# Лекция 15. Анализ на текста

Едно от най-големи приложения на методи за извличане на закономерности от данни е WEB – глобална информационна мрежа. Голямата част от мрежата се състои от текстови документи, затова методите за автоматичен анализ на текста имат много важно значение в контекста на глобалната мрежа. В настояща лекция ще разгледаме подробно няколко техники за анализиране на текстово съдържание на отделни WEB-страници. Тези техники са били разработени в такива научни области като *извличане на информация (information retrieval - IR)* и *машинно самообучение* и включват индексиране, оценяване и категоризация (класификация) на текстови документи.

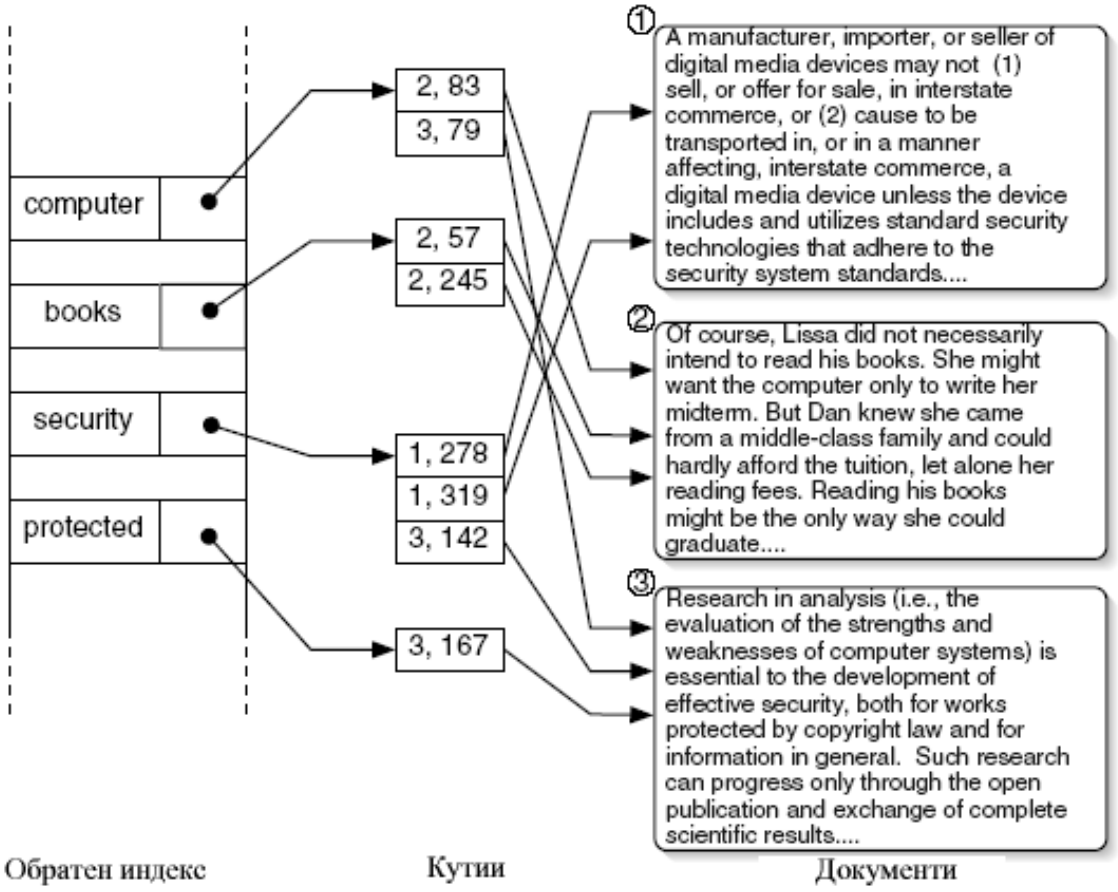
Фокусът на IR е в осигуряването на колкото възможно по-ефективен и по-точен достъп към някое малко подмножество от документи, които са максимално отговарят на интереса на потребителя. Интересът може да бъде изразен, например, като една заявка, зададена от потребителя. Извличането на информацията се състои от два отделни подпроблема: *индексиране на колекцията от документи*, за да подобри изчислителната ефективност на достъпа, и *подреждане на документи* в съгласие с определен критерий за важност, за да подобри точността. *Категоризацията* или класификация на документи е една друга ефективна техника, свързана с извличане на информация. Тя се състои в присвояване на някой документ към една или няколко предварително определени категории. Един класификатор може, например, да се използва за различаване между релевантни и нерелевантни документи (където критерият за релевантността може да бъде персонализиран за отделен потребител или за отделна група от потребители) или да помогне за полуавтоматично създаване на големи WEB-базирани бази от знания или йерархични каталози от теми.

## *15.1. Индексиране*

### 15.1.1. Базови понятия

За ефективно намиране и извличане на текстови документи е необходимо колекцията от документи да бъде обогатена със специализирани структури на данни, които облекчават достъп до документи в отговор на потребителските заявки. Търсенето на под-низове, дори ако то е имплементирано с използване на сложни алгоритми от типа на дървета или масиви на суфикси не е достатъчно за търсене в много големи колекции от текстови документи. В литература бяха предложено множество различни методи за извличане на текста, включително такива ранни подходи като клъстеризация и използване на сигнатурни файлове (signature files). На практика, единствена ефективна техника за работа с много големи множества от документи е *инверсията*. Този метод се опира на създаването на една структура на данни, наречена *обратен индекс (inverted index)*, който асоциира лексическите обекти с техните срещания в колекцията от документи. Лексическите обекти в извличания текст се наричат *термини* и могат да състоят както от думи, така и от изрази. Множеството от интересуващи

нас термини се нарича *речник (vocabulary)* и се означава с $V$. В своята най-проста форма един обратен индекс е речникът, в който всеки ключ е някой термин $\omega \in V$, а асоциираното с него значение $b(\omega)$ е указателят към една допълнителна междинна структура данни, наречена *кутия (bucket)* или *списък на обяви (posting list)*. Кутията, асоциирана с определен термин $\omega$ е по същество един списък от указатели, маркиращи всички появявания на $\omega$ в колекцията от текстове. Общата структура на обратен индекс е показана на фиг. 15.1. В най-простия случай елементите на всяка кутия могат просто да състоят от идентификатора на документа (ДИД) и пореден номер на документа в колекцията. Обаче, в много от случаи е по-удобно да имаш отделен елемент за всяко срещане на термина, където всеки елемент се състои от ДИД и отместване (в символи) на появяването на термина в документа. Този подход има две предимства. Първо, записването на всички срещания на термина позволява на потребителя да му бъде представен кратък контекст (т.е. един фрагмент от окръжаващ термина текст), в който се среща този термин в документа. Второ, информацията за срещане позволява реализацията на приблизителни заявки, такива като извличане на всички документи, в които два указани термина се намират в близки позиции в текста.




