УДК 681.3.

АЛГОРИТМ ПОБУДОВИ ШЕСТИВИМІРНОГО ТЕНЗОРА ДЛЯ ЗАДАЧІ ПОШУКУ ПРИХОВАНИХ СЕМАНТИЧНИХ ЗВ'ЯЗКІВ В КОРПУСАХ ПРИРОДНОМОВНИХ ТЕКСТІВ

Т.Г. Вознюк

Київський національний університет імені Тараса Шевченка, taarraas@gmail.com

Представлено алгоритм побудови шестивимірного тензора, що містить статистичну інформацію про синтаксичні структури речення. Описано допоміжні засоби, які проводять попередній аналіз природномовного тексту. Наведено приклад роботи алгоритму Невід'ємної Факторизації Тензорів для виділення типових семантичних зв'язків всередині речення.

The paper presents an algorithm for constructing six-dimensional tensor containing statistical information on the syntactic structure of sentences. Conducting preliminary analysis of natural language texts aids was described. An example of Non-negaitve Tensor Factorization algorithm highlighting common semantic relations of sentences was given.

Вступ

На сьогоднішній день розробка систем автоматичного перекладу текстів, експертних систем, природномовних інтерфейсів до баз знань та інших систем інтелектуальної обробки природномовних текстів є дуже перспективним напрямом, оскільки дані програми дозволяють автоматизувати роботу, яку до тепер могли виконувати тільки люди. Сучасні системи обробки природної мови ще роблять помилки, проте з кожним роком кількість помилок зменшується. Наразі стоїть задача покращувати роботу таких систем, робити їх більш надійними.

Кожна з систем обробки природномовних текстів включає у себе етап аналізу текстів. Згідно семіотики, в кожній знаковій системі можна виділити три основні аспекти: синтаксис, семантика та прагматика. Оскільки природномовний текст є знаковою системою, то і в його аналізі прийнято виділяти ці три рівні. Статистичні підходи до синтаксичного аналізу текстів у наш час показують досить високі результати, тоді як в семантичному аналізі є досить багато не вирішених проблем. Методи, що здійснюють аналіз прагматики, опираються на знання про семантику тексту. Тому аналіз семантики наразі є найбільш перспективною та затребуваною сферою досліджень у галузі комп'ютерної лінгвістики.

Серед методів синтаксичного аналізу перспективними виглядають методи, що взяли початок від латентного семантичного аналізу (ЛСА) [1]. Це пояснюється тим, що для своєї роботи він не вимагає кропіткої роботи по складанню словників семантичних значень, розмітки текстів додатковими тегами, написання систем правил, тощо. Всю необхідну для навчання інформацію метод ЛСА намагається знайти в самих природномовних текстах, котрих для кожної мови існує у достатній кількості.

ЛСА – це метод обробки природномовних текстів, зокрема векторної семантики, що полягає в аналізі взаємозв'язків множини документів і термів, що в них зустрічаються, отримуючи в результаті множину концептів відносно документів і термів. Метод припускає, що слова, що були вжиті в схожих семантичних значеннях, будуть зустрічатися в схожих контекстах. Недоліком ЛСА було те, що він був заснований на концепції "мішок слів"(bag of words), тобто не враховував синтаксичні властивості слів при складанні бази. Іншим недоліком є той факт, що метод розглядає лише зв'язки між парами об'єктів, зокрема між парами слів.

Для розширення контексту аналізу до трьох та більше слів було розроблено методи ймовірнісного ЛСА [2] та невід'ємної факторизації тензорів (НФТ) [3].

У роботі [4] розглянуто метод, що відходить від концепції "мішок слів". Замість цього використовується трійка – підмет, присудок та додаток. Метод заснований на НФТ.

