Милан Гњатовић

Увод у проналажење информација на вебу

Висока школа електротехнике и рачунарства струковних студија у Београду, 2017.

Наслов:	Увод у проналажење информација на вебу
Аутор:	др Милан Гњатовић
Редни број издања:	прво издање
Издавач:	Висока школа електротехнике и рачунарства струковних студија, Београд, Војводе Степе бр. 283
За издавача:	др Вера Петровић
Рецензенти:	др Перица Штрбац, др Бранимир Тренкић
Техничка обрада:	др Милан Гњатовић
Дизајн корица:	др Милан Гњатовић
ISBN:	978-86-7982-257-4
Штампа:	Развојно-истраживачки центар графичког инжењерства ТМФ, Београд
Тираж:	30

Место и година издања: Београд, 2017.

Наставно веће Високе школе електротехнике и рачунарства струковних студија у Београду је одобрило издавање ове публикације на седници одржаној 02.02.2017.

CIP - Каталогизација у публикацији - Народна библиотека Србије, Београд

004.738.52.021
001.102

ГЊАТОВИЋ, Милан, 1978-
Увод у проналажење информација на вебу / Милан Гњатовић. - 1. изд. - Београд : Висока школа електротехнике и рачунарства струковних студија, 2017 (Београд : Развојно-истраживачки центар графичког инжењерства ТМФ). - XVI, 103 стр. : илустр. ; 25 cm

Тираж 30. - Библиографија уз свако поглавље. - Регистар.

ISBN 978-86-7982-257-4

a) Интернет - Претраживање - Алгоритми b) Информације - Претраживање
COBISS.SR-ID 229610252

Својој породици, уз извињење због времена које сам провео пишући ову књигу, уместо да будем с њима.

Садржај

Листа слика

Листа табела

Предговор

Ова књига пружа концизни увид у изабране концепте, приступе, моделе и алгоритме релевантне за проналажење информација на вебу. Приликом избора тема које ће бити укључене у књигу, водио сам се следећим критеријумима: изабране теме су актуелне и широко заступљене у научноистраживачком раду и практичним применама, а комплексност и обим излагања су прикладни за уводни, једносеместрални курс на основним академским и струковним студијама техничких наука. Циљ књиге је да код читаоца развије разумевање актуелних приступа у овој области, и оспособи га да самостално и критички анализира њихове предности, ограничења и адекватност примене.

У поглављу 1 се разматрају основни појмови у области проналажења информација на вебу, уз осврт на оцењивање ефикасности система за проналажење информација.

У поглављу 2, читалац се упознаје са структуром података инвертовани индекс и буловском претрагом, који омогућавају брзо и флексибилно проналажење информација. Представљене су две рачунарске репрезентације инвертованог индекса (матрица инциденције термина и докумената, и инвертоване листе), а посебна пажња је посвећена њиховим предностима и недостацима.

У поглављу 3 се посматрају генерисање, представљање и претраживање речника индексних термина. У првом делу поглавља су објашњене основне фазе језичке обраде колекције докумената, које омогућавају препознавање речи у различитим појавним облицима (нпр., именице промењене по роду, броју и падежу, инфлективни облици глагола, различита писма, итд.). У другом делу поглавља се разматра проблем оптимизације претраге речника индексних термина.

Поглавље 4 је посвећено контекстно зависном предвиђању корисничких упита. Укратко су поновљени изабрани концепти теорије вероватноће који су релевантни за ово поглавље, и представљена је статистичка концептуализација вероватноће секвенце речи. Потом су представљени статистички језички модели, тзв. н-грами, који на основу неколико по-

следњих речи предвиђају наредну реч, тј., рачунају вероватноћу њеног јављања.

У поглављу 5 се разматра исправљање словних грешака у корисничким упитима. Проналажење прикладног термина из речника се заснива на два принципа. Први принцип је да се неисправни термин замени исправним термином који му је најсличнији. Други принцип је да систем бира индексни термин који је највероватнији у датом контексту. У оквиру првог принципа, изложен је и анализиран алгоритам за одређивање минималног растојања између стрингова, а у оквиру другог принципа је размотрен бајесовски приступ контекстно зависном исправљању словних грешака.

У поглављу 6 се посматра проблем рангирања резултата претраге у складу са њиховом релевантношћу. Укратко су поновљени релевантни појмови из теорије графова, а потом је представљен критеријум за рангирање веб-страница који се заснива на анализи структуре графа који представља скуп веб-страница и хипервезе између њих. Посебна пажња је посвећена особини пристрасности обиласка графа на случајни начин, јер се на њој заснива представљени критеријум за рангирање веб-страница.

У поглављу 7 се разматрају два методолошка приступа класификацији текста. Први приступ се заснива на Бајесовој теореми, и његова примена је илустрована за аутоматску детекцију нежељених електронских порука. Други приступ се заснива на биграмима на нивоу знакова, и његова примена ће бити илустрована за аутоматско идентификовање аутора текста.

Сва поглавља укључују примере, задатке за самостални рад и литературу. Разматрани приступи и алгоритми су независни од језика за који се примењују, а илустровани су примарно за српски језик. Основна знања из теорије вероватноће, линеарне алгебре, структура података и алгоритама могу бити од користи читаоцу, али нису предуслов.

Веб-страница

На адреси www.gnjatovic.info/pronalazenjeinformacija се налази веб-страница посвећена овој књизи. На овој страници ће, поред електронске верзије књиге, бити доступни и додатни материјали везани за књигу. Утисци, предлози и корекције су добродошли.

Поглавље 1
Увод

Израз „проналажење информација“ (енгл. *information retrieval*) није једнозначан. У различитим контекстима, овај израз се односи на проналажење текстуалних докумената, одговарање на питања, препознавање тема и концепата у тексту, сумаризовање текста, структурирање текста, претраживање мултимедијалних садржаја, итд. [1, 2]. У овој књизи се ограничавамо на следећу дефиницију проналажења информација — аутоматско проналажење неструктурираних докумената из дате колекције докумената чији садржај задовољава одређену информациону потребу [4]. Израз „колекција докумената“ се примарно односи на текстуалне садржаје доступне на вебу, а под „аутоматским проналажењем“ подразумевамо да претрагу врши софтвер, док је корисник само иницира постављањем упита.

Ова формулација посебно истиче неструктурираност докумената над којима се врши претрага, што захтева додатна појашњења. Под термином „документ“ подразумевамо *неструктурирани* текст. За податке се каже да су неструктурирани када немају јасно назначену семантичку структуру која је погодна за директну рачунарску обраду. Као пример структурираних података можемо посматрати табелу у релационој бази података (в. табелу 1.1). Сваком податку у табели је придружен атрибут (тј., тип података), чиме се дефинише значење података у посматраном контексту. За разлику од структуриране табеле, текст је неструктуриран. Строго говорећи, текст је такође структуриран на различитим нивоима, укључујући поделу текста на поглавља и пасусе, означене делове који одговарају насловима и поднасловима, синтаксну структуру на нивоу реченице, лексичко-синтактичку кохезију на нивоу текста, референце, итд. Али манифестације ових структура углавном нису прикладне за директну рачунарску обраду. Зато — у инжењерском контексту — за текстове кажемо да су неструктурирани или слабо структурирани (док за лингвистички контекст ово не важи).

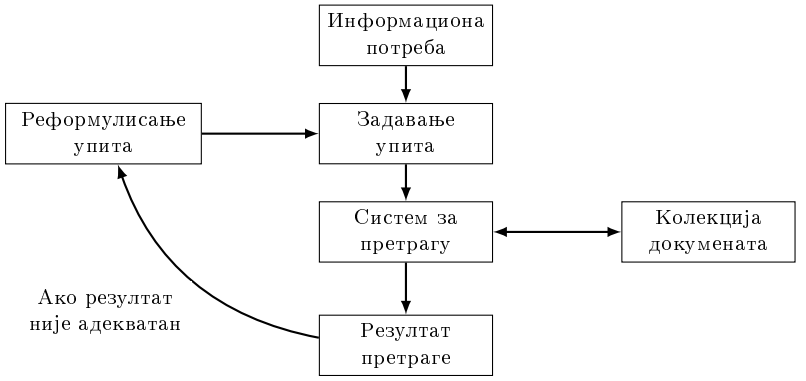
Иако је израз „проналажење информација“ широко прихваћен, важно је приметити његову терминолошку непрецизност. Системи за пронала-

Табела 1.1 Табела у релационој бази података као пример структурираних података.

Идентификатор	**Име**	**Презиме**	...
1001	Милош	Црњански	...
1002	Иво	Андрић	...
⋮	⋮	⋮	⋱

жење информација на вебу не пружају информације које би представљале одговор на информациону потребу корисника. У ствари, ови системи само пружају информацију да ли постоје и где се налазе документи који су некако повезани са корисничким упитом [3]. Ова функционалност се назива *ад хок* проналажењем информација, и њени различити аспекти ће бити разматрани у овој књизи [4].

Дијаграм тока претраге је представљен на сл. 1.1. Да би задовољио своју информациону потребу, корисник формулише упит. У поступку обраде корисничког упита, систем за проналажење информација селектује документе из дате колекције докумената за које процени да садрже информације релевантне за корисника. Уколико корисник није задовољан резултатом обраде упита, може да преформулише упут и иницира нову претрагу.



Слика 1.1 Дијаграм тока претраге.

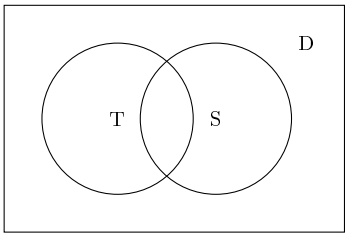
На дијаграму тока претраге је истакнуто да су информациона потреба корисника и кориснички упит различити појмови. Информациона потреба је везана за тему о којој корисник жели да сазна нешто, док кориснички упит представља начин да корисник формулише и пренесе своју информациону потребу систему. Уобичајени начин да се формулише упит је да се спецификује скуп термина који описују информациону потребу, а задатак система је да пронађе документа која садрже задате термине. Међутим, релевантност пронађених докумената се процењује у односу на

то да ли задовољавају информациону потребу корисника, а чињеница да неки документ садржи задати скуп термина није довољна да документ буде релевантан.

Пример 1.1. Посматрајмо кориснички упит који садржи само термин „лук“. Узимајући само овај термин у обзир, не може се закључити да ли се информациона потреба корисника односи на поврће (нпр., бели и црни лук), оружје (нпр., ловачки лук), мостове (нпр., лук на мосту), итд.

Оцењивање ефективности система за проналажење информација се врши на изабраној колекцији докумената $D = \{d_1, d_2, ..., d_n\}$, и изабраном скупу упита $Q = \{q_1, q_2, ..., q_k\}$ који формулишу одређене информационе потребе, при чему је за сваки пар (d_i, q_j) назначено да ли је документ $d_i \in D$ релевантан за информациону потребу формулисану упитом $q_j \in Q$.

Нека за дати упит q_j, скуп $T \subseteq D$ садржи документе који су релевантни за упит, а скуп $S \subseteq D$ садржи документе које је систем селектовао као релевантне за упит (в. сл. 1.2). Два основна параметра за оцењивање



Слика 1.2 Илустрација појмова прецизности и одзива (в. [5]).

ефективности обраде упита q_j су прецизност (енгл. *precision*) и одзив (енгл. *recall*), који се дефинишу на следећи начин [5]:

$$\text{прецизност: } P = \frac{\text{број селектованих релевантних докумената}}{\text{број селектованих докумената}} \times 100 = \frac{|T \cap S|}{|S|} \times 100\,, \tag{1.1}$$

$$\text{одзив: } R = \frac{\text{број селектованих релевантних докумената}}{\text{број релевантних докумената у колекцији}} \times 100 = \frac{|T \cap S|}{|T|} \times 100\,. \tag{1.2}$$

Ови параметри су често обрнуто пропорционални. За корисничке упите са строгим условом претраге се добијају висока прецизност и мали одзив. За корисничке упите са слабим условом претраге се добијају мала прецизност и велики одзив. У оваквим случајевима, контрадикторне вредности ових параметара нису адекватне за оцењивање система.

Пример 1.2. Претпоставимо да за дати упит постоји 10 релевантних докумената у колекцији, а да је систем селектовао 100000 докумената, укључујући и све релевантне документе. Прецизност је мала ($P = 0,01\%$), док је одзив максималан ($R = 100\%$).

Због тога је потребан параметар који узима у обзир и прецизност и одзив система. Ф-мера се дефинише на следећи начин [5]:

$$F = \frac{1}{\alpha \frac{1}{P} + (1-\alpha)\frac{1}{R}}, \tag{1.3}$$

где важи $\alpha \in [0,1]$. За $\alpha < \frac{1}{2}$, Ф-мера наглашава одзив, а за минималну вредност $\alpha = 0$, Ф-мера је једнака одзиву, тј., $F = R$. За $\alpha > \frac{1}{2}$, Ф-мера наглашава прецизност, а за максималну вредност $\alpha = 1$, Ф-мера је једнака прецизности, тј., $F = P$. За $\alpha = \frac{1}{2}$, Ф-мера представља хармонијску средину прецизности и одзива, $\frac{2PR}{P+R}$, и назива се *балансираном Ф-мером*. За дате вредности прецизности и одзива, балансирана Ф-мера увек нагиње мањој вредности.