*Фиг. 15.1. Структура на един типичен обратен индекс за извличане на информация. Всеки елемент от списъците на срещания (кутии) е двойка от индекси, които идентифицират документа и задават отместване в самия документ, показвайки къде се появява терминът, асоциран с тази кутия.*

Размерът на обратния индекс е $O(|V|)$ и тази структура данни обикновено може да бъде запазена в основната памет.

Той може да бъде имплементиран чрез използване на една хеш-таблица, така че очакваното време за достъп е независимо от размера на речника. Кутиите и хранилището на документи обикновено трябва да се пазят върху диск. Това усложнява процеса на създаване на обратния индекс. Ако кутиите могат да се пазят в основната памет, алгоритъм за построяване на индекса е тривиален. Той просто се състои в парсирането на документи, извличане от тях на термини, вмъкването на термина в обратния индекс, ако той още не присъства в него, и, накрая, вмъкването на срещането на термина в кутията. Обаче, ако кутиите се пазят на диска, този подход ще генерира огромното количество опити за достъп до диска, правейки алгоритъмът практически неизползваем, тъй като достъпът до един случаен байт на информация върху диска отнеме време, което до $10^5$ пъти по-дълго от достъпа към един случаен байт от основната памет. Следователно, за много големи колекции от текстове е необходимо да се прибягваме към алгоритми, специализирани за достъп към вторичната памет.

Търсенето на единичен термин $\omega$ в една индексирана колекция от документи е съвсем просто. Първо, ние правим достъп към обратния индекс, за да получим $b(\omega)$. След това ние просто сканираме кутията, на която указва $b(\omega)$, за да получим списъка на срещания. Случаят на булевите заявки с няколко термина е само малко по-сложен. Да предположим, че заявката съдържа литерали, асоциирани с присъствие или отсъствие на $k$ термина. Предишната процедура може да бъде повторена по-отделно за всеки термин, произвеждайuи $k$ списъка на срещания. След това тези списъци се комбинират чрез елементарни операции върху множества, съответстващи на булевите операции, описани в заявката: пресичането за логическото И, обединението – за ИЛИ, и допълнението – за НЕ.

## *15.2. Лексическа обработка*

### 15.2.1. Токенизация

*Токенизацията (tokenization)* е извличането на "плоски" (plain) думи или термини от един документ, които са "очистени" от административни мета данни и структурни или форматиращи елементи, например, премахване на HTML етикети (tags) от изходния (source) файл на HTML на някоя Web страница. Тази операция трябва да бъде изпълнена преди индексирането или преди превръщането на документи във векторното представяне, което се използва за тяхното извличане или категоризация (виж следващите раздели). Токенизацията изглежда да е лесен за решаване проблем, но в много от практически ситуации тази задача може да бъде доста нетривиална.

Случаят на HTML документи е относително лек. Най-простият подход се състои в свеждането на документа към едно *неструктурирано* представяне, т.е. към една "плоска" последователност от думи без никакви специфични релации между тях, освен подреждането по реда. Това може да бъде постигнато чрез запазване само на текста, заключения между етикети <html> и </html>,

премахване на всички етикети, и, възможно, чрез превръщането на низове, които кодират международни символи, в едно стандартно представяне (например Unicode). Полученото неструктурирано представяне позволява прости заявки, отнасящи се за наличие на термини и тяхната позиционна близост. В по-общ случай, допълнителната информация, вградена в HTML, може да бъде използвана за изобразяване на документи в *полу-структурирани* представяния. В този случай представянето е чувствително към присъствие на термини в различни елементи на документа и позволява по-сложни заявки от типа на "намери документи, съдържащи *население* в заглавието на таблица и *глад* в заглавие на документа". По принцип, извличането на полу-структурирани представяния от HTML документи е много лесно, тъй като етикетите предоставят цялата необходима структурна информация. Обаче, трябва да се има предвид, че макар HTML има една ясно дефинирана формална граматика, реалните браузери не налагат стриктното съблюдаване на коректния синтаксис и, като резултат, повечето от Web страници не отговарят точно на синтаксиса на HTML. Следователно, за да бъде използван в практиката, един HTML парсер трябва да толерира грешки и да се включва в себе си механизми за възстановяване (recovery). Един публично достъпен парсер се разпространява с LIBWWW – протоколната библиотека на W3C http://www.w3.org/Library/. HTML парсер, написа на Perl (с методи за извличане на текста) е достъпен от http://search.cpan.org/dist/HTML-Parser/.

Очевидно е, че след извличане на "плосък" текст е необходимо да бъдат премахнати от него всякакви знаци за пунктуация и специални символи. Освен това, заглавните (големи) букви могат да бъдат превърнати в малки, за да се намали броя на индексирани термини.

Освен HTML, текстови документи във Web се срещат в голямо множество от различни формати. Някои формати са частни и не е описани, което ограничава извличане на текстове от подобни типове файлове. Други формати са публично известни, като например Post Script и PDF .