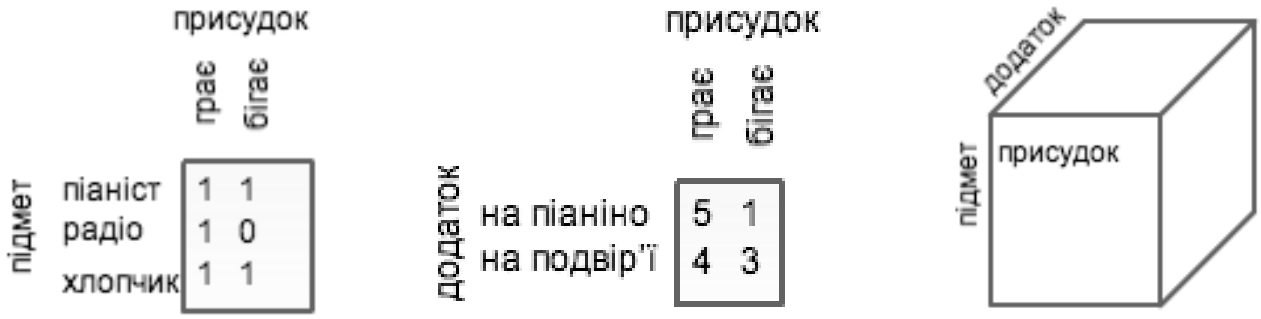
У роботі [5] запропонована ідея використовувати тензори розмірності більшої за 3.

Носій даних – шестивимірний тензор

Основною робочою структурою розроблюваного алгоритму є тензор, в якому буде зберігатися інформація про можливі комбінації слів. Розглянемо для прикладу тензор розмірності два (Рис. 1, а). На його вісях розміщені підмети та присудки. На перетині знаходиться зважений коефіцієнт – частота їх сумісного застосування у природномовних текстах. Так, наприклад, входження "піаніст грає" та "радіо грає" буде більше за нуль, оскільки такі комбінації підмета та присудка зустрічаються у текстах. Можна розглянути інший тензор розмірності два (Рис. 1, б). На його вісях розміщені присудки та додатки, а у ньому частотна оцінки для пари "грає на піаніно" також буде додатною.

Тензори розмірності три були введені через неповноту моделей заснованих на тензорах розмірності два. Для наведеного вище випадку, фраза "радіо грає на піаніно" буде коректною, оскільки її складові "радіо грає" та "грає на піаніно" є коректними. Розглянемо тензор розмірності 3 (Рис. 1, в). Його вісі – підмети, присудки та додатки. Трійка "радіо грає на піаніно" буде входити в нього з значенням нуль, оскільки в природномовних

текстах такі фрази можуть зустрічатися з надзвичайно малою імовірністю. Тоді як “піаніст грає на піаніно” буде входити з вагою більшою за нуль.



а б в

Рис. 1. а – приклад тензору “підмет-присудок”; б – приклад тензору “присудок-додаток”; в – приклад 3-вимірного тензору

Використовуючи аналогічні міркування можна створити тензори ще більших розмірностей. Новими вісями можуть бути прямий об’єкт, не прямий об’єкт, прийменник, модифікатор заперечення, кількості, часу тощо.

З іншого боку обсяг тензора зростає експоненційно в залежності від розмірності, тобто повністю заповнений тензор розмірності n з словником, який містить m слів для кожної категорії, буде займати обсяг рівний m^n. Якщо оцінити кількість слів $m \sim 10^5$, то для $n = 6$, кількість комірок буде дорівнювати $m^n = (10^5)^6 = 10^{30} \sim 2^{64}$, що надмірним навіть для сучасних комп’ютерів. Обмеження на 6 вимірів у тензора викликано не алгоритмічними особливостями, а можливостями комп’ютерів, а тому алгоритм потенційно може бути застосований на тензорах більшої розмірності.