Пример 1.3. У претходном примеру, вредности прецизности и одзива су биле: $P = 0,01\%$ и $R = 100\%$. Балансирана Ф-мера износи $F = \frac{2PR}{P+R} \approx 0,02\%$, тј., нагиње прецизности и представља адекватнију оцену система.

Избор вредности α зависи од тога чему желимо да дамо предност: прецизности или одзиву. „Рекреативни“ корисник система за претраживање веба жели да сви резултати на првој страници приказа буду релевантни, и нема намеру да прегледа све релевантне документе о теми коју претражује. За оваквог корисника, важнија је прецизност. „Професионални“ корисник система (попут научника, правника, аналитичара, итд.) жели да пронађе све релевантне документе, чак и ако то значи да ће морати да прегледа и ирелевантне документе. За оваквог корисника, важнији је одзив.

1.1 Задаци

Задатак 1.1. Претпоставите да за дати упит постоји 120 релевантних докумената у колекцији, и да је систем селектовао 60 докумената, од којих је 40 релевантно. Израчунајте прецизност, одзив и балансирану Ф-меру.

Задатак 1.2. Претпоставите да за дати упит постоји 80 релевантних докумената у колекцији, и да је систем селектовао 40 докумената, од којих је 30 релевантно. Израчунајте прецизност, одзив и Ф-меру за вредност параметра $\alpha = \frac{1}{4}$.

Задатак 1.3. Зашто је, у општем случају, балансирана Ф-мера адекватнија од аритметичке средине прецизности и одзива.

Литература

1. Allen J et al. (2003) Challenges in Information Retrieval and Language Modeling. In: Challenges in Information Retrieval and Language Modeling. SIGIR Forum, 37 (1). pp. 31–47.
2. Ceri S, Bozzon A, Brambilla M, Della Valle E, Fraternali P, Quarteroni S (2013) Web Information Retrieval, Springer-Verlag Berlin Heidelberg.
3. Lancaster FW (1968) Information Retrieval Systems: Characteristics, Testing and Evaluation, Wiley, New York.
4. Manning CD, Raghavan P, Schütze H (2008) Introduction to Information Retrieval, Cambridge University Press.
5. Manning C, Schuetze H (1999) Foundations of Statistical Natural Language Processing, The MIT Press, Cambridge, Massachusetts, London, England.

Поглавље 2
Инвертовани индекс и буловска претрага

Посматрајмо релативно једноставни проблем развоја софтверског система чији је задатак да утврди да ли дати документ садржи одређену реч. Интуитивно решење је да систем чита садржину документа реч по реч (тзв. линеарно претраживање, в. сл. 2.1), док не наиђе на дату реч или крај документа.

Слика 2.1 Илустрација линеарног претраживања документа по задатој речи.

Међутим, линеарно претраживање није прикладно за велике колекције докумената [4]. Најочигледнији недостатак овог приступа је што не омогућава брзу претрагу великих колекција. Тренутно доступне колекције докумената су редова величине милијарду (10^9) и билион (10^{12}) речи, и њих није могуће линеарно претражити у прихватљивом времену. Ово је озбиљни недостатак, јер је ефикасна обрада корисничких упита постала суштински захтев за системе за проналажење информација. Једна од основних одлика савремених система је да је време обраде корисничких упита толико кратко, да корисник има субјективни осећај да је одзив тренутан. Корисници подразумевају брзи одзив система, и не би требало очекивати да би добро прихватили систем са спорим одзивом.

Други недостатак линеарног претраживања великих колекција докумената је што није погодно за обраду упита са флексибилним условима претраге, нпр., када корисник тражи документа која садрже термине који се налазе на одређеној међусобној удаљености или у истој реченици. Коначно, трећи недостатак линеарног претраживања је што није погодно за рангирање пронађених докумената у односу на то колико су релевантни за корисника.

У овом поглављу ћемо размотрити:

- структуру података *инвертовани индекс* (енгл. *inverted index*), и
- *буловску претрагу* (енгл. *Boolean retrieval*) над инвертованим индексом,

који омогућавају брзу и флексибилну претрагу колекције докумената, док ће проблем рангирања пронађених докумената по релевантности бити размотран у поглављу 6. Да бисмо објаснили основну идеју инвертованог индекса, претпоставимо да је сваком документу у колекцији придружен скуп *индексних термина*[1] који описују његов садржај.

Пример 2.1. За потребе илустрације, у овом поглављу ћемо посматрати нереално малу колекцију која садржи само три документа, чије целобројне шифре ћемо означити са D_1, D_2 и D_3. Табела 2.1 садржи скуп индексних термина који описују ова три документа.

Табела 2.1 Индексни термини.

Шифра документа	Индексни термини
D_1	афлатоксин, млеко
D_2	млеко, анализа
D_3	млеко, влада, афлатоксин

Кардиналност везе између докумената и термина је $(M:N)$, тј., сваком документу може да се придружи више индексних термина, и сваки термин може да буде придружен описима више докумената. Уместо да за сваки документ евидентирамо придружене термине, за сваки термин можемо да евидентирамо документа која описује. Тако настаје инвертовани индекс.

Пример 2.2. Табела 2.2 представља „инвертовану“ табелу из примера 2.1, тј., колоне и врсте у табели 2.2 су замениле места у односу на изворну табелу 2.1. Из ове „инверзије“ потиче назив структуре података „инвертовани индекс“.

[1] У поглављу 3 ћемо размотрити како се генеришу индексни термини — засад само претпостављамо да су дати.

Математички прецизније би било рећи да табела 2.2 представља транспоновану, а не инвертовану табелу 2.1. Међутим, израз „инвертовани индекс“ је широко прихваћен, и његово значење се јасно разликује од значења израза „инверзна матрица“ (енгл. *inverse matrix*), па нема разлога за одступање од устаљене терминологије.

Табела 2.2 Основна идеја инвертованог индекса: за сваки термин се евидентирају документи које описује.

Индексни термин	Шифре докумената
афлатоксин	D_1, D_3
анализа	D_2
влада	D_3
млеко	D_1, D_2, D_3

У овом поглављу ћемо размотрити две рачунарске репрезентације инвертованог индекса: матрицу инциденције термина и докумената (в. секцију 2.1), и инвертоване листе (в. секцију 2.3).

2.1 Матрица инциденције термина и докумената

За дату колекцију докумената, матрица инциденције термина и докумената се генерише на следећи начин:

- сваком индексном термину се додељује једна врста,
- сваком документу се додељује једна колона,
- цифра 1 у ћелији на пресеку врсте t и колоне d значи да термин t описује документ шифре d; у супротном је вредност ћелије 0.

Притом, индексни термини представљени по врстама матрице и шифре докумената представљене по колонама матрице су сортирани растуће. Ово се ради због оптимизације претраге над матрицом инциденције, и о томе ће бити речи касније.

Пример 2.3. Табела 2.3 садржи матрицу инциденције термина и докумената која одговара колекцији докумената представљеној табелом 2.2.

Табела 2.3 Матрица инциденције термина и докумената која одговара колекцији докумената представљеној табелом 2.2.

шифре су сортиране растуће →

термини су сортирани растуће ↓

	D_1	D_2	D_3	...
афлатоксин	1	0	1	
анализа	0	1	0	
влада	0	0	1	
млеко	1	1	1	
⋮				

Пре него што наставимо даље, подсетимо се укратко основних Булових [1] функција. Функције конјункције ($\wedge$), дисјункције ($\vee$) и негације ($\neg$) над скупом $\{0,1\}$ су представљене у табели 2.4. Треба приметити да ове три функције чине генераторски скуп скупа Булових функција, што значи да се помоћу њих могу изразити све Булове функције. У наставку ћемо размотрити извршавање упита који садрже просте услове типа $x \wedge y$, $x \vee y$ и $\neg x$, а пошто ове три операције чине генераторски скуп, изложени принципи се могу генерализовати за упите који садрже логичке изразе произвољне комплексности.

Табела 2.4 Булове функције конјункције, дисјункције и негације.

конјункција

x	y	$x \wedge y$
0	0	0
0	1	0
1	0	0
1	1	1

дисјункција

x	y	$x \vee y$
0	0	0
0	1	1
1	0	1
1	1	1

негација

x	$\neg x$
0	1
1	0

Буловску претрагу [2, 3, 4] над матрицом инциденције ћемо илустровати кроз низ примера над матрицом датом у табели 2.3.

Пример 2.4. Претпоставимо следећи услов претраге: пронаћи документе који садрже термине „млеко“ и „афлатоксин“. Да би се овај упит обрадио, потребно је из матрице инциденције издвојити врсте које одговарају задатим терминима, а потом на њима применити операцију конјункције. Конјункцију бирамо због везника „и“ у поставци упита. Јединице у резултату означавају шифре пронађених докумената. У овом примеру, шифре пронађених докумената су D_1 и D_3 (в. сл. 2.2).

Пример 2.5. Претпоставимо следећи услов претраге: пронаћи документе који садрже термине „анализа“ или „влада“. Слично као у претходном

Слика 2.2 Буловска претрага над матрицом инциденције (слика уз пример 2.4).

	D_1	D_2	D_3
афлатоксин	1	0	1
млеко	1	1	1
∧ (конјункција)	1	0	1
	↓		↓
Резултат:	D_1		D_3

примеру, из матрице инциденције се издвоје врсте које одговарају задатим терминима. Због везника „<u>или</u>“ у поставци упита, над издвојеним врстама се примењује операција дисјункције. Шифре пронађених докумената су D_2 и D_3 (в. сл. 2.3).

Слика 2.3 Буловска претрага над матрицом инциденције (слика уз пример 2.5).

	D_1	D_2	D_3
анализа	0	1	0
влада	0	0	1
∨ (дисјункција)	0	1	1
		↓	↓
Резултат:		D_2	D_3

Пример 2.6. Претпоставимо следећи услов претраге: пронађи документе који <u>не</u> садрже термин „влада“. Из матрице инциденције се издвоји врста која одговара задатом термину, и на њој се примени операција негације. Овај избор је условљен негацијом глагола у поставци упита. У овом примеру, шифре пронађених докумената су D_1 и D_2 (в. сл. 2.4).

Слика 2.4 Буловска претрага над матрицом инциденције (слика уз пример 2.6).

	D_1	D_2	D_3
влада	0	0	1
¬ (негација)	1	1	0
	↓	↓	
Резултат:	D_1	D_2	

Пример 2.7. Претпоставимо следећи услов претраге: пронађи документе који садрже термине „млеко“ <u>и</u> „афлатоксин“, <u>и не</u> садрже термин „влада“. Поставка овог упита садржи три термина, и из ње се види да је потребно користити операције конјункције и негације (везник „<u>и</u>“ и негација глагола). Из матрице инциденције се издвоје врсте које одговарају задатим терминима, а обрада упита се врши на следећи начин:

- прво се на врсти која одговара термину „влада“ примени операција негације,

- а потом се над све три врсте примени операција конјункције.

Шифра пронађеног документа је D_1 (в. сл. 2.5).

Слика 2.5 Буловска претрага над матрицом инциденције (слика уз пример 2.7).

	D_1	D_2	D_3
афлатоксин	1	0	1
млеко	1	1	1
¬ влада	1	1	0
∧ (конјункција)	1	0	0
	↓		
Резултат:	D_1		

2.1.1 Предности и недостаци матрице инциденције

Главна предност буловске претраге матрице инциденције је ефикасност [5, 6]. Буловска претрага је знатно бржа од линеарне претраге колекције докумената, јер се операције извршавају над низовима битова, а не врши се директно претраживање текста. Ово, наравно, захтева да се колекција докумената претходно преслика у матрицу инциденције термина и докумената. Након тога, сваки пут када корисник зада упит, претрага се врши над матрицом инциденције, а не над самом колекцијом докумената. Када дође до промена у колекцији (нпр., прошири се новим документима), потребно је опет пресликати колекцију у матрицу инциденције. Важно је приметити да се пресликавање колекције докумената у матрицу не одвија у реалном времену (нпр., када корисник зада упит). Друга предност матрице инциденције је што се помоћу операција конјункције ($\wedge$), дисјункције ($\vee$) и негације ($\neg$) могу изразити све Булове функције. То омогућава обраду упита који садрже логичке изразе произвољне комплексности.

Озбиљни недостатак матрице инциденције ћемо илустровати на примеру [4].

Пример 2.8. Претпоставимо да колекција има милион докумената, од којих сваки има у просеку по 1000 речи, и да у колекцији постоји 500 000 различитих речи. Матрица инциденције термина и докумената која представља ову колекцију има 500 000 врста и милион колона, тј., $5 \cdot 10^{11}$ битова. Оволико велика матрица се не може сместити у рачунарску меморију.