Токенизацията на PostScript или PDF файлове може да бъде сложна за реализация тъй като това не са формати за данни, а алгоритмичните описания на това, как документът трябва да бъде представян. В частност, PostScript е един интерпретационен език за програмиране в който представяне на текста се управлява от едно множество *команди за показване (show commands)*. Аргументите на тези команди за показване са текстови низове и двумерни координати. Обаче, тези низове не е задължително цели думи. Например, за да бъдат изпълнени такива печатарски операции като показване на опашки на букви (kerning), лигатура или пренос, думите обикновено се разделят на фрагменти и за всеки фрагмент се издават отделни команди за показване. Командите за показване не е задължително да бъдат в реда на четене на думите, така че за правилната реконструкция на границите на думи е необходимо да бъдат проследявани двумерните позиции на всеки "показван" низ и да бъде използвана информация за шрифта. PostScript е почти винаги се генерира автоматично от други програми от типа на драйверите на принтери, което още повече усложняват нещата, тъй като различните генератори следват различни съглашения и подходи. На практика е не винаги е възможно абсолютно перфектно преобразуване. Един от примери за преобразуватели от PostScript в

“плосък” текст е Pstotxt, разработен в Нова Зеландия в рамките на проекта Virtual Paper (http://research.compaq.com/SRC/virtualpaper/home.html).

PDF е един двоичен формат, който се базира на същия основен модел за показване както и PostScript, но който може да съдържа и допълнителните парчета от информация, включващи описателните и административни мета данни, както и структурни елементи, хипервръзки и дори звук и видео. В термините на предоставеното съдържание, PDF файлове са по-близки по своята структура до Web страници, от колкото до PostScript файлове. PDF файлове могат да съдържат растерните изображения на сканирани текстови документи. За да бъде извлечен текст от подобни растерни изображения е необходимо да бъдат използвани системи за оптично разпознаване на символи (OCR).

### 15.2.3. Комбиниране на версиите на текста и намаляване на речника

Често е желателно да бъдат намалени всички морфологични варианти на дадената дума, свеждайки ги до един единствен индексиран термин. Например, един интересуващ ни документ може да съдържа няколко срещания на такива думи като *fish, fishes, fisher* и *fishers*, но те няма да бъдат извлечени в отговор на заявката с ключова дума fishing, ако терминът *fishing* никога не се среща в текста. Процесът на извличане на основи на думите (stemming) се състои в намаляването на думите до тяхната основна (коренна) форма (като *fish* – в нашия пример), която става действителен индексиран термин. Освен ефекта, оказващ влияние върху извличането на информация, процесът на извличане на основите на думите може да се ползва и за намаляване на размера на индекса.

Един от най-известни алгоритми за извличане на основите на думите за английски език е създаден от Porter (1980). Той се базира върху един предварително създаден списък на суфикси с асоциирани с тях правила. Един пример на такова редактиращо правило е “ако суфиксът на думата е IZATION и префиксът съдържа най-малко една гласна, след която следва някоя съгласна, то замени този суфикс с IZE”. Това правило ще трансформира, например, думата BINARIZATION в думата BINARIZE. Алгоритъмът на Porter прилага правила в пет последователни стъпки. Изходен (source) код на няколко езика и подробното описание на алгоритъма са достъпни от `http://www.tartarus.org/~martin/PorterStemmer/`.

Една друга техника, която често се използва за управление на речника, е премахване на “спиращи” думи, т.е. често срещани думи от типа на артикли (членове), предлози и наречия, които не носят информация за семантичното съдържание на документа. Тъй като “спиращи” думи се срещат много често, тяхното премахване от речника може значително да намали размера на индекса. На практика, в един голям текст “спиращи” думи могат да заемат до 20-30% от целия обем. Естествено е, че премахването на “спиращи” думи също така подобрява изчислителната ефективност на процеса на извличане на информацията.

## *15.3. Подреждане, базирано на съдържание*

Една булева заявка към някоя машина за търсене във Web връща няколко хиляди подходящи (matching) документа, обаче един обикновен потребител е в състояние да провери само една малка част тях. По тази причина подреждането на подходящи документи в съгласие с тяхна релевантност към заявката на потребителя е един фундаментален проблем. В този раздел ще разгледаме няколко класически подхода, които не използват структурата на хипервръзки във Web.

### 15.3.1. Модел на векторното пространство

Текстови документи могат да бъдат удобно представени в многомерно векторно пространство, в който термите са асоциирани с компонентите на вектори. По-точно, един текстови документ $d$ може да се представи като последователността от термини $d = (\omega_1, \omega_2, \ldots, \omega_{|d|})$, където $|d|$ е дължината на документа, а $\omega_t \in V$. Представянето на $d$ във векторното пространство се дефинира като един вектор от реални числа $\mathbf{x} \in \Re^{|V|}$, където всеки компонент $x_j$ е статистиката, отнасяща се до срещането на j-я елемент на речника в документа. Най-простото векторно представяне е булево, т.е. $x_j \in \{0, 1\}$, указвайки на присъствие или отсъствие на термина $\omega_j$ в представения документ.

Векторното представяне понякога се нарича “ чанта с думи”, отбелязвайки, че векторите на документи са инвариантни по отношение на пермутациите на термините, тъй като оригиналният ред на думи $\omega_1, \ldots, \omega_{|v|}$ е просто липсва. Представянията от този тип са привлекателни със своята простота. Освен това, макар че те са губещи от информационна гледна точка, много от задачите по извличане и категоризация на текста могат да бъдат решени доста добре, на практиката, чрез използване на този векторен модел. Обърнете внимание, че обикновено общият брой на термини в едно множество от документи е значително по-голямо от броя на различни термини във всеки един от документи: $|V| >> |d|$, така че векторното представяне има тенденцията да бъде много *разпръснато (sparse).* Това свойство може да бъде използвано с полза както при запазването на паметта, така и при конструиране на алгоритми.

