулярное разложение, использующуеся в методе ЛСА, позволяет понизить размерность матрицы, уменьшить объем вычислений и избавиться от избыточных данных.

7.2 Латентное семантическое индексирование

Результат, выдаваемый обычной поисковой машиной, основан на непосредственном сравнении запроса и документа. Пользователю выдаются только те документы, в которых присутствует одно или несколько слов из запроса. Результат, полученный таким образом, часто бывает неточен – он может содержать «лишние» документы (не относящиеся к теме запроса пользователя) и не содержать тематических документов, в которых отсутствуют слова из запроса. Для получения лучших результатов необходимо учитывать смысловую близость запроса пользователя и выдаваемых документов.

Латентное семантическое индексирование(LSI) позволяет улучшить результат поиска. При отборе документов учитывается не только наличие в них слов из запроса, но и смысловая близость к запросу (измеренная с использованием метода ЛСА). Семантический поиск находит выражения, которые соответствуют поисковым запросам, пользователя и выдает результаты, которые близки им по смыслу. Страница с результатом поиска может не содержать тех слов, которые вы искали, но слова на странице будут близки к вашему поисковому запросу.

Этот подход используется в поисковой машине Google. Например, запрос «apple computer» будет интерпертирован как относящийся к компьютерной компании Apple, поэтому среди результатов будут документы, описывающие технологии этой компании (даже если эти документы не содержат слов «apple» или «computer»).

8 Заключение

Таким образом, поставленная цель достигнута полностью.

Результатами данной работы можно считать следующее:

- Освоена математическая модель, лежащая в основе метода латентного семантического анализа, а также изучен используемый в нём математический аппарат;
- Проведен анализ численных методов сингулярного разложения, в результате которого для данной реализации метода ЛСА был выбран модифицированный алгоритм Голуба-Кахана-Рейнча;
- Спроектировано и разработатно оригинальное ПО, практически реализующее метод латентного семантического анализа;
- Показана эффективность применения разработанного ПО для выявления смысловых корреляций между текстами для ряда примеров как на русском, так и на английском языке;
- Проведен анализ существующих разработок использующих метод ЛСА.

Приложение 1. Расчет коэффициента ранговой корреляции Спирмена ([9]).

1 Определить, какие два признака или две иерархии признаков будут участвовать в сопоставлении как переменные A и B.

2 Проранжировать значения переменной A, начисляя ранг 1 наименьшему значению. Занести ранги в первый столбец таблицы по порядку номеров испытуемых или признаков.

3 Проранжировать значения переменной B, в соответствии с теми же правилами. Занести ранги во второй столбец таблицы по порядку номеров испытуемых или признаков.

4 Подсчитать разности d между рангами A и B по каждой строке таблицы и занести в третий столбец таблицы.

5 Занести каждую разность в квадрат: d^2. Эти значения занести в четвертый столбец таблицы.

6 Подсчитать сумму d^2.

7 При наличии одинаковых рангов расчитать поправки:

$$T_a = \sum \left(a^3 - a\right)/12$$

$$T_b = \sum \left(b^3 - b\right)/12$$

где a - объем каждой группы одинаковых рангов в ранговом ряду A; b - объем каждой группы одинаковых рангов в ранговом ряду B.

8 Расчитать коэффициент ранговой корреляции r_s по формуле:

8.1 при отсутствии одинаковых рангов

$$r_s = 1 - 6 \cdot \frac{\sum d^2}{N \cdot \left(N^2 - 1\right)}$$

8.2 при наличии одинаковых рангов

$$r_s = \frac{1 - 6\frac{(\sum d_i^2 + T_a + T_b)}{N(N^2 - 1)}}{\sqrt{\left(1 - \frac{12}{N(N^2 - 1)} T_a\right)\left(1 - \frac{12}{N(N^2 - 1)} T_b\right)}}$$

где $sum(d^2)$ - сумма квадратов разностей между рангами; T_a и T_b - поправки на одинаковые ранги; N - количество испытуемых или признаков, участвовавших в ранжировании.