За реалне колекције докумената, матрица инциденције је толико велика да се не може сместити у рачунарску меморију. Да бисмо превазишли овај недостатак, морамо да редукујемо величину матрице инциден-

ције. Пошто матрица садржи само нуле и јединице, не морамо да евидентирамо целу матрицу — довољно је да евидентирамо ћелије у којима се јавља једна цифра, а за ћелије које нису евидентиране ће се подразумевати да садрже другу цифру. Питање које се овде јавља је да ли да евидентирамо само ћелије које садрже нуле, или само ћелије које садрже јединице. У следећем примеру ћемо показати да је број јединица у матрици инциденције знатно мањи од броја нула, тако да се евидентирањем јединица остварује већа уштеда простора.

Пример 2.9. Матрица инциденције коју смо разматрали у претходном примеру садржи $5 \cdot 10^{11}$ ћелија. Колекција садржи милион (10^6) докумената, од којих сваки има у просеку по 1000 речи, па број јединица у матрици не може бити већи од милијарду (10^9). То значи да јединице чине максимално $0,2\%$ матрице у овом примеру, док нуле чине минимално $99,8\%$ матрице. Другим речима, редукована матрица инциденције која евидентира само јединице је бар 500 пута мања од почетне матрице инциденције.

По дефиницији матрице инциденције термина и докумената, сваком индексном термину се придружује низ битова (тј., низ нула и јединица).Ови низови су исте дужине, која одговара броју докумената у колекцији, што нам омогућава да примењујемо операције конјункције и дисјункције над њима. Када се изврши редуковање величине матрице, за сваки индексни термин се евидентирају само јединице. То има две импликације.

- У почетној матрици инциденције, редно место јединице у низу указује на документ на који се посматрана јединица односи. Кад из оваквог низа избацимо нуле, губимо информацију о претходним положајима јединица. Зато се у „редукованим“ низовима који се придружују индексним терминима свака јединица замењује шифром документа на који се односи. Тиме се спречава губитак информација о повезаности индексних термина и докумената (в. табелу 2.5).
- У општем случају, различити индексни термини описују различити број докумената, па су „редуковани“ низови придружени индексним терминима такође различитих дужина. За представљање структуре која садржи секвенце различитих дужина је погодније користити листе уместо низова, што ћемо размотрити у следећој секцији.

Пример 2.10. Табела 2.5 приказује комплетну матрицу инциденције и редуковану матрицу инциденције.

2.2 Осврт на низове и листе

Елементи низа су у меморији смештени секвенцијално, тј., један иза другог (в. сл. 2.6). Секвенцијално смештање омогућава директни при-

Табела 2.5 (а) Комплетна матрица инциденције термина и докумената; (б) редукована матрица инциденције термина и докумената.

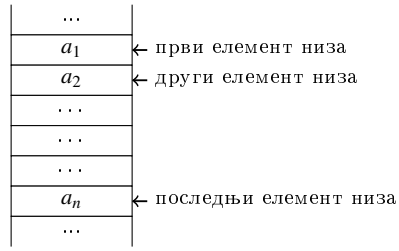
	D_1	D_2	D_3
афлатоксин	1	0	1
анализа	0	1	0
влада	0	0	1
млеко	1	1	1

(а)

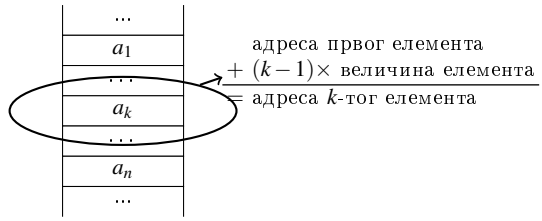
афлатоксин	D_1, D_3
анализа	D_2
влада	D_3
млеко	D_1, D_2, D_3

(б)

ступ елементима низа. Меморијска адреса k-тог елемента се израчунава на основу адресе првог елемента и индекса k (в. сл. 2.7). Приступање произвољном елементу низа је ефикасно, јер не захтева приступање осталим елементима низа.



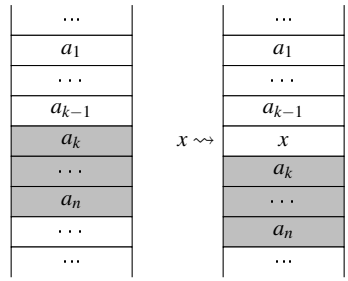
Слика 2.6 Меморијска репрезентација низа.



Слика 2.7 Директни приступ елементу низа.

Раније смо напоменули да су шифре докумената у низовима сортиране растуће, да би претрага над инвертованим индексом могла да се оптимизује. За разлику од приступања елементима низа, додавање елемента у сортирани низ и уклањање елемента из сортираног низа су временски захтевни. Нпр., додавање новог елемента x испред k-тог елемента захтева да се сваки елемент подниза $a_k, a_{k+1}, ..., a_n$ премести на меморијску локацију елемента који му непосредно следи, при чему се прво премешта елемент a_n, па a_{n-1}, итд., а на крају a_k (в. сл. 2.8). Уклањање елемента из низа захтева сличну операцију. Осим тога, низови увек имају задату дужину (тј., број елемената), што представља недостатак у посматра-

ном контексту, јер је број докумената које описује неки индексни термин променљив. Ови недостаци се могу превазићи употребом листи.

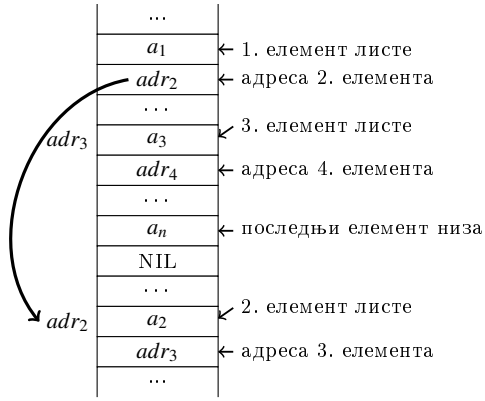


Слика 2.8 Додавање елемента у сортирани низ.

На концептуалном нивоу, елементи листе су такође поређани секвенцијално, тј., један иза другог. Међутим, физичка репрезентација листе у меморији не подразумева секвенцијално смештање елемената. Уместо тога, сваки елемент једноструко уланчане листе има два дела:

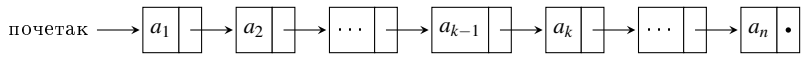
- вредност елемента,
- меморијску адресу наредног елемента, при чему последњи елемент листе у овом делу има нулту вредност коју ћемо означавати NIL или тачком (•).

Меморијска репрезентација листи је илустрована на сл. 2.9, а уобичајена графичка репрезентација једноструко уланчане листе је дата на сл. 2.10.



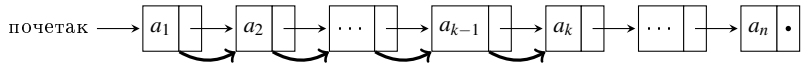
Слика 2.9 Меморијска репрезентација листе.

За разлику од низова, елементима једноструко уланчане листе се, у општем случају, не може приступити директно. Нпр., ако желимо да приступимо k-том елементу, морамо да:



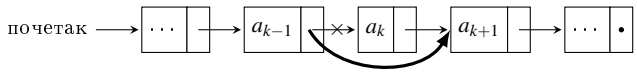
Слика 2.10 Уобичајена графичка репрезентација једноструко уланчане листе.

- приступимо 1. елементу, и прочитамо адресу 2. елемента,
- онда приступимо 2. елементу, и прочитамо адресу 3. елемента,
- итд., све до k-тог елемента (в. сл. 2.11).



Слика 2.11 Приступ елементима листе.

Ово је временски захтевно, и представља недостатак листи у односу на низове. Међутим, додавање елемента у сортирану листу и уклањање елемента из сортиране листе су ефикаснији него код низова. Нпр., ако желимо да уклонимо k-ти елемент листе, довољно је да елементу на месту $k-1$ као адресу следећег елемента упишемо адресу елемента на месту $k+1$ (в. сл. 2.12). Додавање елемента у листу захтева сличну операцију.

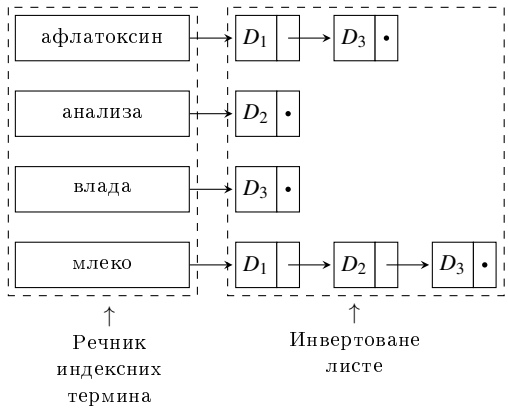


Слика 2.12 Уклањање елемента из листе.

2.3 Инвертоване листе

Вратимо се сад на проблем превелике матрице инциденције термина и докумената. Већ смо илустровали да је за реалне колекције докумената матрица инциденције толико велика да се не може сместити у рачунарску меморију. Њена величина се редукује тако што се евидентирају само оне ћелије у матрици у којима се јављају јединице, а за представљање редуковане матрице се користе једноструко уланчане листе.

Пример 2.11. Представимо редуковану матрицу инциденције, дату у табели 2.5(б), користећи листе. Резултујућа структура података, чији је графички приказ дат на слици 2.13, је инвертовани индекс. Леви део овог приказа, који садржи термине „афлатоксин“, „анализа“ итд., представља *речник индексних термина* и биће детаљније размотрен у поглављу 3. Десни део приказа називамо *инвертованим листама*.



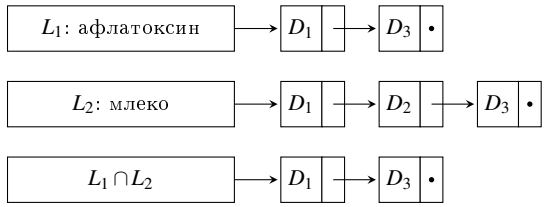
Слика 2.13 Инвертовани индекс.

За буловску претрагу над матрицом инциденције термина и докумената смо примењивали операције конјункције, дисјункције и негације. Над инвертованим листама ћемо примењивати операције пресека, уније и разлике скупова. Везе између ових операција су представљене у табели 2.6, а објаснићемо их кроз примере који следе.

Табела 2.6 Везе између логичких и скуповних операција у контексту претраживања инвертованог индекса.

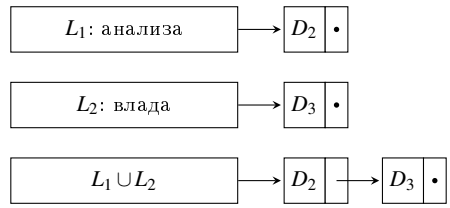
Матрица инциденције		Инвертоване листе
конјункција ($\wedge$)	$\Longleftrightarrow$	пресек ($\cap$)
дисјункција ($\vee$)	$\Longleftrightarrow$	унија ($\cup$)
негација ($\neg$)	$\Longleftrightarrow$	разлика ($\setminus$)

Пример 2.12. Претпоставимо следећи услов претраге: пронаћи документе који садрже термине „млеко“ и „афлатоксин“. Слично као у примеру 2.4, из инвертованог индекса се издвоје листе које одговарају задатим терминима, и на њима се примени операција пресека (в. сл. 2.14). Операцију пресека бирамо због везника „и“ у поставци упита. Пресек ове две листе садржи шифре пронађених докумената.



Слика 2.14 Буловска претрага над инвертованим индексом (слика уз пример 2.12).

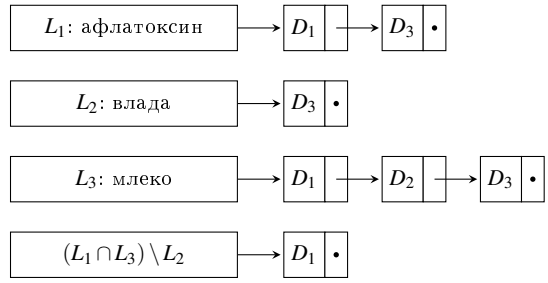
Пример 2.13. Претпоставимо следећи услов претраге: пронаћи документе који садрже термине „анализа“ или „влада“. Слично као у примеру 2.5, из инвертованог индекса се издвоје листе које одговарају задатим терминима, и над њима се примени операција уније. Операцију уније бирамо због везника „или“ у поставци упита. Шифре пронађених докумената су садржане у резултујућој листи (в. сл. 2.15).



Слика 2.15 Буловска претрага над инвертованим индексом (слика уз пример 2.13).