### 15.3.2. Сходство на документи

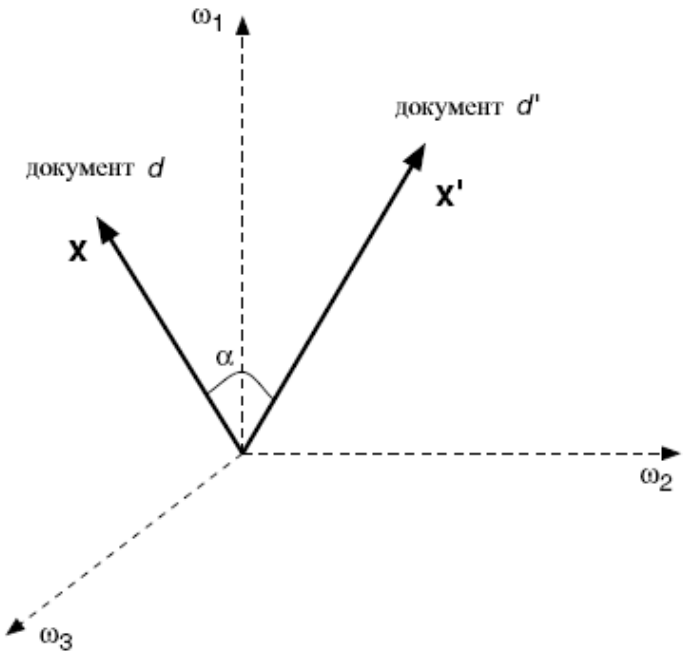
Сходството между два документа $d$ и $d'$ може да се дефинира като една функция $s(d, d') \in \Re$. Тази функция, освен други неща, позволява подреждане на документи по отношение на заявката (чрез измерване на сходството между всеки документ и заявката). Един класически подход се базира на векторното представяне и метриката, дефинирана чрез *коефициент на косинуса*. Тази мярка е просто косинус на ъгъла, формиран от векторното представяне на два документа $\boldsymbol{x}$ и $\boldsymbol{x'}$ (виж фиг. 15.2).

$$\cos(\mathbf{x}, \mathbf{x}') = \frac{\mathbf{x}^T \mathbf{x}'}{\|\mathbf{x}\| \cdot \|\mathbf{x}'\|} = \frac{\mathbf{x}^T \mathbf{x}'}{\sqrt{\mathbf{x}^T \mathbf{x}} \cdot \sqrt{\mathbf{x}'^T \mathbf{x}'}} \quad (15.1)$$

където индекс "T" означава оператор на транспозицията, а $\boldsymbol{x}^{\mathrm{T}}\boldsymbol{y}$ означава скаларното произведение на двата вектора $\mathbf{x}, \mathbf{y} \in \Re^m$, определено като

$$x^{\mathrm{T}} y \doteq \sum_{i=1}^{m} x_i y_i . \quad (15.2)$$

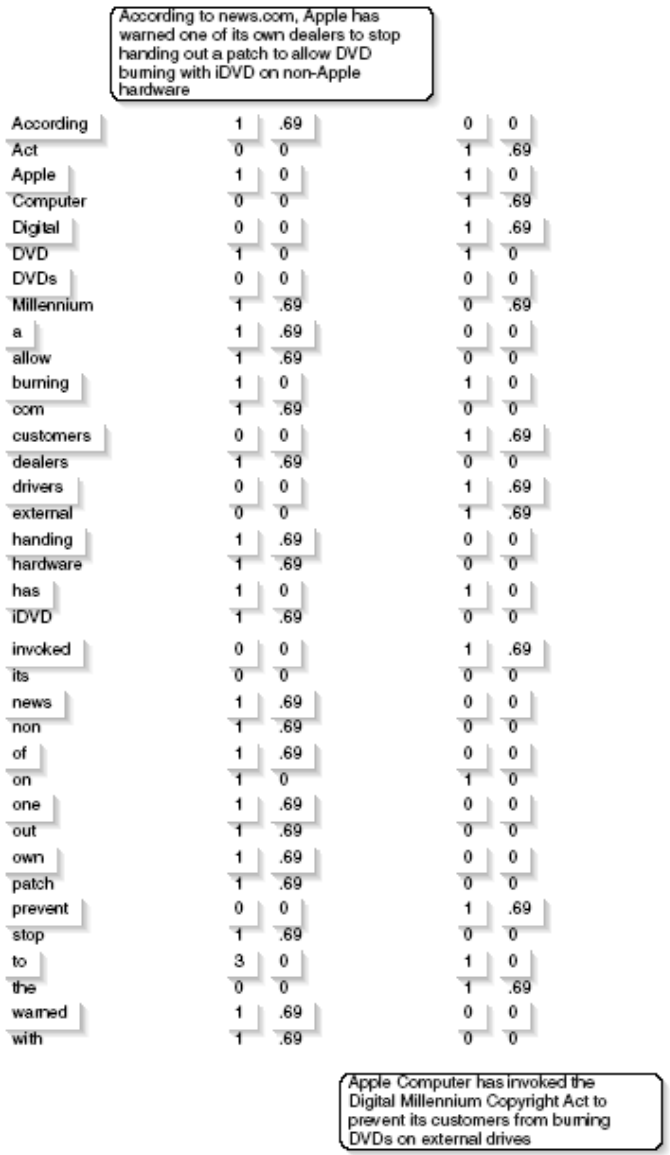
Обърнете внимание, че в случая на разпръснати вектори $\boldsymbol{x}$ и $\boldsymbol{y}$, асоциирани с два документа $d$ и $d'$, тази сума може да се изчисли ефективно за време $O(|d| + |d'|)$.




*Фиг. 15.2. Мярката на косинус при сходството на документи*

Чрез разширяване на булевия векторен модел и въвеждане на реални тегла, асоциирани с термините в документа, могат да бъдат постигнати няколко подобрения. Една по-информативна тегловна схема се състои в изчисляването на действителния брой на срещания на всеки термин в документа. В този случай $x_j \in N$ е бройка на срещания на терма в съответния документ (виж фиг. 15.3). $\boldsymbol{x}$ може да бъде умножен на константата $1/|d|$, за да се получи вектор от *честотите на срещания* на термини (*TF- term frequency*) в документа.

Една важна фамилия на тегловни схеми, която комбинира честотите на срещания на термини (относителни за всеки документ) с една "абсолютна" мярка за важността на термина, се нарича *обратна честота на срещане на документа (IDF – invert document frequency)*. IDF намалява, когато в зададена колекция броят на документи, в които терминът участва, се увеличава. Така че термините, които са глобално се срещат рядко, получават по-големи тегла.