Приложение 2. Листинг программного алгоритма LSA.

```
import java.io.File;
import java.io.FileOutputStream;
import org.w3c.dom.Document;
import org.w3c.dom.NodeList;
public class LSA {
   static double T = 0;
  public static void main(String argv[]) {
      double in[][];
      try {
         DocumentBuilder builder = DocumentBuilderFactory.newInstance().newDocumentBuilder();
         Document doc = builder.parse(new File("src\\input.xml"));
         NodeList keywordsNodes = doc.getElementsByTagName("keyword");
         NodeList paragraphsNodes = doc.getElementsByTagName("paragraph");
```

```
String keywords[] = new String[keywordsNodes.getLength()];

String paragraphs[] = new String[paragraphsNodes.getLength()];

in = new double[keywordsNodes.getLength()][paragraphsNodes.getLength()];

int i = 0, j = 0;

for (i = 0; i < keywords.length; i++) {

   for (j = 0; j < paragraphs.length; j++) {

      keywords[i] = keywordsNodes.item(i).getFirstChild().getNodeValue().trim();

      paragraphs[j] = paragraphsNodes.item(j).getFirstChild().getNodeValue();

      in[i][j] = occurIn(paragraphs[j], keywords[i]);


   }

}


for (i = 0; i < keywords.length; i++)

{

   System.out.println("Key word "+i+":"+keywords[i]);

}

for (j = 0; j < paragraphs.length; j++)

{

   System.out.println("Paragraph "+j+":"+paragraphs[j]);

}

int rank = (new
Integer(doc.getElementsByTagName("rank").item(0).getFirstChild().getNodeValue())).intValue();

Matrix a = new Matrix(in);

Matrix b = spearman(a);

SingularValueDecomposition svd = new SingularValueDecomposition(a);

Matrix mask = svd.getS();
```

```
        for (int k = rank; k < mask.getRowDimension(); k++) {
            mask.set(k, k, 0);
        }
        Matrix c = svd.getU().times(mask).times(svd.getV().transpose());
        Matrix d = spearman(c);
        System.out.println("INITIAL MATRIX:");
        a.print(4, 2);
        System.out.println("INITIAL CORRELATIONS:");
        b.print(4, 2);
        System.out.println("MATRIX AFTER SVD (RANK=" + rank + "):");
        c.print(4, 2);
        System.out.println("RESULTING CORRELATIONS:");
        d.print(4, 2);
    } catch (Exception e) {
        e.printStackTrace();
    }


}
private static int occurIn(String _in, String _sample) {
    int result = 0;
    String in = _in.toLowerCase();
    String sample = _sample.toLowerCase();
    StringTokenizer tk = new StringTokenizer(in);
    while (tk.hasMoreTokens()) {
        String word = tk.nextToken();
        if (word.indexOf(sample) != -1) result++;
```

```
    }
    return result;
}
private static Matrix spearman(Matrix a) {
    double vals[][] = a.transpose().getArrayCopy();
    double result[][] = new double[vals.length][vals.length];
    for (int i = 0; i < vals.length; i++) {
        for (int j = 0; j < vals.length; j++) {
            result[i][j] = spearman(vals[i], vals[j]);
        }
    }
    return new Matrix(result);
}

private static double spearman(double a[], double b[]) {
    double x = 0;
    double Ta, Tb;
    computeRanks(a);
    Ta = T;
    computeRanks(b);
    Tb = T;
    int N = a.length;
    for (int i = 0; i < a.length; i++) {
        double d = a[i] - b[i];
        x += d * d;
    }
    double nom = 1 - 6 * (x + Ta + Tb) / (N * (N * N - 1));
```

```
    double denom = Math.sqrt((1 - 12 * Ta / (N * (N * N - 1))) * (1 - 12 * Tb / (N * (N * N - 1))));
    return nom / denom;
}

private static double[] computeRanks(double a[]) {
    int i = 0;
    int N = a.length;
    Map a_map = new LinkedHashMap();
    for (i = 0; i < N; i++) {
        a_map.put(new Integer(i), new Double(a[i]));
    }
    Object entries[] = a_map.entrySet().toArray();
    Arrays.sort(entries, new Comparator() {
        public int compare(Object a, Object b) {
            if (((Double) (((Map.Entry) a).getValue())).doubleValue() >
                    ((Double) (((Map.Entry) b).getValue())).doubleValue())
                return 1;
            if (((Double) (((Map.Entry) a).getValue())).doubleValue() ==
                    ((Double) (((Map.Entry) b).getValue())).doubleValue())
                return 0;
            if (((Double) (((Map.Entry) a).getValue())).doubleValue() <
                    ((Double) (((Map.Entry) b).getValue())).doubleValue())
                return -1;
            return 0;
        }
    });
    Map rank = new LinkedHashMap();
```

```
            for (i = 0; i < N; i++) {
                rank.put(((Map.Entry) entries[i]).getKey(), new Double(i + 1));
            }
            T = 0;
            int from_index = -1;
            for (i = 0; i < N - 1; i++) {
                if (from_index < 0) {
                    if (((Double) ((Map.Entry) entries[i]).getValue()).doubleValue() == ((Double) ((Map.Entry) entries[i + 1]).getValue()).doubleValue()) {
                        from_index = i;
                    }
                } else if ((i == N - 1) || (((Double) ((Map.Entry) entries[i]).getValue()).doubleValue() != ((Double) ((Map.Entry) entries[i + 1]).getValue()).doubleValue())) {
                    for (int k = from_index; k < (i + 1); k++) {
                        rank.put(((Map.Entry) entries[k]).getKey(), new Double(0.5 * (i + from_index) + 1));
                    }
                double t = i - from_index + 1;
                    T += (t * t * t - t);
                    from_index = -1;
                }
            }
            T = T / 12;
            for (i = 0; i < N; i++) {
                a[i] = ((Double) rank.get(new Integer(i))).doubleValue();
            }
            return a;
        }
    }
```

Литература

[1] А. В. Заболеева-Зотова, А. Ю. Пастухов, П. В. Сердюков, Н. А. Козлова, С. А. Чернов *Латентный семантический анализ: новые решения в Internet.* Информационные технологии, 2001 номер 6

[2] Landauer, T. K., Foltz, P., and Laham, D. 1998. *An Introduction to Latent Semantic Analysis.* DiscourseProcesses, 25: 259-284.

[3] *Веб-сайт Pearson Knowledge Technologies – http://www.k-a-t.com*

[4] Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K. and Harshman, R.A. 1990. *Indexing by Latent Semantic Analysis.* Journal of the American Society for Information Science, 41: 391-407.

[5] Foltz, P. W. 1996. *Latent Semantic Analysis for text-based research.* Behavior Research Methods, Instruments and Computers. 28, 197-202.

[6] Landauer, T. K., & Dumais, S. T. (1997). *A solution to Plato's problem: The Latent Semantic Analysis theory of the acquisition, induction, and representation of knowledge.* Psychological Review, 104(2), 211-240.

[7] Тыртышников Е.Е. *Курс линейной алгебры*. − М., 2004

[8] Дж. Голуб, Ч. Ван Лоун *Матричные вычисления.* − М., "Мир", 1999

[9] Афанасьев В. В. *Теория вероятностей в примерах и задачах.* − Ярославль, 1994