Іншою проблемою є знаходження даних, які будуть оброблені процедурами аналізу. Досить популярний для таких задач є корпус природномовних текстів з енциклопедії Вікіпедія. Її популярність пов’язана з доступністю та обсягом даних, документованістю формату, уніфікованістю структури даних для всіх мов, наявністю додаткових тегів та перехресних посилань. Враховуючи вищезазначене, використано саме Вікіпедію як корпус. Проте алгоритм може працювати із будь-яким корпусом. Для використання іншого корпусу необхідно лише реалізувати інший генератор вхідного потоку(stream) текстів.

Побудова шестивимірного тензору

Після визначення структури носія даних, визначимо метод його заповнення. Спочатку буде описано використані інструменти для виконання попереднього синтаксичного аналізу текстів корпусу. Далі процес виділення шісток слів з тексту. У кінці буде описано структуру збереження даних.

Попередній синтаксичний аналіз текстів. Для попереднього синтаксичного аналізу використано модуль генерації типізованих зв’язків Стенфордського Парсеру [6]. Входом модулю є речення природною мовою. Виходом – множина трійок <тип зв’язку, головне слово, залежне слово>. Наведемо опис використаних типів зв’язку [7]:

dobj: прямий об’єкт.

Прямим об’єктом дієслівної групи є іменникова фраза, що знаходиться в знахідному відмінку.

Приклад : Вони виграли лотерею. dobj(виграли, лотерею);

iobj: не прямий об’єкт.

Не прямим об’єктом дієслівної групи є іменникова фраза, що знаходиться в давальному відмінку.

Приклад : Вона подзвонила мені. iobj(подзвонила, мені);

nsubj: іменний суб’єкт.

Іменний суб’єкт є іменниковою групою, що є синтаксичним підметом речення.

Приклад : Вони виграли лотерею. nsubj(вони, виграли);

prep: прийменниковий модифікатор.

Прийменниковий модифікатор дієслова, прикметника або іменника – будь-який прийменник, який служить для зміни значення дієслова, прикметника, іменника, або навіть іншого прийменника. В цьому випадку ми використовуємо не сам прийменниковий модифікатор, а розширюємо до всієї прийменникової групи, і використовуємо її як єдине слово.

Приклад : Я побачив кота в капелюсі. prep(кота, в). Прийменникова група : в_капелюсі;

root: корінь.

Корінь граматичного відношення вказує на корінь речення. Вироджена вершина “ROOT” використовується як головне слово.

Приклад : Я люблю смажену картоплю. root(ROOT, люблю);

xcomp: відкрите доповнення речення.

Відкрите доповнення речення дієслівної групи – це доповнення речення без особистого підмета, чий референт визначається зовнішнім підметом.

Приклад : Він казав, що ти любиш плавати. xcomp(любиш, плавати).

Всі шість описаних типів зв’язку використовуються як окремі поля тензору, хоча зрозуміло, що не для кожного речення вони будуть існувати.

Наведемо приклад дерева відношень для речення «Bell, based in Los Angeles, makes and distributes electronic, computer and building products». Отриманий результат:

nsubj(makes-8, Bell-1)
nsubj(distributes-10, Bell-1)
partmod(Bell-1, based-3)
nn(Angeles-6, Los-5)
prep in(based-3, Angeles-6)
root(ROOT-0, makes-8)
conj and(makes-8, distributes-10)
amod(products-16, electronic-11)
conj and(electronic-11, computer-13)
amod(products-16, computer-13)
conj and(electronic-11, building-15)
amod(products-16, building-15)
dobj(makes-8, products-16)
dobj(distributes-10, products-16)

Алгоритм виділення елементів тензора. Якщо розглянути слова тексту як вершини, а граматичні відношення Стенфордського Парсеру – як ребра, то отримаємо орієнтований граф (Рис. 2). Цей граф має вид дерева. Його вершиною буде головне дієслово головного речення. Наступна підзадача, яку треба вирішити - пошук та виділення шестірок в цьому графі. Слід пам’ятати, що прості речення в складному утворюють незалежні граматичні одиниці, тому кожна граматична основа речення дозволяє отримати по одному елементу тензора, при чому ці елементи не мають перетинатися по словам. Для розбиття складного речення на прості можна скористатися деревом виведення речення (parsetree).