Пример 2.14. Претпоставимо следећи услов претраге: пронаћи документе који садрже термине „млеко“ и „афлатоксин“, и не садрже термин „влада“. Слично као у примеру 2.7, из инвертованог индекса се издвоје листе које одговарају задатим терминима. Означимо листе које одговарају терминима „афлатоксин“, „влада“ и „млеко“ са L_1, L_2, L_3, респективно. Обрада упита се врши на следећи начин:

- Прво се израчуна пресек листи L_1 и L_3. Пресек ових листи $L' = L_1 \cap L_3$ чине шифре докумената који садрже термине „афлатоксин“ и „млеко“.
- Потом се израчуна разлика листи $L' \setminus L_2$. Резултујућа листа садржи шифру пронађеног документа (в. сл. 2.16).



Слика 2.16 Буловска претрага над инвертованим индексом (слика уз пример 2.14).

На почетку овог поглавља смо истакли да је брзи одзив на кориснички упит суштински захтев за системе за претрагу. Зато системи не претражују директно колекцију докумената, већ се претрага врши над инвертованим индексом, што је знатно убрзава. Да бисмо додатно скратили време обраде корисничког упита, можемо да оптимизујемо још два аспекта претраге инвертованог индекса:

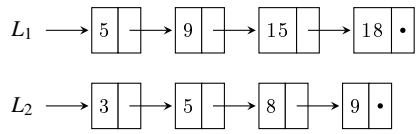
- проналажење термина из корисничког упита у речнику индексних термина, што ћемо размотрити у поглављу 3,
- извршавање операција пресека, уније и разлике листи, што разматрамо у наставку овог поглавља.

За шифре докумената можемо, без губитка општости, претпоставити да су целобројне вредности. Приликом дефинисања инвертованог индекса смо усвојили правило да су инвертоване листе сортиране растуће, јер се скуповне операције извршавају брже над сортираним листама. Да бисмо ово илустровали, анализирајмо алгоритам за пресек две листе. Када листе не би биле сортиране, за сваки елемент прве листе би требало проћи кроз другу листу и проверити да ли се налази у њој. Временска комплексност оваквог алгоритма је $O(mn)$, где су m и n дужине листи над којима се извршава операција пресека. Пошто су листе сортиране, можемо да дефинишемо ефикаснији алгоритам, временске комплексности $O(m+n)$, који је представљен на сл. 2.17 [4].

```
PRESEK(p1, p2)
  rezultat ← ⟨⟩
  while p1 ≠ NIL and p2 ≠ NIL
    if docID(p1) = docID(p2)
      ADD(rezultat, docID(p1))
      p1 ← next(p1)
      p2 ← next(p2)
    else
      if docID(p1) < docID(p2)
        p1 ← next(p1)
      else
        p2 ← next(p2)
  return rezultat
```

Слика 2.17 Алгоритам за пресек две сортиране листе [4].

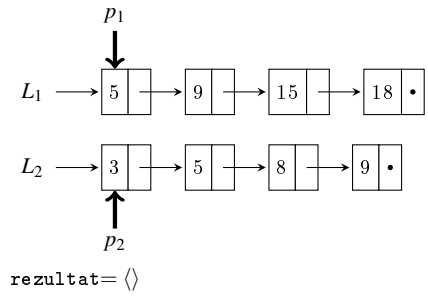
Пример 2.15. Илуструјмо алгоритам дат на сл. 2.17 за две листе приказане на сл. 2.18.



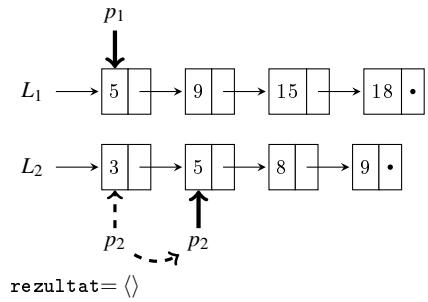
Слика 2.18 Улазни аргументи за алгоритам за пресек две сортиране листе.

На почетку, p_1 и p_2 показују на прве елементе листи L_1 и L_2, а резултујућа листа је празна (в. сл. 2.19). Вредности на које показују p_1 и p_2 нису једнаке ($5 \neq 3$), па се у првој итерацији алгоритма показивач на мању

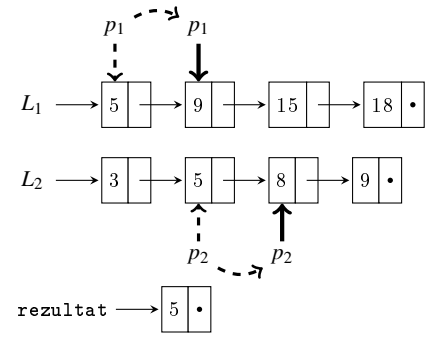
вредност (а то је тренутно p_2) премешта на наредни елемент у листи (в. сл. 2.20). У другој итерацији, вредности на које показују p_1 и p_2 су једнаке ($5 = 5$). Зато се вредност на коју указују додаје у резултујућу листу, а p_1 и p_2 се премештају на наредне елементе у листама (в. сл. 2.21).



Слика 2.19 Почетно стање извршавања алгоритма за пресек две сортиране листе.

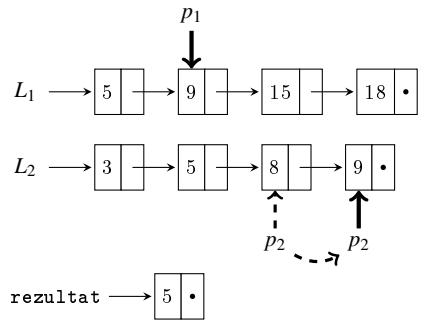


Слика 2.20 Прва итерација алгоритма за пресек две сортиране листе.



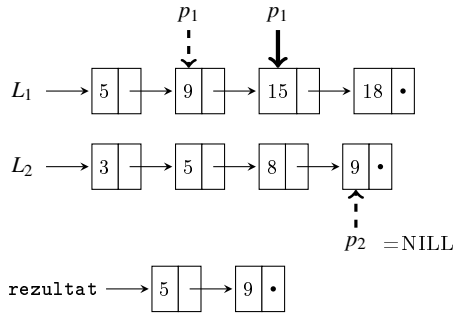
Слика 2.21 Друга итерација алгоритма за пресек две сортиране листе.

У трећој итерацији алгоритма, вредности на које тренутно показују p_1 и p_2 опет нису једнаке ($9 \neq 8$), па се показивач на мању вредност (а то је опет p_2) премешта на наредни елемент у листи (в. сл. 2.22). У



Слика 2.22 Трећа итерација алгоритма за пресек две сортиране листе.

четвртој итерацији, вредности на које показују p_1 и p_2 су једнаке ($9 = 9$). Зато се вредност на коју указују додаје у резултујућу листу, а p_1 и p_2 се премештају на наредне елементе у листама (в. сл. 2.23). Тиме смо стигли до краја друге листе ($p_2 = \text{NIL}$), чиме се извршавање алгоритма завршава. Приметите да прву листу нисмо прочитали до краја, нити је то било потребно (в. задатак 2.4). На слични начин можемо да дефинишемо и алгоритме за унију и разлику две сортиране листе (в. задатке 2.5 и 2.6).



Слика 2.23 Четврта итерација алгоритма за пресек две сортиране листе.

2.3.1 Предности и недостаци операција над инвертованим листама

Предност сортираних листи је што се скуповне операције над њима брже извршавају. Након анализе алгоритма за пресек две сортиране листе датог на сл. 2.17, види се да је резултат овог алгоритма такође сортирана листа. Исто би важило и за алгоритме за унију и разлику две листе. Ово је неопходно да бисмо могли да вршимо надовезивање скуповних операција, тј., да излазна листа из једног алгоритма буде улазни аргумент за други алгоритам.

Пример 2.16. Посматрајмо опет пример 2.14 који претпоставља следећи услов претраге: пронаћи документе који садрже термине „млеко“ и „афлатоксин“, и не садрже термин „влада“. На концептуалном нивоу, видели смо да се резултујућа листа може израчунати на следећи начин:

$$(L_1 \cap L_3) \setminus L_2 \ ,$$

где су L_1, L_2, L_3 листе које одговарају терминима „афлатоксин“, „влада“ и „млеко“, респективно. На имплементационом нивоу, резултујућа листа се може израчунати угнежђеним позивом фукције:

$$\mathit{RAZLIKA}(\mathit{PRESEK}(L_1, L_3), L_2) \ ,$$

тј., прво се израчуна пресек листи L_1 и L_3, а потом резултат тог пресека постаје први улазни аргумент за алгоритам за израчунавање разлике листи. Међутим, оваква имплементација захтева да резултат извршавања алгоритма за пресек листи такође буде сортирана листа. У општем случају, да бисмо могли да надовезујемо операције над листама, неопходно је да резултати свих појединачних операција буду сортирани.

У оваквој имплементацији, крајњи резултат обраде упита, који се приказује кориснику, је такође сортирана листа. Озбиљни недостатак се огледа у томе што критеријум сортирања резултујуће листе нема везе са релевантношћу докумената. Уместо тога, кориснику треба приказати резултате који су сортирани према релевантности, тј., прво најрелевантније, а потом мање релевантне. Превазилажење овог недостатка ћемо размотрити у поглављу 6.

2.4 Услови удаљености

Индексни термин може да се јави на више места у истом документу. Структура инвертованог индекса коју смо до сада посматрали не укључује податке о броју јављања датог термина у документу, нити евидентирала локације у документу на којима се термин јавља. Стога, ова структура није адекватна за обраду упита који садрже услов да се термини налазе на одређеној међусобној удаљености.

Пример 2.17. Претпоставимо следећи услов претраге: пронаћи документе који садрже секвенцу речи „Нови Београд“. Јасно је да се овај услов не може свести на слабији услов: пронаћи документе који садрже термине „Нови“ и „Београд“, јер би систем тада селектовао документе који садрже

ове термине на било којим местима у тексту, нпр., „У Београд је стигао нови[2] . . . “.

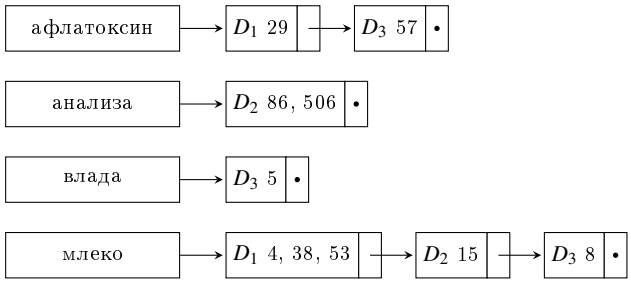
У општем случају, кориснику би требало омогућити да приликом задавања упита дефинише услове удаљености између два или више термина. Удаљеност између термина се може спецификовати на више начина, нпр.:

- термини су суседни или на одређеној међусобној удаљености у тексту,
- термини се налазе у истој реченици, итд.

Да бисмо могли да обрадимо упите који садрже услове удаљености, инвертовани индекс морамо да проширимо подацима о локацијама термина у документима.

Пример 2.18. Претпоставимо да желимо да омогућимо кориснику да спецификује услов да су термини суседни или на одређеној међусобној удаљености. Инвертовани индекс проширујемо подацима о локацијама термина у документу. Нпр., проширени инвертовани индекс приказан на сл. 2.24 укључује следеће податке:

- термин „анализа“ се јавља два пута у документу D_2, на местима 86. и 506.,
- термин „млеко“ се јавља три пута у документу D_1, на местима 4., 38. и 53., и по једном у документима D_2 (на месту 15.) и D_3 (на месту 8.), итд.



Слика 2.24 Инвертовани индекс проширен подацима о локацијама термина у документу.

Услов претраге из примера 2.17 се сада може формулисати на следећи начин: пронаћи документе који садрже термине „нови“ и „Београд“, при чему се термин „нови“ налази непосредно испред термина „Београд“.

[2] Овде смо придев „нови“ написали малим почетним словом, иако је у упиту написан великим. У поглављу 3 (секција 3.1.3) ћемо видети зашто је оправдано да не узимамо у обзир величину слова приликом обраде упита.

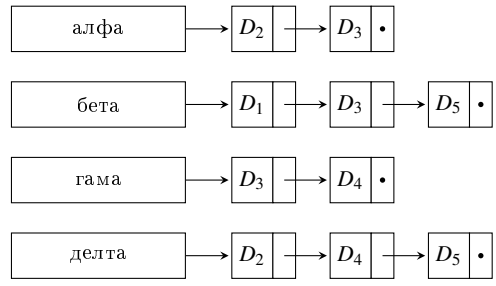
Подаци о локацији термина се могу односити и на редни број реченице у којој се термин јавља, или редни број пасуса, итд. У општем случају, подаци о локацији термина рефлектују концептуализацију услова удаљености, која зависи од конкретне спецификације система за проналажење информација. Притом, основна идеја проширења инвертованог индекса остаје иста.

Функционалност унапређене претраге долази са извесном ценом. Прво, временска комплексност алгоритма за пресек листи се повећава, јер се не упоређују само шифре докумената, већ и позиције термина у документима (в. задатак 2.7). Друго, величина инвертованог индекса се знатно повећава. Повећање величине инвертованог индекса зависи од типа докумената у колекцији, и језика на ком су написани, али се као смерница може узети претпоставка да се увођењем података о локацијама термина инвертовани индекс увећава два до четири пута [4].