*Фиг. 15.3. Векторно представяне на документи. За всеки от тези два документа левият вектор съдържа броя на срещания на всеки термин, докато десният вектор се базира на теглата TF-IDF (Уравнение 15.5).*

Формално, нека $D = \{d_1, \ldots, d_n\}$ е някоя колекция от документи и за всеки терм $\omega_j$ нека $n_{ij}$ означава броя на срещания на $\omega_j$ в $d_i$, а $n_j$ е броят на документи, съдържащи $\omega_j$ най-малко един път. Тогава, ще дефинираме:

$$\mathrm{TF}_{ij} \doteq \frac{n_{ij}}{|d_i|}, \qquad (15.3)$$

$$\mathrm{IDF}_j \doteq \log \frac{n_j}{n}. \qquad (15.4)$$

Тука логаритмическата функция се използва като намаляващ фактор.

TF-IDF теглото (Salton et al. 1983) на $\omega_j$ в $d_i$ може да се изчисли като:

$$x_{ij} = TF_{ij} \cdot IDF_j \qquad (15.5)$$

или, алтернативно, като:

$$x_{ij} = \frac{TF_{ij}}{\max\limits_{\varpi_k \in d} TF_{ik}} \times \frac{IDF_j}{\max\limits_{\varpi_k \in d} IDF_k} \quad (15.6)$$

Претеглянето през IDF често се използва като една ефективна евристика. Неотдавна Papineni (2001) е предложил теоретичната обосновка, доказвайки че IDF е оптималното тегло на термина по отношение на минимизирането на една функция на разстояние, която е обобщение на относителната ентропия.

### 15.3.3. Извличане на информация и мерки за оценяване

Да предположим, че имаме някоя колекция $D$ от $n$ документа. Всеки документ е представен като един m-мерен вектор, където $m = |V|$, а $V$ е множеството от термини, които се срещат в колекцията $D$. Нека $\mathbf{q} \in \Re^m$ означава векторът, асоцииран със заявката на потребителя (термините, които се срещат в заявката, но не са в $V$ се игнорират). На всеки документ се присвоява някаква оценка за неговата релевантност към заявката, изчислена чрез сходството $s(x_i, q), i= 1, ..., n$. Множеството $R$ от *извличани документи*, които се представят на потребителя, може да бъде формирано чрез събиране на най-добре класирани документи съгласно мярката на сходство. Качеството на тази представена на потребителя колекция може да се дефинира чрез сравняване на $R$ с множеството от документи $R^*$, които действително са релевантни на заявката[1].

Две от най-често използвани метрики за сравнение на $R$ и $R^*$ са *прецизност (precision)* и *припомняне (recall)*. Прецизността $\pi$ се дефинира като тази част от извличаните документи, които са действително релевантни. Припомнянето $\rho$ се дефинира като тази част от релевантни документи, които се извличат от системата. По точно:

$$\pi \doteq \frac{|R \cap R^\star|}{|R|}, \qquad \rho \doteq \frac{|R \cap R^\star|}{|R^\star|}.$$

Обърнете внимание, че в този контекст отношение между релевантни и нерелевантни документи обикновено е много малко. По тази причина, други често използвани (в задачи за класификация) мерки за оценяване от типа на точността (accuracy) или пропорцията на грешки (error rate), в които знаменателят се състои от $|D|$, няма да бъдат подходящи (те ще толерират извличане на нула документи, намалявайки по този начин пропорцията на грешките и, следователно, получавайки много висока точност). Понякога, прецизност и припомняне се комбинират в едно единствено число, наречено мярката $F_\beta$, дефинирана като

$$F_\beta \doteq \frac{(\beta^2 + 1)\pi\rho}{\beta^2\pi + \rho}. \quad (15.7)$$

[1] Естествено, че това е възможно само за специално подготвени колекции от документи.

където $\beta$ е някоя неотрицателна константа. Обърнете внимание, че когато $\beta = 1$, то получената мярка $F_1$ е хармоничното средно от прецизността и припомнянето, а когато $\beta$ се стреми към нула (или безкрайността), то съответната мярката $F_\beta$ се стреми към прецизността (или припомнянето).

## 15.4. Вероятностно извличане на информация

Изследователи в областта на извличането на информация отдавна задавали въпроса, как да бъде определен някой критерий за оптималност за подреждане на извличаните документи в отговор за заявката, зададена от потребителя. В 1977 г. Robertson формулира един принцип на оптималността, наречен *принцип за вероятностното подреждане (PRP – probabilistic ranking principle)*, който утвърждава следното:

> *Ако отговорът на една система за извличане на информация на всяка заявка е подреждането на документи по реда на намаляване на вероятността за тяхната релевантност към предпочитания на потребителя, който изпрати тази заявка, където вероятностите са оценени колкото може по-точно на базата на всички налични данни, достъпни на системата за тази цел, то общата ефективност на такава система за нейния потребител ще бъде най-добрата от всички, които могат да бъдат постигнати на базата на тези данни.*

PRP-принципът прилича на Бейсово оптимално правило за задачите на класификацията (вижте лекции по *Машинно самообучение* за подробностите). Бейсовото оптимално решение (максимално апостериорно вероятна хипотеза) се постига чрез осигуряване, че апостериорната вероятност на правилния клас (при наблюдавани данни) трябва да бъде по-голяма от апостериорната вероятност на всеки друг клас. PRP-принципът може да бъде формулиран математически чрез въвеждане на една допълнителна булева променлива $R$ (релевантност) и чрез дефиниране на модела за условната вероятност $P(R| d, q)$ на релевантността на документа $d$ към потребностите на потребителя (например, изразени чрез заявката $q$). Правилността на изказания по-горе принцип може да бъде доказана по следния начин. Нека да означим с $c$ цената на извличане на един релевантен документ, а чрез $\overline{c}$ - цената за извличане на един нерелевантен документ, като считаме, че $c < \overline{c}$. За да минимизираме общата цена, един документ $d$ трябва да бъде извлечен като следващия по ред, ако:

$$cP(R \mid d,q) + \overline{c}(1 - P(R \mid d,q)) \leqslant cP(R \mid d',q) + \overline{c}(1 - P(R \mid d',q)) \quad (15.8)$$

за всеки документ $d'$, който още не е извлечен. Тъй като $c < \overline{c}$, то това условие е еквивалентно на:

$$P(R \mid d,q) \geqslant P(R \mid d',q),$$

т.е. документите трябва да бъдат извлечени по реда на намаляването на вероятността за тяхната релевантност.