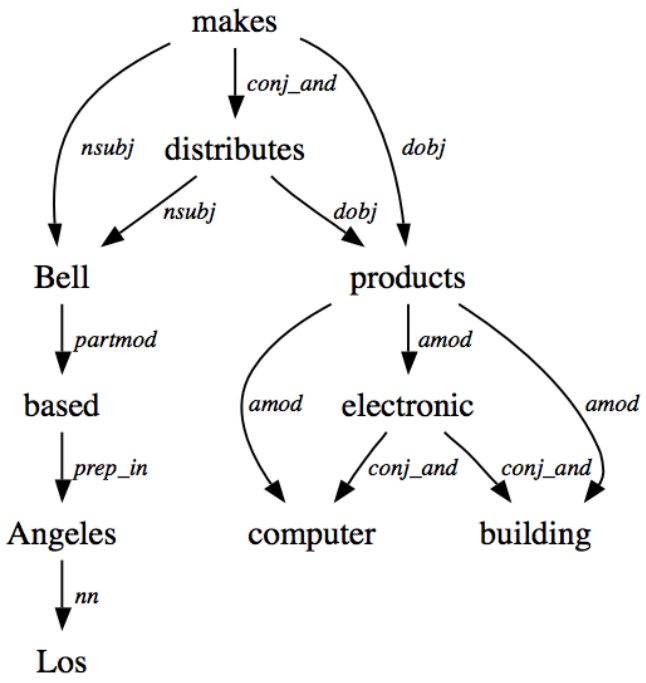


Рис. 2. Приклад графа дерева підпорядкувань

Для виділення слів розбитих по типам для кожного простого речення, що входить до складного, скористаємося рекурсивним алгоритмом:

```
Map<Type, Word> recursiveExtract(Node n) //n – вершина простого речення
begin
   Map<Type, Word> currentSentence;
   for_each(c from n.leafs)
   begin
      currentSentence[c.type] = c.dependentWord; // додаємо нову пару до поточного речення
      newPairs = recursiveExtract(c.dependentNode); // рекурсивний виклик до синівської вершини
      currentSentence.addAll(newPairs);          // додаємо всі пари, що знайдені при аналізі піддерева виводу,
                                                 // що починаються з поточної вершини

   end
end
```

З цієї більш загальної структури даних, нам необхідні лише 6 типів зв'язку. А також потрібно обійти всі прості речення складного. Результуючу множину векторів отримаємо наступною операцією:

```
Set<Vector[6]> parseText(String text)
begin
   Set<Vector[6]> extractedElements;
   for_each(sentence from text)
      for_each(simple_sentence from sentence)
      begin
          t = recursiveExtract(simple_sentence.rootNode());
          v = (t[root], t[nsubj], t[dobj], t[iobj], t[prep], t[xcomp]);
          extractedElements.add(v);
      end
end
```

Розбиття тексту на речення, а також розбиття складного речення на прості виконується відповідними засобами Стенфордського Парсера.

Збереження даних та оптимізація пам'яті. Після обробки кожної статті Вікіпедії необхідно об'єднати новий результат з вже існуючим, що представляє технічну проблему. Як було сказано вище, кількість елементів в пам'яті може досягати 2^{64}, що дорівнює шістнадцяти мільйонам гігабайт. Така кількість даних в оперативній пам'яті сучасного комп'ютера не може вміститися. Складений тензор є дуже розрідженим. Оцінимо необхідний для зберігання обсяг пам'яті.

Слід врахувати, що не всі елементи шісток будуть заповнені словами. Наприклад, речення "Я пишу статтю" має лише підмет, присудок, а також прямий об'єкт, тому фактично є три слова в векторі, і три порожніх вироджених слова, які також займають місце. За рахунок порожніх місць, представлення тензору як множини входжень кожного елемента буде займати більше місця ніж вихідний корпус.