2.5 Задаци

Задатак 2.1. За инвертовани индекс приказан на сл. 2.25, објасните обраду следећих упита:

- Пронаћи све документе који садрже термине „алфа“ и „гама“.
- Пронаћи све документе који садрже термин „бета“ и не садрже термин „алфа“.
- Пронаћи све документе који садрже термине „бета“ или „делта“, и не садрже термин „алфа“.



Слика 2.25 Инвертовани индекс, слика уз задатак 2.1.

Задатак 2.2. Алгоритам за пресек сортираних листи приказан на сл. 2.17 укључује две претпоставке: улазни аргументи су сортиране листе, и улазни аргументи су листе у којима су сви елементи различити (тј., нема понављања елемената у листи). Да ли је друга претпоставка оправдана у контексту претраге над инвертованим индексом?

Задатак 2.3. Покажите да је временска комплексност алгоритма за пресек сортираних листи, датог на сл. 2.17, једнака $O(m+n)$, где су m и n дужине листи.

Задатак 2.4. Алгоритам за пресек сортираних листи дат на сл. 2.17 се извршава док се не дође до краја једне од улазних листи (в. пример 2.15). Објасните зашто у општем случају није потребно прочитати обе листе до краја.

Задатак 2.5. Дефинишите алгоритам за унију сортираних листи.

Задатак 2.6. Дефинишите алгоритам за разлику сортираних листи.

Задатак 2.7. Дефинишите алгоритам за пресек сортираних листи које садрже и податке о локацијама термина у документима.

Литература

1. Boole G (1854) An Investigation of the Laws of Thought on Which are Founded the Mathematical Theories of Logic and Probabilities. Macmillan and Co. Reissued by Cambridge University Press, 2009.
2. Ceri S, Bozzon A, Brambilla M, Della Valle E, Fraternali P, Quarteroni S (2013) Web Information Retrieval, Springer-Verlag Berlin Heidelberg.
3. Lancaster FW, Fayen EG (1973) Information Retrieval On-Line, Melville, New York.
4. Manning CD, Raghavan P, Schütze H (2008) Introduction to Information Retrieval, Cambridge University Press.
5. Witten IH, Moffat A, and Bell TC (1999) Managing Gigabytes: Compressing and Indexing Documents and Images, 2nd edition, Morgan Kaufmann.
6. Zobel J, Moffat A (2006) Inverted files for text search engines, ACM Computing Surveys 38(2).

Поглавље 3
Речник индексних термина

У овом поглављу разматрамо аспекте конструкције инвертованог индекса који су везани за речник индексних термина. Следећа питања су нам од интереса:

- Како се генерише речник индексних термина? У претходном поглављу смо подразумевали да је сваком документу у инвертованом индексу придружен скуп индексних термина који га описују. У овом поглављу објашњавамо поступак њиховог генерисања.
- Како оспособити систем да препозна термин у различитим појавним облицима (нпр., именице које су промењене по роду, броју и падежу, инфлективни облици глагола, различита писма, итд.)? Када систему за проналажење информација на вебу задамо упит који садржи неки термин, тада не очекујемо да систем пронађе само документе који садрже задати појавни облик тог термина, већ било који његов појавни облик. Нпр., ако Гуглу задамо упит „софтвери“ (именица у номинативу множине, написана на ћирилици), систем ће пронаћи документе који садрже именицу „софтвер“ у различитим појавним облицима: „софтвер“ (у једнини), „софтвера“ (генитив) „softver“ (на латиници), итд.
- Како се оптимизује претрага речника индексних термина? Већ смо истакли важност претраге инвертованог индекса у реалном времену. Оптимизација скуповних операција над инвертованим листама коју смо разматрали у претходном поглављу представља само део решења. У овом поглављу разматрамо проблем оптимизације претраге речника индексних термина.

3.1 Језичка обрада колекције докумената

Након читања колекције докумената, извршава се језичка обрада текста која резултује генерисањем речника индексних термина. Основне фазе језичке обраде текста су [2]:

- токенизација,
- избацивање честих речи,
- нормализација,
- лематизација.

3.1.1 Токенизација

Токенизација подразумева дељење текста на токене, тј., низове знакова који представљају семантичке јединице које су погодне за даљу обраду. Интуитивно, токенизација се може посматрати као дељење текста на појединачне речи, при чему се занемарују знакови интерпункције.

Пример 3.1. Нека је део изворног текста у документу:

„Danas stižu rezultati analize mleka i proizvoda od mleka.“

Резултат токенизације је следећи низ речи:

Danas	stižu	rezultati	analize	mleka	i	proizvoda	od	mleka

У општем случају, токенизација се не може свести на проблем издвајања речи које су раздвојене празним простором (белинама) или знаковима интерпункције (нпр., тачком, зарезом, итд.). Размотримо следеће примере, карактеристичне за српски језик.

Пример 3.2. У словним скраћеницама (нпр., „д.о.о.“), датумима (нпр., „1.1.2017.“), записима времена (нпр., „17.45“) и Интернет-адресама (нпр., „viser.edu.rs“), тачке не раздвајају засебне токене. Слично важи и за празни простор у именима градова (нпр., „Нови Сад“), записима бројева (нпр., „1 000 000“) и телефонских бројева („+381 63 123 4567“).

Правила за токенизацију текста у великој мери зависе од језика. У немачком језику је честа употреба сложених именица које се пишу без белина, док се у кинеском језику белине између речи у потпуности изостављају. Међутим, чак и кад бисмо узели у обзир сва правила интерпункције и правописа у неком језику, проблем токенизације не би био у потпуности решен. Садржаји на вебу нису контролисани, и њихов квалитет варира у великој мери. Дизајнери система за проналажење информација не смеју да очекују да ће садржаји докумената бити интерпункцијски и правописно исправни. Због тога се за токенизацију користе различити приступи — од унапред припремљених речника, до статистичких метода машинског учења [2].

3.1.2 Избацивање честих речи

Дизајнери система за проналажење информација могу да одлуче да речи које се често јављају у документима (попут везника) не укључе у речник индексних термина. За овакве речи се користи термин — *стоп-речи*. Главни разлог за овакву одлуку лежи у претпоставци да реч која се често јавља носи малу информациону вредност, па није од значаја за претрагу. Објаснимо ово на примеру.

Пример 3.3. У игри „Вешала“ имате ограничени број покушаја да погодите реч коју је замислио други учесник. Пробајте да погодите следећу реч:

_ _ _ А _ _ А.

Задатак није сасвим тривијалан, јер постоји више речи које се уклапају у овај шаблон. Да је задати шаблон гласио:

Љ _ _ _ Ш _ _,

већина матерњих говорника српског језика би брзо погодила да је тражена реч — љуљашка. Иако је у оба шаблона задата иста реч, други шаблон је знатно лакши. Разлог је следећи: у српском језику, сугласници дати у другом шаблону се јављају знатно ређе од самогласника датих у првом шаблону. Интуитивно је јасно да слова која се јављају ретко носе већу информациону вредност него слова која се јављају често. Слично важи и за речи, и зато је оправдано да речи које се јављају често не укључимо у речник индексних термина.

Поступак за детекцију честих речи је релативно једноставан. За све речи у колекцији докумената се израчунају учесталости јављања, и подскуп најучесталијих речи се искључи из речника. Једна од предности избацивања честих речи из речника индексних термина је што се тиме смањује и величина инвертованог индекса, односно убрзава се процес претраге. Са друге стране, занемаривање стоп-речи може да онемогући претраживање колекције докумената по фразама. Нпр., претпоставимо да смо предлог „од“ избацили из речника индексних термина. Кориснички упит који садржи синтагму „мали од палубе“ (заједно са предлогом) је прецизнији од упита који садржи само термине „мали“ и „палуба“. У овом случају, избацивање предлога „од“ би резултовало мањом прецизношћу. Због овога, употреба стоп-речи у системима за проналажење информација варира — неки системи уопште не користе стоп-речи, неки користе мали број стоп-речи (7–12 термина), док неки системи користе велики број стоп-речи (200–300 термина) [2].

Пример 3.4. У примеру 3.1, резултат токенизације је био:

Danas	stižu	rezultati	analize	mleka	i	proizvoda	od	mleka

Под претпоставком да везник „и“ и предлог „од“ третирамо као стоп-речи, резултат избацивања честих речи је:

Danas	stižu	rezultati	analize	mleka	proizvoda	mleka

3.1.3 Нормализација и лематизација

Неки токени се међусобно само површински разликују, и приликом претраге желимо да их третирамо као да су исти. Нпр.:

- словна скраћеница за фразу „друштво са ограниченом одговорношћу“ може да се пише на два начина: „д.о.о.“ и „доо“,
- речи могу да буду написане различитим писмима („млеко“ и „mleko“),
- речи могу да садрже слова различитих величина („Млеко“ и „млеко“).

Нормализација подразумева процес изједначавања токена који се само површински разликују. Уклањање површинских разлика може да укључује превођење токена на исто писмо, коришћење само малих слова за све токене, уједначавање словних скраћеница, итд.

У примеру 2.17 (у поглављу 2) смо наговестили да се величина слова не узима у обзир приликом обраде упита. Сада видимо да је то последица нормализације токена.

Пример 3.5. У примеру 3.4, резултат фазе избацивања честих речи је био:

Danas	stižu	rezultati	analize	mleka	proizvoda	mleka

Након нормализације ових токена добијамо:

данас	стижу	резултати	анализе	млека	производа	млека

Поступак нормализације токена није искључиво условљен правописним или граматичким разлозима. Незанемарљиви број корисника који користе латиницу за унос упита на српском језику често не користе слова са дијакритичким знаковима — нпр., уместо слова „č“, „ž“, „š“, ови корисници уносе слова „c“, „z“, „s“, респективно. Без обзира на разлоге за овакво понашање корисника (навика, лењост или незнање), дизајнери система не смеју да инсистирају да корисници користе дијакритичке знакове. Уместо тога, функционалност система за проналажење информација треба да укључује и обраду упита који не садрже дијакритичке знакове. Да би се ово постигло, нормализација токена може да подразумева и занемаривање дијакритичких знакова, иако њихово уклањање може да промени значење речи (упоредите „šišanje“ и „sisanje“) [2].

Неки токени се не разликују само површински, већ представљају различите појавне облике исте речи. Нпр., токени „човеку“ и „људи“ представљају облике именице „човек“, и приликом претраге желимо да их третирамо као да су исти. Због тога, језичка обрада текста укључује и фазу лематизације, која подразумева свођење изведених облика речи на основне облике (тзв. леме). У поступку лематизације, именице се своде на номинатив једнине („софтвери“ → „софтвер“ „пса“ → „пас“), глаголи на инфинитив („идем“ → „ићи“), итд.

Пример 3.6. У примеру 3.5, резултат нормализације токена је:

данас | стижу | резултати | анализе | млека | производа | млека

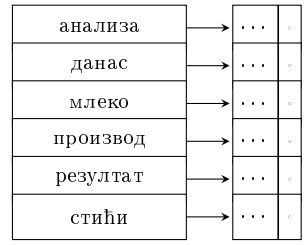
Након лематизације, добијамо следеће индексне термине:

данас | стићи | резултат | анализа | млеко | производ | млеко

Фазе језичке обраде описане у примерима 3.1–3.6 су сумиране у табели 3.1. Од резултујућих лема се генерише речник индексних термина, илустрован на сл. 3.1.

Табела 3.1 Фазе језичке обраде колекције докумената и корисничких упита.

Изворни текст:	„Danas stižu rezultati analize mleka i proizvoda od mleka.“
Токенизација:	Danas | stižu | rezultati | analize | mleka | i | proizvoda | od | mleka
Изб. честих речи:	Danas | stižu | rezultati | analize | mleka | proizvoda | mleka
Нормализација:	данас | стижу | резултати | анализе | млека | производа | млека
Лематизација:	данас | стићи | резултат | анализа | млеко | производ | млеко



Слика 3.1 Речник индексних термина генерисан након језичке обраде представљене у табели 3.1.

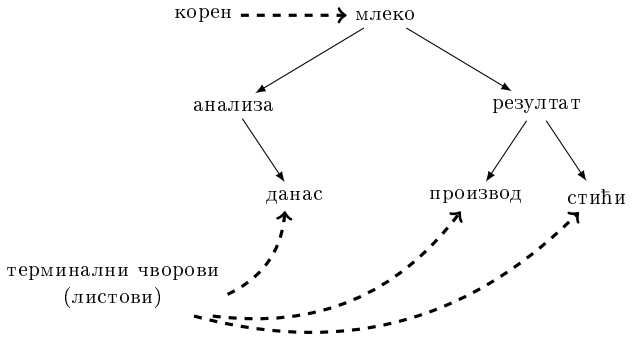
Треба приметити да кориснички упити пролазе исте фазе језичке обраде. Идеја је да се задате речи из упита сведу на леме, тако да могу да се упаре са постојећим индексним терминима у речнику. Нпр., претпоставимо да корисник задаје упит: *анализе млека у Србији*. Након језичке

обраде упита добијамо леме — *анализа*, *млеко*, *србија* — које се користе за претрагу. Резултат претраге чине документи који садрже речи *анализа*, *млеко* и *Србија* у било ком појавном облику.