За да построим модела за вероятността на релевантност се нуждаем от определени опростяващи допускания. Най-простият възможен подход се нарича *извличане при двоична независимост (BIR – binary independence retrieval)*

(Robertson and Jones 1976), който подобно на наивен Бейсов класификатор постулира условната независимост между термините. Подходът предполага, че генерирането на един документ може да бъде моделирано като подхвърляне на $|V|$ независими монети, като срещането на всяка дума в документа се подчинява на разпределението на Бернули.

Нека да въведем шансовете[2] на релевантността и да приложим теоремата на Бейс:

$$O(R \mid d,q) = \frac{P(R=1 \mid d,q)}{P(R=0 \mid d,q)} = \frac{P(d \mid R=1,q)}{P(d \mid R=0,q)} \cdot \frac{P(R=1 \mid q)}{P(R=0 \mid q)}. \quad (15.9)$$

Обърнете внимание, че последната дроб е шансовете на $R$ при зададена $q$ – постоянна величина за цялата колекция от документи (тя зависи само от заявката), следователно тя може да не се взима под внимание. Прилагайки BIR-предположението, първата дроб може да бъде преписана по следния начин:

$$(15.10) \quad \frac{P(d \mid R=1,q)}{P(d \mid R=0,q)} = \prod_{i=1}^{|V|} \frac{x_j P(\omega_j \mid R=1,q) + (1-x_j)(1-P(\omega_j \mid R=1,q))}{x_j P(\omega_j \mid R=0,q) + (1-x_j)(1-P(\omega_j \mid R=0,q))}.$$

Ние трябва да оценим параметрите:

$\theta_j \doteq P(\omega_j \mid R=1,q)$ и $\eta_j \doteq P(\omega_j \mid R=0,q)$

Ако предположим, че $\theta_j = \eta_j$, ако $\omega_j$ не присъства в $q$, то накрая получаваме:

$$O(R \mid d,q) = O(R \mid q) \prod_{j:\omega_j \in q} x_j \frac{\theta_j}{\eta_j} \cdot (1-x_j) \frac{1-\theta_j}{1-\eta_j}, \quad (15.11)$$

където произведението се прави само по индекси $j$, които са асоциирани с термините $\omega_j$, присъстващи в заявката. Това може да се препише по следния начин:

$$O(R \mid d,q) = O(R \mid q) \cdot \prod_{j \in q} \frac{1-\theta_j}{1-\eta_j} \cdot \prod_{\substack{j:\omega_j \in q, \\ x_j=1}} \frac{\theta_j(1-\eta_j)}{\eta_j(1-\theta_j)}, \quad (15.12)$$

където само последната дроб зависи от документа. Следователно, *величината на ранга за извличане (RSV – retrieval status value)* на един документ се изчислява като логаритъм от последната дроб:

$$\mathrm{RSV}(d) = \sum_{j:\omega_j \in d \cup q} \log \frac{\theta_j(1-\eta_j)}{\eta_j(1-\theta_j)}. \quad (15.13)$$

По тази причина, извличаните документи трябва да бъдат подредени в реда, обратен на RVS.

[2] Шансове на едно случайно събитие A е отношението на вероятността, че това събитие ще се случи, към вероятността, че то няма да се случи: $O(A) = \frac{P(A)}{P(\neg A)}$

## 15.5. Анализ на латентна (скрита) семантика

Подходът към извличане на документи, базиран на сходството между векторите, може да постигне удовлетворителната степен на припомнянето само тогава, когато термините от заявката присъстват в релевантните документи. От другата страна, потребителите често формулират своите заявки, избирайки термини, които изразяват *понятия*, свързани със съдържанието на документи, които те бих искали да намерят. Естественият език има много голяма изразителна сила и дори на лексическото ниво неговата голяма изменчивост в следствие на синонимия и полисемия (многозначност) предизвиква сериозни затруднения за методите за извличане на информация, базирани на съвпадения на термини. Синонимията означава, че едно и също понятие може да бъде изразено чрез използване на различни множества от термини. Синонимията негативно влияе на припомнянето. Полисемията означава, че едни и същи термини могат да се използват в различни семантични контексти. Полисемията негативно влияе на прецизността. Като цяло, синонимията и полисемията водят до сложни релации между термини и понятия, които не могат да бъдат уловени чрез процедурата на просто съвпадение.

По този начин, макар че една заявка може да бъде концептуално много близка до определено множество от документи, асоцииранят с нея вектор може да се окаже ортогонален или близо до ортогонален към векторите на тези документи, само защото авторите на документите и потребителят използвали един и същ език по различни начини. Друг по-статистически ориентиран начин за размисъл върху тази ситуация е да представим, че термините, които присъстват в един документ, представляват една относително малка част от целият речник, което намалява вероятността, че двата документа използват едно и също множество от думи, за да изразят едно и също понятие.

*Индексирането на скритата семантика (LSI – latent semantic indexing)* е една статистическа техника, която опитва да оцени скритата структура, която генерира термините при зададени понятия. Тя използва техниката от линейната алгебра, известна като *декомпозиция на сингуларни стойности (SVD – singular value decomposition)*, за да намери най-важни асоциативни шаблони между думи и понятия. LSI е метод, воден от данни, при който за намирането на статистически най-значими съответствия между термините се използва една голяма колекция от изречения или документи.