Скориставшись властивістю природномовних текстів, яка полягає в обмеженості кількості унікальних слів та сталих словосполучень природної мови, приходимо до першої оптимізації розміру даних. Визначимо словник елементів тензору, в якому зберігаються пари "унікальне слово" та "порядковий номер". Для частини корпусу Вікіпедії Simple English кількість унікальних слів та словосполучень дорівнює 658058, при середній довжині 12 символів. Це означає, що замінивши слово його числовим представленням розміром 4 байти (для кількості слів достатньо трьох байтів, проте більшість систем мають 32-розрядну архітектуру, а тому швидше оперують з чотирьохбайтними числами), розмір тензору можна зменшити втричі.

З іншого боку, метод латентного семантичного аналізу полягає у пошуку сталих сполучень слів, тому кожний елемент тензору має входити в нього декілька разів без змін. Тобто обсяг тензору можна зменшити ввівши додаткове значення – кількість входжень елемента.

Описана структура даних найкраще підпадає під концепцію баз даних. На Рис. 3 показано ER-модель в нотації Чена для цієї бази. Тут ми маємо дві таблиці – таблиця слів (словник) та таблиця елементів тензора (тензор). Ключем таблиці словника є деякий унікальний номер, що однозначно ідентифікує слово, яке задається довільним текстом (поле «Текст»). Полями кожного елемента тензора є його ключ – ідентифікатор та кількість входжень цього елемента в корпусі. Координати елемента тензора задаються зв'язками типу 1 до багатьох, що в

базах даних зазвичай представляються додатковими полями таблиці. Звідси маємо ще шість атрибутів, які є зовнішніми ключами на елементи таблиці слів.

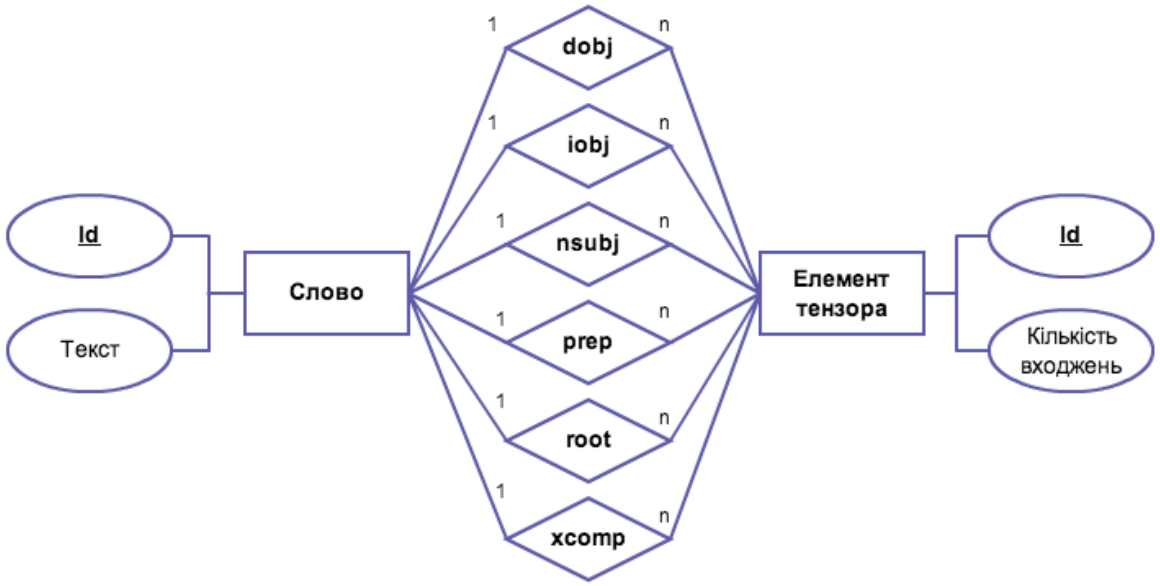


Рис. 3. ER-модель тензора в нотації Чена

Невід'ємна факторизація тензорів

На попередніх етапах отримано тензор, яким вже можна користуватися для аналізу текстів. Проте він має два недоліки. Перший з недоліків – це його великий розмір.