3.2 Представљање речника индексних термина бинарним стаблом

Након језичке обраде корисничког упита, задатак система за претрагу је да утврди да ли се леме изведене из упита налазе у речнику индексних термина и, ако се налазе, идентификује показиваче на одговарајуће листе шифри докумената. Слично као што су листе шифри докумената у инвертованом индексу сортиране да бисмо могли да оптимизујемо скуповне операције над листама (в. поглавље 2), речник индексних термина се представља структуром података која омогућава оптимизацију његове претраге. У овом поглављу ћемо размотрити представљање речника индексних термина сортираним бинарним стаблом. Сваки чвор сортираног бинарног стабла има највише два детета, при чему је вредност левог детета мања од вредности посматраног чвора, док је вредност десног детета већа од вредности посматраног чвора. Сваки чвор, осим корена, има тачно једног родитеља, а чворови без деце се називају терминалним чворовима (или листовима).

Пример 3.7. Један од начина да се речник индексних термина приказан на сл. 3.1 представи бинарним стаблом је дат на сл. 3.2.



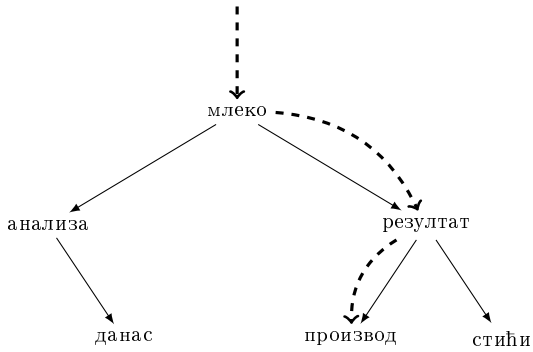
Слика 3.2 Уравнотежено сортирано бинарно стабло које представља речник индексних термина приказан на сл. 3.1.

Претрага сортираног бинарног стабла по задатом термину се може дефинисати рекурзивно, на следећи начин:

- Ако је стабло празно, претрага се завршава — задати термин се не налази у речнику.
- Ако је термин по коме се врши претрага једнак термину у корену стабла, претрага се завршава — задати термин је пронађен.
- Ако је задати термин мањи[1] од термина у корену стабла, претрага се рекурзивно наставља за подстабло одређено левим дететом текућег корена.
- Ако је задати термин већи од термина у корену стабла, претрага се рекурзивно наставља за подстабло одређено десним дететом текућег корена.

Претрага увек креће од корена стабла, и одвија се по путањи до чвора који садржи задати термин или до једног од терминалних чворова (уколико задати термин не постоји у стаблу).

Пример 3.8. Претрага речника по терминима „производ“, односно „афлатоксин“, је илустрована на сл. 3.3 и 3.4. Путања претраге (тј., транзиција између чворова који су били предмет претраге) је представљена испрекиданим линијама.

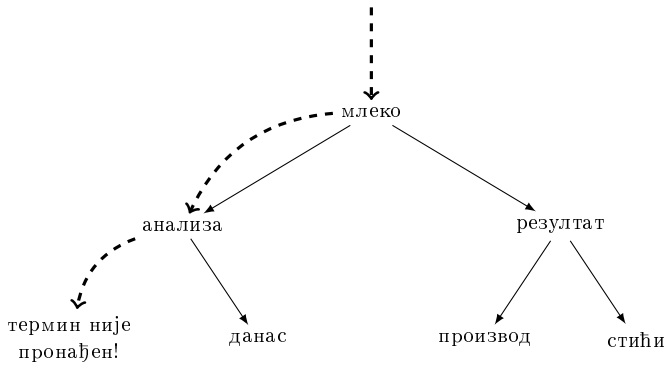


Слика 3.3 Претрага речника по термину „производ“.

Без обзира на то да ли се термин по коме се врши претрага налази у стаблу или не, број корака потребних да се изврши претраживање стабла није већи од висине стабла[2] увећане за један. За бинарно стабло кажемо да је уравнотежено (енгл. *balanced binary tree*) када је минималне висине за дати скуп термина. У оптималном случају, n индексних термина се може сместити у уравнотежено стабло висине $\lfloor \log_2 n \rfloor$, а временска сложеност алгоритма за претрагу овог бинарног стабла је $O(\log n)$.

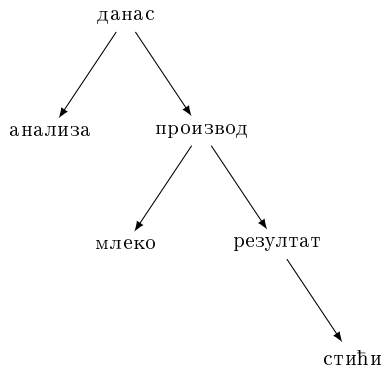
[1] Сматраћемо да је термин t_1 мањи од термина t_2 ако му претходи, не обавезно непосредно, у азбучном редоследу.

[2] Висина стабла се дефинише као број грана од корена до најудаљенијег листа.



Слика 3.4 Претрага речника по термину „афлатоксин“.

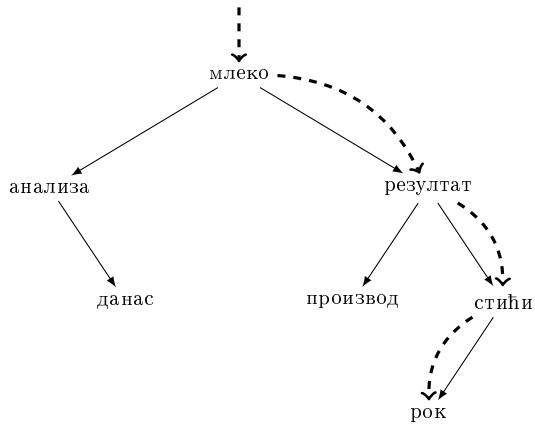
Пример 3.9. Бинарно стабло приказано на сл. 3.2 је уравнотежено, висине 2. Бинарно стабло приказано на сл. 3.5 представља исти речник индексних термина, али није уравнотежено (висина овог стабла је 3), што значи да је максимални број корака потребних да се изврши претраживање стабла увећан у односу на претходно стабло.



Слика 3.5 Неуравнотежено бинарно стабло које представља речник индексних термина приказан на сл. 3.1.

Нови термин се у бинарно стабло увек додаје као терминални чвор. Поступак додавања термина је исти као и поступак проналажења термина у стаблу, све док се не дође до терминалног чвора. Након тога, ако је нови термин мањи од термина у текућем терминалном чвору, додаје му се као лево дете. У супротном, додаје се као десно дете текућег терминалног чвора.

Пример 3.10. Додавање термина „рок“ у бинарно стабло је илустровано на сл. 3.6. Путања претраге која претходи додавању новог терминалног чвора је представљена испрекиданим линијама.

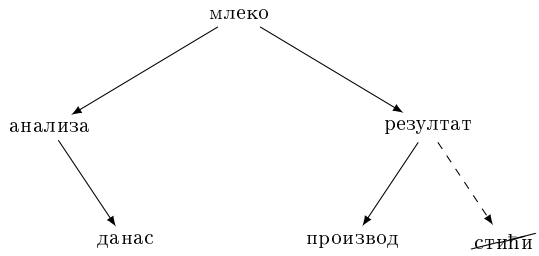


Слика 3.6 Додавање термина у сортирано бинарно стабло.

Поступак уклањања термина из речника зависи од тога да ли се термин налази у терминалном или нетерминалном чвору. Ако се термин налази у терминалном чвору, довољно је уклонити чвор из стабла. У супротном, нетерминални чвор који садржи термин се уклања, а на његово место долази терминални чвор који се бира на један од следећих начина:

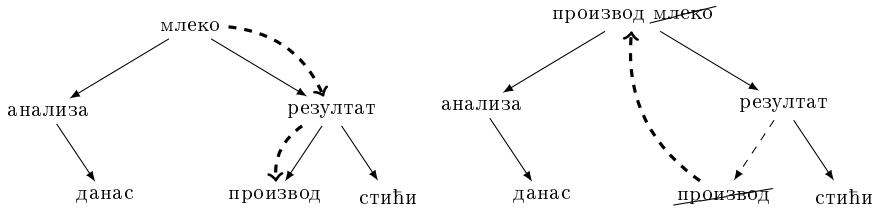
- од десног детета (ако постоји) посматраног чвора, спуштамо се левим гранама до терминалног чвора, или
- од левог детета (ако постоји) посматраног чвора, спуштамо се десним гранама до терминалног чвора.

Пример 3.11. Уклањање термина „стићи“ из бинарног стабла је илустровано на сл. 3.7. Два начина да се уклони нетерминални чвор „млеко“ су илустрована на сл. 3.8 и 3.9.

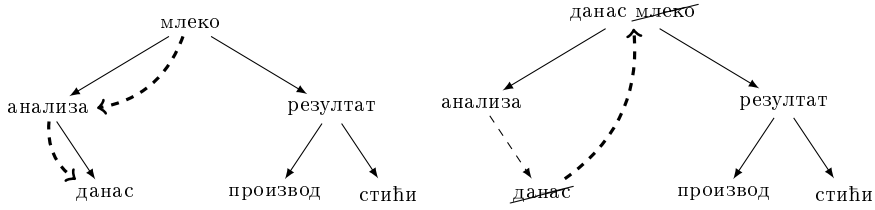


Слика 3.7 Уклањање терминалног чвора из сортираног бинарног стабла.

Описани поступци за додавање чвора у стабло и уклањање чвора из стабла чувају особину сортираности стабла (в. задатак 3.6), али могу да доведу до тога да стабло постане неуравнотежено, што смањује ефикасност претраге. Због тога је пожељно одржавати стабло уравнотеженим, о чему ће бити речи у наредној секцији.



Слика 3.8 Уклањање нетерминалног чвора „млеко“ из сортираног бинарног стабла — прво решење. На левом делу слике је илустровано селектовање терминалног чвора који ће доћи на место чвора „млеко“. На десном делу слике је приказан изглед стабла након уклањања чвора „млеко“.



Слика 3.9 Уклањање нетерминалног чвора „млеко“ из сортираног бинарног стабла — друго решење. На левом делу слике је илустровано селектовање терминалног чвора који ће доћи на место чвора „млеко“. На десном делу слике је приказан изглед стабла након уклањања чвора „млеко“.

3.2.1 Дej-Стаут-Воренов алгоритам

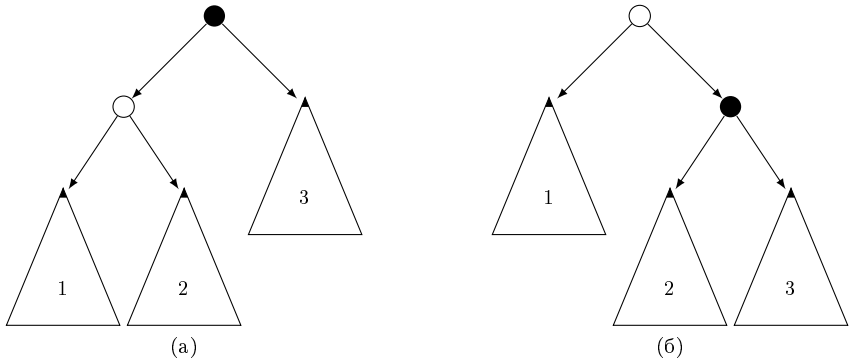
У наставку ћемо размотрити Дej-Стаут-Воренов (Day–Stout–Warren) алгоритам за уравнотежење стабла [1, 3, 4], који се заснива на операцији ротирања чворова у стаблу. Сл. 3.10 илуструје десну и леву ротацију чвора. Трансформација стабла (а) $\rightarrow$ (б) представља десну ротацију чвора $\bullet$. Трансформација стабла (б) $\rightarrow$ (а) представља леву ротацију чвора $\circ$. Троуглови представљају подстабла.

Алгоритам за уравнотежење стабла се извршава у две фазе:

- у првој фази, сортирано бинарно стабло се трансформише у сортирану једноструко уланчану листу,
- у другој фази, листа чворова која је резултат прве фазе преводи се у уравнотежено бинарно стабло.