### 15.5.1. Индексиране на скритата семантика и текстови документи

Нека $\boldsymbol{X}$ означава матрицата термини-документи, дефинирана като

$$X = [x_1 \cdots x_n]^{\mathrm{T}}. \quad (15.15)$$

Всеки ред в тази матрица е вектор, представящ един документ. Всяка колонка съдържа броя на срещания на един термин във всеки документ от дадената колекция. Техниката на LSI се състои в изчисляването на SVD на $\boldsymbol{X}$ и нулиране на всички сингуларни значения освен $K$ най-големи. За практическите

приложения $K$ често се избира в диапазона между 100 и 1000. Това може да се разглежда като аналогията на метода на основни компоненти (PCA), прилаган за намаляването на размерността на пространството на признаци, който разглеждахме в лекцията, посветена на намаляването на обема на данни. След SVD документите се изобразяват в едно по-маломерно пространство на скритата семантика, създадено чрез избиране на направления с максимална ковариация.

Да разгледаме работата на LSI върху един пример. В таблица 15.1 е показана колекцията от 10 документа. Първите пет от тях са свързани с операционната система Linux, а останалите пет – с новини в областта на генетиката. За простота ще се фокусираме само на често срещани термини, в частност на тези, които се срещат най-малко два пъти. След изтриването на такива общи думи като ‘the” и “goes”, получената матрица термини-документи $\boldsymbol{X}$ е показана в средата на таблица 15.1 (транспозицията на X се използва само за пестене на място). Прилагането на SVD в този пример води до следната диагонална матрица от сингулярни стойности

$$\boldsymbol{\Sigma} = \text{diag}(2.57, 2.49, 1.99, 1.9, 1.68, 1.53, 0.94, 0.66, 0.36, 0.10).$$

Чрез установяване на $\hat{\boldsymbol{\Sigma}} = diag(2.57,2.49,0,0,0,0,0,0,0,0)$ получаваме реконструкцията на $\boldsymbol{X}$: $\hat{\mathbf{X}} = \mathbf{U}\hat{\boldsymbol{\Sigma}}\mathbf{V}^{\mathbf{T}}$, която е показана в дъното на таблица 15.1.

Както се вижда, реконструираната матрица $\hat{\mathbf{X}}$ не е толкова разредена, както оригиналната матрица $\boldsymbol{X}$. Новите тегла на термините са изчислени за съвместно срещане на думи и извеждат релации между думите, които наподобяват синонимията, най-малко в тесния смисъл на думата. Например, една заявка с термин “Linux” сега ще назначава относително голяма оценка на документи 2 и 3, макар че те изобщо не съдържат търсената ключова дума. Този резултат може да се обясни с това, че SVD изведе връзката между “Debian” или “Gentoo” и “Linux” чрез съвместно срещането на други думи (самите те никога не се срещат заедно в използвания пример). Аналогично, от таблицата може да се види, че “DNA” и “Dolly” имат документни вектори близки до документния вектор на “genome”.

При реализацията на метода за изчисление на SVD за много голяма матрица от документи, важно е да бъде използвано предимството от нейната разреденост. За тази цел са разработени множество известни приближени числови методи (виж, например Berry and Browne 1999).

| | |
|---|---|
| $d_1$: | Indian government goes for open-source software |
| $d_2$: | Debian 3.0 Woody released |
| $d_3$: | Wine 2.0 released with fixes for Gentoo 1.4 and Debian 3.0 |
| $d_4$: | gnuPOD released: iPod on Linux... with GPLed software |
| $d_5$: | Gentoo servers running an open-source mySQL database |
| $d_6$: | Dolly the sheep not totally identical clone |
| $d_7$: | DNA news: introduced low-cost human genome DNA chip |
| $d_8$: | Malaria-parasite genome database on the Web |
| $d_9$: | UK sets up genome bank to protect rare sheep breeds |
| $d_{10}$: | Dolly's DNA Damaged |

| | $d_1$ | $d_2$ | $d_3$ | $d_4$ | $d_5$ | $d_6$ | $d_7$ | $d_8$ | $d_9$ | $d_{10}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| open-source | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| software | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| Linux | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| released | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| Debian | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Gentoo | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| database | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| Dolly | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| sheep | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| genome | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| DNA | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 1 |

| | $d_1$ | $d_2$ | $d_3$ | $d_4$ | $d_5$ | $d_6$ | $d_7$ | $d_8$ | $d_9$ | $d_{10}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| open-source | 0.34 | 0.28 | 0.38 | 0.42 | 0.24 | 0.00 | 0.04 | 0.07 | 0.02 | 0.01 |
| software | 0.44 | 0.37 | 0.50 | 0.55 | 0.31 | −0.01 | −0.03 | 0.06 | 0.00 | −0.02 |
| Linux | 0.44 | 0.37 | 0.50 | 0.55 | 0.31 | −0.01 | −0.03 | 0.06 | 0.00 | −0.02 |
| released | 0.63 | 0.53 | 0.72 | 0.79 | 0.45 | −0.01 | −0.05 | 0.09 | −0.00 | −0.04 |
| Debian | 0.39 | 0.33 | 0.44 | 0.48 | 0.28 | −0.01 | −0.03 | 0.06 | 0.00 | −0.02 |
| Gentoo | 0.36 | 0.30 | 0.41 | 0.45 | 0.26 | 0.00 | 0.03 | 0.07 | 0.02 | 0.01 |
| database | 0.17 | 0.14 | 0.19 | 0.21 | 0.14 | 0.04 | 0.25 | 0.11 | 0.09 | 0.12 |
| Dolly | −0.01 | −0.01 | −0.01 | −0.02 | 0.03 | 0.08 | 0.45 | 0.13 | 0.14 | 0.21 |
| sheep | −0.00 | −0.00 | −0.00 | −0.01 | 0.03 | 0.06 | 0.34 | 0.10 | 0.11 | 0.16 |
| genome | 0.02 | 0.01 | 0.02 | 0.01 | 0.10 | 0.19 | 1.11 | 0.34 | 0.36 | 0.53 |
| DNA | −0.03 | −0.04 | −0.04 | −0.06 | 0.11 | 0.30 | 1.70 | 0.51 | 0.55 | 0.81 |