Другий недолік полягає у тому, що в тензор входять тільки фрази, які знайдені в корпусі. З іншого боку при аналізі необхідно, щоб фраза знайдена в аналізованому тексті в точності збігалася з фразою в тензорі. Таким чином ця система вважає правильними лише ті фрази, що знайдені в корпусі. Одним з шляхів вирішення цієї проблеми може бути аналіз дуже великого корпуса, що покриває всі можливі комбінації слів. Проте кількість таких комбінацій є дуже великою, оскільки вона зростає експоненційно з введенням кожного нового слова до мови. Тому такий підхід для вирішення цього недоліку можливий лише при звуженні набору слів мови, а значить не підходить для аналізу довільного природномовного тексту.

Обидві проблеми можливо в певній мірі вирішити за допомогою НТФ. Вона полягає у розкладанні тензора на добуток тензорів меншого порядку. Один з алгоритмів факторизації викладений в роботі [7]. В рамках даної статі буде розглянуто обґрунтування застосування цього алгоритму для виділення сталих семантичних зв'язків.

Розглянемо цей підхід на короткому прикладі з 3-вимірним тензором. Нехай у нього входять наступні елементи:

(дівчинка, збирати, гриби) -> 1,
(хлопчик, збирати, гриби) -> 1,
(хлопчик, збирати, ягоди) -> 1,
(біолог, досліджувати, гриби) -> 1.

В такому представленні прихованих семантичних зв'язків, запит до системи "Чи може дівчинка збирати ягоди?" дасть негативну відповідь. Проте це не відповідає дійсності, бо з контексту видно, що і дівчинка і хлопчик з словом збирати входять в схожі речення, не викликаючи протиріч на семантичному рівні сприйняття тексту.

Розкладемо цей тензор розміру (a,b,c) на 3 матриці – A, B і C.

Добуток матриць визначимо наступним чином:

$$T_f(x,y,z)=\sum_{i=1}^{n}A(x,i)B(y,i)C(z,i),\ \ x=\overline{1,a},\ \ \ y=\overline{1,b},\ \ \ z=\overline{1,c}.$$

Цільова задача факторизації:

$$T_f^{\ *}=\underset{A(x,i)\in\Re^+,B(y,i)\in\Re^+,C(z,i)\in\Re^+,i=\overline{1,n},x=\overline{1,a},y=\overline{1,b},z=\overline{1,c}}{\arg\min}\left\|T_f-T\right\|,$$

де n – кількість-фактор вимірів, тобто кількість груп слів що вживаються в різних значеннях. Чим більше кількість фактор-вимірів, тим меншою буде різниця $\left\|T_f^* - T\right\|$, а тому факторизований тензор буде більш схожий на вихідний. З іншого боку чим менше кількість фактор-вимірів, тим більшим буде "розмиття" даних, а тому система зможе передбачати наявність більшої кількості зв'язків, яких не було в вихідному тексті, і ймовірність помилки буде зростати. Значення кількості фактор-вимірів необхідно підбирати для кожного корпусу окремо, а також виходячи з максимально допустимої кількості помилок роботи системи.