Кораци прве фазе су следећи:

1. На почетку извршавања алгоритма, корен стабла представља текући чвор.
2. Све док текући чвор t има лево дете l, понавља се следећа секвенца операција:
   - десна ротација чвора t,



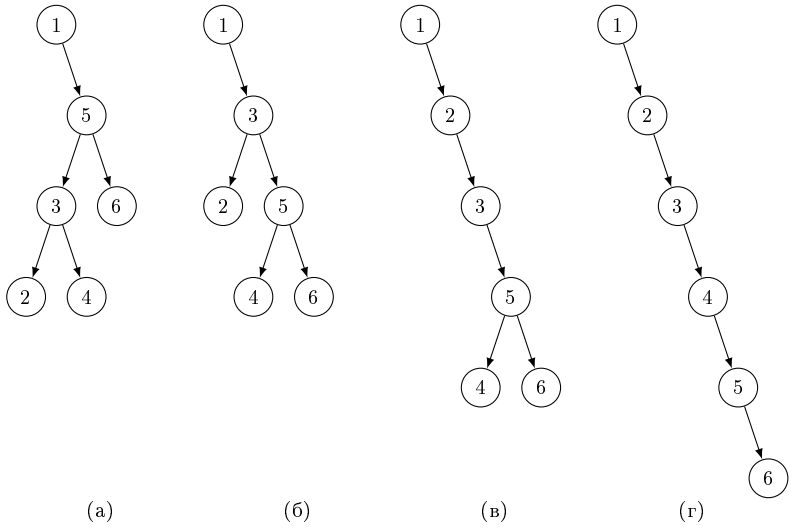
Слика 3.10 Ротације чвора (в. [4]).

– l постаје текући чвор.

3. Ако текући чвор има десно дете, онда оно постаје текући чвор — а извршавање алгоритма се враћа на корак 2. Ако текући чвор нема десно дете, алгоритам је завршен.

Пример 3.12. Прва фаза алгоритма за уравнотежење сортираног бинарног стабла је илустрована на сл. 3.11. Почетно стабло је приказано на делу слике (а). На почетку, корен ① је текући чвор. Међутим, пошто он нема лево дете, ⑤ постаје текући чвор. Овај чвор има лево дете, па се врши десна ротација чвора ⑤. Структура стабла у овом тренутку је приказана на делу слике (б), а нови текући чвор је ③. Чвор ③ има лево дете, па се врши десна ротација овог чвора. Нова структура стабла је приказана на делу слике (в), а текући чвор је ②. Чвор ② нема лево дете, па његово десно дете, чвор ③, постаје текући чвор. Сада ни чвор ③ нема лево дете, па ⑤ постаје текући чвор. Он има лево дете, па се врши десна ротација чвора ⑤. Структура стабла у овом тренутку је приказана на делу слике (г), а текући чвор је ④. Чвор ④ нема лево дете, па његово десно дете, чвор ⑤, постаје текући чвор. Ни чвор ⑤ нема лево дете, па чвор ⑥ постаје текући чвор. Овај чвор нема децу, што значи да је прва фаза извршавања алгоритма завршена. На делу слике (г) је приказана сортирана, једноструко уланчана листа која представља резултат прве фазе.

У другој фази извршавања алгоритма се листа чворова преводи у уравнотежено стабло. За лакше представљање ове фазе, дефинишимо појам окоснице стабла. Окосницу стабла чине чворови (и гране између њих) на путањи која почиње од корена и прати само десне гране. Нпр., окосницу стабла на сл. 3.11(а) чине чворови ①, ⑤ и ⑥, док окосница стабла на сл. 3.11(г) обухвата цело стабло. Кораци друге фазе извршавања алгоритма се могу описати на следећи начин. Почевши од корена стабла, сваки други чвор на окосници стабла се ротира улево, под условом да



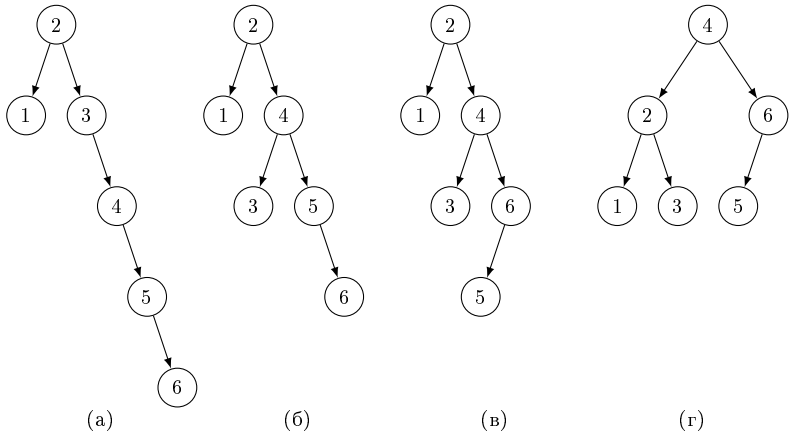
Слика 3.11 Прва фаза извршавања алгоритма за уравнотежење сортираног бинарног стабла.

има десно дете. Ово се понавља све док резултујуће стабло не постане уравнотежено.

Пример 3.13. Превођење листе чворова дате на сл. 3.11(г) у уравнотежено бинарно стабло је илустровано на сл. 3.12. Почевши од чвора ①, сваки други чвор у листи се ротира улево. Структуре стабла након ротирања чворова ①, ③ и ⑤ су приказане на деловима слике (а), (б), (в), респективно. Пошто стабло приказано на делу слике (в) још увек није уравнотежено, понавља се поступак ротирања чворова на његовој окосници, коју сад чине чворови ②, ④ и ⑥. Чвор ② се ротира улево, а ротација чвора ⑥ није могућа, јер нема десно дете. Резултујуће стабло је уравнотежено и приказано на делу слике (г).

Временска сложеност овог алгоритма је линеарна, тј., $O(n)$ за стабло које садржи n чворова, док му је просторна сложеност константна [4]. Уравнотежење стабла се може вршити приликом сваког додавања и уклањања чвора, али је ефикасније када се врши повремено, нпр., кад висина стабла постане већа од унапред задате вредности [3, 5].

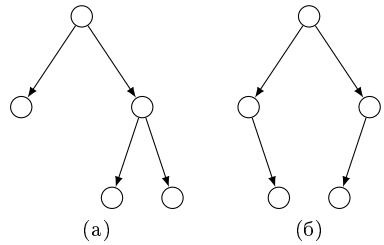
Постоји разлика између уравнотежених и савршено уравнотежених бинарних стабала. Уравнотежено бинарно стабло минимизује максималну дубину чворова и просечну дубину чворова. Савршено уравнотежено бинарно стабло задовољава још један услов: за сваки чвор стабла важи да се бројеви његових левих и десних потомака разликују максимално за 1. На сл. 3.13 су приказана два бинарна стабла са



Слика 3.12 Друга фаза извршавања алгоритма за уравнотежење сортираног бинарног стабла.

истим бројем чворова, од којих је лево стабло (само) уравнотежено, а десно савршено уравнотежено. Треба приметити да је за ефикасну претрагу довољно да стабло буде уравнотежено (тј., није потребно да буде и савршено уравнотежено) [4].

Слика 3.13 (а) Уравнотежено бинарно стабло и (б) савршено уравнотежено бинарно стабло (в. [4]).



3.3 Задаци

Задатак 3.1. За дату колекцију података, извршите токенизацију, уклањање предлога и везника, нормализацију и лематизацију, и прикажите инвертовани индекс (са листама):

- D_1: У Београду нови аутобуси
- D_2: Toplije u Novom Sadu

- D_3: У Новом Саду топлије него у Београду
- D_4: Гужва у Новом Београду

Задатак 3.2. Претпоставите да се у поступку нормализације сви токени преводе на ћирилично писмо, и да се латинична слова „lj“, „nj“, „dž“ (итд.) преводе у „љ“, „њ“, „џ“ (итд.), респективно. Овим поступком би се токени „konjunktura“, „konjunktivitis“ и „injekcija“ превели у „коњунктура“, „коњунктивитис“ и „ињекција“ — који се разликују од правилно написаних речи на ћирилици: „конјунктура“, „конјунктивитис“ и „инјекција“. Објасните проблем у претраживању инвертованог индекса који се на овај начин проузрокује, и предложите решење.

Задатак 3.3. За дате низове термина, прикажите бинарна сортирана стабла која осликавају редослед термина у низовима. Речи сортирајте према азбучном редоследу.

а) бета, делта, зета, епсилон, алфа, гама,
б) гама, епсилон, зета, делта, алфа, бета.

Прикажите ова стабла када се из њих уклони чвор који одговара термину бета.

Задатак 3.4. Зашто је сортирано бинарно стабло прикладније од сортиране листе за представљање речника индексних термина?

Задатак 3.5. Зашто је уравнотеженост бинарног стабла које представља речник индексних термина важна за ефикасну претрагу речника?

Задатак 3.6. Покажите да описани поступци за додавање чвора у стабло и уклањање чвора из стабла у општем случају чувају особину сортираности стабла.

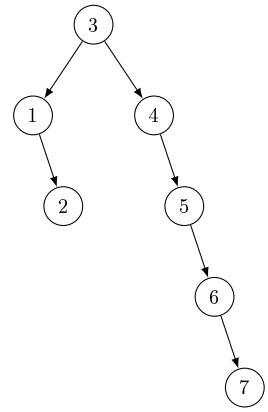
Задатак 3.7. Одредите временску сложеност описаних операција за додавање чвора у стабло и уклањање чвора из стабла.

Задатак 3.8. Покажите да операције ротирања чвора у општем случају чувају особину сортираности стабла.

Задатак 3.9. Илуструјте извршавање Дej-Стаут-Вореновог алгоритма за уравнотежење стабла приказаног на сл. 3.14.

Литература

1. Day, AC (1976) Balancing a Binary Tree. Computer Journal 19 (4): 360–361.
2. Manning CD, Raghavan P, Schütze H (2008) Introduction to Information Retrieval, Cambridge University Press.
3. Rolfe TJ (2002) One-time Binary Search Tree Balancing: The Day/Stout/Warren (DSW) Algorithm. In SIGCSE Bull. 34.4, pp. 85–88.



Слика 3.14 Слика уз задатак 3.9.

4. Stout QF, Warren BL (1986) Tree Rebalancing in Optimal Time and Space. In Commun. ACM 29.9, pp. 902–908.
5. Williams HE, Zobel J, Heinz S (2001) Self-adjusting trees in practice for large text collections. In Software: Practice and Experience 31.10, pp. 925–939.

Поглавље 4
Контекстно зависно предвиђање корисничких упита

Савремени системи за проналажење информација на вебу укључују функционалност контекстно зависног предвиђања корисничких упита. То значи да систем, за потенцијално некомплетни кориснички упит, понуди скуп највероватнијих упита који га комплетирају.

Пример 4.1. У тренутку писања овог примера, за унети упит „летовање“, Гугл је предвидео следећи скуп потенцијалних наставака: „летовање *у Грчкој*“, „летовање *у Србији*“, „летовање *Бугарска*“. За непотпуно унету реч „летов“, Гугл је понудио: „летов*ање*“ и „летов*и*“.

Једна од основних особина природних језика је да редослед речи у говорним и писменим манифестацијама није произвољан, чак и у језицима који допуштају флексибилни редослед речи. Практично, то значи да за дати низ речи често можемо исправно да претпоставимо наредну реч, на основу искуства и знања о свету.

Пример 4.2. Да ли можете да предвидите реч која наставља дати низ речи:

„данас је леп . . . ”?

Кандидата има више. Неки су *очекивани*, попут именице „дан” (нпр., „данас је леп дан”) или везника „и” (нпр., „данас је леп и сунчан”), а неки су мање вероватни, иако могући, попут придева „нов” (нпр., „данас је леп Нови Београд”). Ипак, већини матерњих говорника српског језика ће именица „дан” бити први избор. То није случајно, јер се именица „дан” у свакодневној употреби српског језика јавља након низа речи „данас је леп” чешће од било које друге речи. Као илустрацију ове тврдње можете да упоредите број резултата које Гугл врати за упите „данас је леп” и „данас је леп дан”. У тренутку писања овог примера, број резултата за други упит је чинио приближно 40% броја резултата за први упит[1].

[1] Упити из овог примера садрже знаке навода, што значи да Гугл тражи документа који садрже задате фразе „данас је леп” и „данас је леп дан” (в. поглавље 2). Осим тога,

Пример 4.3. Процењивање вероватноће наредне речи је уско повезано са процењивањем вероватноће секвенце речи. Интуитивно је јасно да постоји одређена ненулта вероватноћа јављања секвенце речи:

„свега овога не би било да је Пера отишао право у полицију”,

као и да секвенца која садржи исте речи, у промењеном редоследу, има веома малу вероватноћу јављања:

„је овога право би не било свега Пера полицију у отишао да”.

Н-грами су статистички модели који на основу претходних $(\mathrm{H}-1)$ речи предвиђају наредну реч, тј. рачунају вероватноћу њеног јављања [6, 3]. Овакви модели се често називају *језичким моделима*, и имају широку област примене у обради природног језика, аутоматском препознавању говора, машинском превођењу, препознавању рукописа, идентификацији аутора текста, аугментативној комуникацији, итд. Ово поглавље разматра н-граме у контексту предвиђања корисничких упита.