*Таблица 15.1. Пример на прилагането на LSI. Горе: колекцията от документи. Термините, използвани за анализа, са подчертани. В центъра: матрицата термини – документи $\boldsymbol{X}^T$. Долу: реконструираната матрица термини – документи $\hat{\mathbf{X}}$ след проектирането в подпространството с размерност K = 2.*

## 15.6. Намиране на асоциации между думите

Един от аспекти на анализа на текста се състои в намиране на интересни асоциации между думите чрез използване на методи за асоциативен анализ, описани подробно в лекции 11 и 12. Да предположим, че интересуващи ни

документи са представени във вид на матрицата „документи – думи", както това е показано в Таблица 15.2. Важно е да се отбележи, че всяка клетка на матрицата съдържа *нормализирана* честота на срещане на съответната дума в дадения документ. Нормализацията е направена чрез разделяне на честотата на срещане на всяка дума върху сумата на честотите на срещане на тази дума във всички документи. Една от причините за такава нормализация е да осигури, че получената стойност на поддръжка е между нула и единица. Обаче, по-важната причина е да осигури, че всички данни са представени върху една и съща скала, така че множествата от думи, които се променят по един и същ начин, да имат подобни стойности на поддръжка.

| *Документ* | *Дума$_1$* | *Дума$_2$* | *Дума$_3$* | *Дума$_4$* | *Дума$_5$* | *Дума$_6$* |
|---|---|---|---|---|---|---|
| $d_1$ | 0.3 | 0.6 | 0 | 0 | 0 | 0.2 |
| $d_2$ | 0.1 | 0.2 | 0 | 0 | 0 | 0.2 |
| $d_3$ | 0.4 | 0.2 | 0.7 | 0 | 0 | 0.2 |
| $d_4$ | 0.2 | 0 | 0.3 | 0 | 0 | 0.1 |
| $d_5$ | 0 | 0 | 0 | 1.0 | 1.0 | 1.0 |

*Таблица 15.2. Нормализирана матрица документи-думи*

Тъй като всяка колонка на такава матрица може да се разглежда като непрекъснат атрибут, то един от възможните способи към нея да бъде приложен някой традиционен алгоритъм за асоциативен (булев) анализ (от типа на Apriori – виж лекция 11) се състои в дискретизацията (и след това - в бинаризация) на непрекъснатите атрибути. Обаче, при анализа на текста ние се интересуваме в намиране на асоциации между думите (например, между data и mining), а не в асоциациите между интервалите на честоти на срещания на тези думи (например, data$\in$[1,4] и mining$\in$[2, 3]). По тази причина можем да опитаме да бинаризираме съответните непрекъснати атрибути *без* да извършваме никаква дискретизация. Например, можем да превърнем нашата матрица в матрица, съдържаща само стойности 0/1 чрез прилагане на една проста прагова трансформация – стойността на една клетка от таблицата става равна на 1, ако съответната нормализирана честота на срещане на думата в тази клетка е над зададения праг $t$ и 0 – в противен случай. Основният проблем с този подход е в избора на подходящия праг за бинаризацията. Ако той е установен твърде високо, то е възможно да изпуснем някои интересни асоциации, ако прагът е избран прекалено малък – възниква възможността за генериране на голямо количество фалшиви асоциации.

В настоящия раздел ще разгледаме методологията за намиране на асоциации между думите, известна като *min-Apriori* (Han et all. 1997). Подобно на традиционния асоциативен анализ едно артикулно множество се разглежда като колекция от думи, докато неговата поддръжка измерва степента на асоциацията между думите. Поддръжката на някое артикулно множество може да бъде изчислена на основата на нормализираната честота на срещане на съответните думи. Например, да разгледаме документ $d_1$ от Таблица 15.2. Нормализираните честоти на срещане на *Дума$_1$* и *Дума$_2$* в този документ са, съответно, 0.3 и 0.6. Някой може да помисли, че един разумен подход за изчисляване на асоциацията между тези думи е да се вземе средното аритметично на тези нормализирани стойности, т.е. (0.3 +0.6)/2 = 0.45. По този начин факторът за поддръжка на

съответното артикулно множество може да бъде изчислен чрез сумиране на средни нормализирани честоти на срещане по всички документи, т.е.

$$s(\{Дума\ , Дума_2\}) = \frac{0.3+0.6}{2} + \frac{0.1+0.2}{2} + \frac{0.4+0.2}{2} + \frac{0.2+0}{2} = 1$$

Този резултат в никаква степен не е случаен – тъй като честотата на срещане на всяка дума се нормализира до 1, осредняването на нормализираните честоти прави поддръжката на всяко артикулно множество да е равна на 1. Следователно, при този подход всички артикулни множества стават често срещани, което прави невъзможно намирането на интересни шаблони.

В min-Apriori асоциацията между думи в един документ се изчислява като минималната стойност на техните нормализирани честоти на срещане, т.е. $min(Дума_1, Дума_2) = min(0.3, 0.6) = 0.3$. Поддръжката на едно артикулно множество се изчислява чрез сумиране на неговите асоциации по всички документи.

$$s(\{Дума\ , Дума_2\}) = \min(0.3,0.6) + \min(0.1,0.2) + \min(0.4,0.2) + \min(0.2,0) = 0.6$$

Тази мярка за поддръжка, дефинирана в min-Apriori, има следните желателни свойства, които й правят подходяща за намиране на асоциации между думите н документи:

1. Поддръжката нараства монотонно, когато нормализираната честота на срещане на една дума нараства.
2. Поддръжката нараства монотонно, когато броят на документи, съдържащи думата, нараства.
3. Поддръжката има анти-монотонното свойство. Например, да разгледаме една двойка от артикулни множества *{A, B}* и *{A, B, C}*. Тъй като $min(\{A, B\}) \geq min(\{A, B, C\})$, то и $s(\{A, B\}) \geq s(\{A, B, C\})$. По този начин поддръжката намалява монотонно, когато броят на думите в едно артикулно множества нараства.

Стандартният алгоритъм Apriori може лесно да бъде модифициран за намиране на асоциации между думи с използване на новата дефиниция на поддръжката.