$A =$

	1	2
хлопчик	0.75	0
дівчинка	0.75	0
біолог	0	1

$B =$

	1	2
досліджувати	0	1
збирати	1	0

$C =$

	1	2
гриби	1	1
ягоди	1	0

Таким чином, оцінка того, що (дівчинка, збирає, ягоди) з порядковими номерами (2,2,2) в таблиці є семантично коректною фразою, яка дорівнює

$$T_f(2,2,2) = A(2,1)B(2,1)C(2,1) + A(2,2)B(2,2)C(2,2) = 0.75*1*1 + 0*0*0 = 0.75.$$

Проте оцінки трьох інших фраз (дівчинка, збирати, гриби), (хлопчик, збирати, гриби), (хлопчик, збирати, ягоди) з порядковими номерами відповідно (2, 2, 1), (1, 2, 1), (1, 2, 2) також впали до 0.75. Тому в цільовій системі необхідно використовувати деяке порогове значення, вибране для найкращого співвідношення оцінок покриття та точності (recall/precision). Для даного простого прикладу будь-який поріг в інтервалі (0, 0.75) гарантує найвищі можливі показники точності та покриття.

Нарешті розглянемо, що сталося з оцінкою фрази (біолог, досліджувати, гриби):

$$T_f(3,1,1) = A(3,1)B(1,1)C(1,1) + A(3,2)B(1,2)C(1,2) = 0*0*1 + 1*1*1 = 1.$$

Тому оцінка для цієї фрази не змінилася.

В більш загальному випадку, тензор розмірності m з розмірами $\bar{s} = (s_1, s_2, \ldots, s_m)$ буде представлений як добуток m матриць з n категорій наступним чином:

$$T(\bar{v}) = \sum_{i=1}^{n} \prod_{j=1}^{m} M_j(\overline{v_j}, i).$$

Цільова функція визначається аналогічно до 3-вимірного варіанту:

$$T_f^* = \underset{M_k(i,j)\in\Re^+, i=\overline{1,s_i}, j=\overline{1,n}, k=\overline{1,m}}{\arg\min} \left\|T_f - T\right\|.$$

Висновки

Розглянуто один з підходів до задачі виявлення стійких семантичних зв'язків за допомогою аналізу статистичних даних та запропоновано алгоритм реалізації. Особливістю наведеного алгоритму є використання контексту з 6 елементів. Оскільки обсяг даних тензора росте експоненційно в залежності від кількості елементів контексту, велику увагу приділено інженерним та алгоритмічним аспектам оптимізації пам'яті, яка використовується алгоритмом. Також у роботі обґрунтовано необхідність факторизації отриманого тензора методами НФТ та наведено приклад факторизації для корпусу з декількох речень.

1. *Deerwester; Scott C. (Chicago, IL), Dumais; Susan T. (Berkeley Heights, NJ), Furnas; George W. (Madison, NJ), Harshman; Richard A. (London, CA), Landauer; Thomas K. (Summit, NJ), Lochbaum; Karen E. (Chatham, NJ), Streeter; Lynn A. (Summit, NJ)*. Computer information retrieval using latent semantic structure, 1988.
2. *Thomas Hofmann*. Probabilistic Latent Semantic Indexing // Proceedings of the Twenty-Second Annual International SIGIR Conference on Research and Development in Information Retrieval (SIGIR-99), 1999.
3. *Max Welling and Markus Weber*. Positive Tensor Factorization // Pattern Recognition Letters. – 2001. –22 (12), P. 1255–1261.
4. *TIM VAN DE CRUYS*. A non-negative tensor factorization model for selectional preference induction // Natural Language Engineering. –2010. – 16 (4): 417–437.
5. *Марченко О.О.* Застосування методів невід'ємної факторизації тензорів для визначення стійких семантичних відношень при обробці великих текстових корпусів // Вісник Київського університету: Серія: фізико-математичні науки. – 2012. – № 4.
6. *Daniel Cer, Marie-Catherine de Marneffe, Daniel Jurafsky, and Christopher D. Manning.* Parsing to Stanford Dependencies: Trade-offs between speed and accuracy. 2010.
7. *Marie-Catherine de Marnee and Christopher D. Manning*. Stanford typed dependencies manual, 2008.
8. *Max Welling and Markus Weber*. Positive Tensor Factorization. Pattern Recognition Letters, 2001.