Контекстно зависно предвиђање корисничких упита је специјални случај аутоматизованог исправљања словних грешака у корисничким уносима, што је детаљније размотрено у поглављу 5.

4.1 Процена максималне изгледности

Пре него што формално уведемо н-граме, подсетићемо се основних појмова из теорије вероватноће. Вероватноћу случајног догађаја A означавамо $P(A)$, при чему увек важи:

$$0 \leq P(A) \leq 1\,. \tag{4.1}$$

Ако је $P(A) = 0$, догађај A је немогућ, а ако је $P(A) = 1$, догађај A је сигуран. У простору једнако вероватних елементарних догађаја важи[2]:

$$P(A) = \frac{\text{број повољних исхода}}{\text{број могућих исхода}}\,. \tag{4.2}$$

Размотримо ову дефиницију на неколико једноставних примера.

треба напоменути да Гугл користимо само као илустрацију. Из добијених резултата се не би могло закључити да је вероватноћа да после низа речи „данас је леп” следи реч „дан” једнака 40%, јер број резултата које Гугл враћа одговара броју докумената који садрже задате фразе, а не укупни број јављања задатих фраза у пронађеним документима.

[2] Ово је упрошћена класична (Лапласова) дефиниција вероватноће.

Пример 4.4. Колика је вероватноћа да се при бацању коцке добије број 5? Приликом бацања коцке постоји шест могућих исхода $\{1,2,3,4,5,6\}$, од којих је само један повољан $\{5\}$. Зато је вероватноћа да се добије број 5 једнака $\frac{1}{6}$.

Пример 4.5. Колика је вероватноћа да се при бацању коцке добије парни број? Опет постоји шест могућих исхода $\{1,2,3,4,5,6\}$, од којих су три повољна $\{2,4,6\}$, па је тражена вероватноћа $\frac{3}{6} = \frac{1}{2}$.

Догађаји A и B су зависни уколико реализација догађаја A утиче на реализацију догађаја B. Вероватноћу реализације догађаја B под условом да се реализовао догађај A означаваћемо $P(B|A)$. Вероватноћу да се реализују догађаји A и B означаваћемо $P(AB)$. Важи:

$$P(AB) = P(A)P(B|A)\ . \tag{4.3}$$

Вратимо се сад на пример 4.2. Оно што нас интесује је како можемо да проценимо следеће вероватноће:

- вероватноћу јављања речи „дан” после секвенце речи „данас је леп”:

$$P(\text{„дан”}|\text{„данас је леп”})\ , \tag{4.4}$$

- вероватноћу секвенце речи „данас је леп дан”:

$$P(\text{„данас је леп дан”})\ . \tag{4.5}$$

Израчунавање ових вероватноћа се такође заснива на бројању исхода. Бројање речи се врши у оквиру изабраног језичког корпуса[3]. Постоји више приступа бројању исхода, од којих сви имају предности и мане. Овде ћемо размотрити приступ који се заснива на *процени максималне изгледности* (енгл. *maximum likelihood estimation*) применом н-грама. Да бисмо објаснили овај приступ, потребно је да прво размотримо приступе израчунавању вероватноће применом простог бројања и условних вероватноћа.

4.1.1 Просто бројање

Проценимо вероватноћу јављања речи „дан” после секвенце „данас је леп”, тј., P(„дан”|„данас је леп”), применом простог бројања. У изабраном језичком корпусу се преброје:

[3] Овде користимо израз „језички корпус”, који подразумева колекцију текстова или снимака говора. У контексту проналажења информација на вебу, можемо користити и специфичнији израз „колекција докумената”.

- сви позитивни исходи, тј., сва јављања секвенце „данас је леп дан” — означимо овај број са C(„данас је леп дан”).
- сви могући исходи, тј., сва јављања секвенце „данас је леп” после које долази било која реч — означимо овај број са C(„данас је леп”).

Тада је:

$$P(\text{„дан”}|\text{„данас је леп”}) = \frac{C(\text{„данас је леп дан”})}{C(\text{„данас је леп”})} . \tag{4.6}$$

Да би се проценила вероватноћа секвенце речи „данас је леп дан”, тј., P(„данас је леп дан”),у изабраном језичком корпусу се преброје:

- сви позитивни исходи, тј., сва јављања секвенце „данас је леп дан” — означимо овај број са C(„данас је леп дан”).
- сви могући исходи, тј., сва јављања свих секвенци које садрже четири речи (јер је то број речи у секвенци „данас је леп дан”) — означимо овај број са C(секвенца 4 речи).

Тада је:

$$P(\text{„данас је леп дан”}) = \frac{C(\text{„данас је леп дан”})}{C(\text{секвенца 4 речи})} . \tag{4.7}$$

Иако интуитиван, овај приступ израчунавању вероватноћа речи и секвенци речи није адекватан. У теорији, кад би језички корпус над којим се врши пребројавање био репрезентативан, овај приступ би био прихватљив. Репрезентативност језичког корпуса се односи на обим у којем корпус укључује целокупни опсег разноврсности у оквиру датог језика [1], тј., на обим заступљености и узајамне балансираности различитих типова текстова и различитих језичких феномена који се јављају у посматраном језику, итд. Међутим, питање продукције репрезентативних језичких корпуса је још увек отворено. Један од разлога је што је језик подложан променама — стално настају нове реченице које нису присутне у изабраном језичком корпусу. Осим тога, критеријуми за продукцију језичких корпуса уобичајено произлазе из интуитивних, а самим тим и непоузданих, увида истраживача, и зато се не може очекивати да ће језички корпуси, ма колико пажљиво били произведени, садржати језичке манифестације свих релевантних језичких феномена [9, 10].

Пример 4.6. Чак ни језички корпус који садржи документа доступна на вебу није довољан да се простим бројањем поуздано процене вероватноће речи и секвенци речи. Као илустрацију, смислите синтаксно и семантички исправну реченицу коју Гугл не може да нађе.

4.1.2 Условне вероватноће

Да бисмо илустровали овај приступ, секвенцу речи „данас је леп дан" ћемо посматрати као реализацију два догађаја:

- A — реализовала се секвенца речи „данас је леп",
- B — реализовала се реч „дан".

Тада, по формули (4.3), важи:

$$\begin{aligned} P(\text{„данас је леп дан"}) &= P(AB) \\ &= P(A)P(B|A) \\ &= P(\text{„данас је леп"})P(\text{„дан"}|\text{„данас је леп"}) \,. \end{aligned} \tag{4.8}$$

Вишеструком применом формуле (4.3), добијамо:

$$\begin{aligned} P(\text{„данас је леп дан"}) &= P(\text{„данас је леп"})P(\text{„дан"}|\text{„данас је леп"}) \\ &= P(\text{„данас је"})P(\text{„леп"}|\text{„данас је"})P(\text{„дан"}|\text{„данас је леп"}) \\ &= P(\text{„данас"})P(\text{„је"}|\text{„данас"})P(\text{„леп"}|\text{„данас је"})P(\text{„дан"}|\text{„данас је леп"}) \,. \end{aligned} \tag{4.9}$$

Ову формулу можемо да уопштимо за произвољну секвенцу четири речи. Вероватноћа јављања секвенце w_1, w_2, w_3, w_4 је:

$$P(w_1w_2w_3w_4) = P(w_1)P(w_2|w_1)P(w_3|w_1w_2)P(w_4|w_1w_2w_3) \,. \tag{4.10}$$

У општем случају, када секвенца W садржи n речи, важи:

$$\begin{aligned} P(W) &= P(w_1w_2...w_n) \\ &= P(w_1)P(w_2|w_1)P(w_3|w_1w_2)...P(w_n|w_1w_2...w_{n-1}) \,. \end{aligned} \tag{4.11}$$

Треба приметити да се процењивање вероватноће $P(w_n|w_1w_2...w_{n-1})$ опет своди на пребројавање секвенци речи у корпусу, тако да ни овај приступ није адекватан, из истих разлога због којих ни претходни приступ није адекватан.

4.1.3 Процена максималне изгледности биграма

Основна идеја н-грама је да се приликом рачунања вероватноће јављања неке речи не узимају у обзир све речи у секвенци која јој претходи, већ само неколико последњих речи које јој непосредно претходе. Језички модел који у обзир узима само једну реч која непосредно претходи посматраној речи се назива *биграм*. Овај модел се заснива на тзв. Марковљевој претпоставци да важи:

$$P(w_n|w_1w_2...w_{n-1}) \approx P(w_n|w_{n-1}) \,. \tag{4.12}$$

Формулу (4.11) можемо сад да упростимо:

$$\begin{aligned} P(W) &= P(w_1w_2...w_n) \\ &= P(w_1)P(w_2|w_1)P(w_3|w_1w_2)...P(w_n|w_1w_2...w_{n-1}) \\ &= P(w_1)P(w_2|w_1)P(w_3|w_2)...P(w_n|w_{n-1}) \,. \end{aligned} \tag{4.13}$$

Пример 4.7. Ако претпоставимо да вероватноћа јављања неке речи зависи само од речи која јој непосредно претходи, тада је.:

$$P(\text{„дан”}|\text{„данас је леп”}) \approx P(\text{„дан”}|\text{„леп”}) \,. \tag{4.14}$$

По формули (4.13), вероватноћа јављања секвенце „данас је леп дан” је:

$$\begin{aligned} &P(\text{„данас је леп дан”}) \\ &= P(\text{„данас”})P(\text{„је”}|\text{„данас”})P(\text{„леп”}|\text{„је”})P(\text{„дан”}|\text{„леп”}) \,. \end{aligned} \tag{4.15}$$

Интуитивно је јасно да је вероватније да се реч „дан” јави после секвенце речи „данас је леп” него само после речи „леп” (в. задатак 4.1). У ствари, што више предисторије узмемо у обзир, процена вероватноће би требало да буде тачнија. Прихватањем Марковљеве претпоставке намерно редукујемо предисторију посматране речи на само једну реч. Наизглед, то ће довести до лошије проценене вероватноће — али није тако. У секцији 4.1.1 смо размотрили проблеме који онемогућавају прецизно процењивање вероватноће речи са знатном предисторијом: $P(w_n|w_1w_2...w_{n-1})$. Главни проблем је што не можемо бити сигурни да су бројеви јављања секвенци речи $w_1w_2...w_{n-1}$ и $w_1w_2...w_{n-1}w_n$ репрезентативни. Са друге стране, рачунање вероватноће речи са редукованом предисторијом $P(w_i|w_j)$ се може свести на пребројавање речи у језичком корпусу:

$$P(w_i|w_j) = \frac{C(w_jw_i)}{\sum\limits_{w_x \in V} C(w_jw_x)} \,, \tag{4.16}$$

где су:

- $C(w_jw_i)$ — број јављања секвенце речи w_jw_i,
- $\sum\limits_{w_x \in V} C(w_jw_x)$ — број јављања секвенци које садрже две речи, при чему је прва реч w_j, а друга је произвољна реч w_x из речника V.

Број јављања секвенци речи w_jw_x, при чему је реч w_x произвољна, је једнак броју јављања речи w_j, па се претходна формула може упростити:

$$P(w_i|w_j) = \frac{C(w_jw_i)}{C(w_j)} \,, \tag{4.17}$$

где су:

- $C(w_j w_i)$ — број јављања секвенце речи $w_j w_i$,
- $C(w_j)$ — број јављања речи w_j.

Формула (4.17) се назива *проценом максималне изгледности*.

Пример 4.8. Да бисмо илустровали процену максималне изгледности, намерно бирамо мали језички корпус:

$< p >$ данас је леп дан $< /p >$
$< p >$ данас је нерадни дан $< /p >$
$< p >$ доћи ће данас $< /p >$

Овај корпус садржи само три реченице, при чему су све речи намерно написане малим словима, а сви знакови интерпункције уклоњени. Токени $< p >$ и $< /p >$ означавају почетак, односно крај реченице, и такође се узимају у обзир приликом бројања речи. У формули (4.13), сви чланови производа су у облику $P(w_i|w_j)$, и могу да се рачунају по формули (4.17). Иако није експлицитно наведено, ово важи и за први члан $P(w_1)$. Овај члан означава вероватноћу да је w_1 прва реч у реченици, што можемо записати $P(w_1| < p >)$. Такође, вероватноћа да је w_n последња реч у реченици се означава $P(< /p > |w_n)$. Када се у обзир узму токени $< p >$ и $< /p >$, формула (4.13) постаје:

$$P(W) = P(w_1| < p >)P(w_2|w_1)P(w_3|w_2)...P(w_n|w_{n-1})P(< /p > |w_n)\,. \quad (4.18)$$

Вероватноће неких биграма, израчунате по формули (4.17), су дате у наставку:

$$P(\text{„данас”}| < p >) = \frac{2}{3}\,, \qquad P(\text{„је”}|\text{„данас”}) = \frac{2}{3}\,, \quad (4.19)$$
$$P(\text{„доћи”}| < p >) = \frac{1}{3}\,, \qquad P(\text{„леп”}|\text{„је”}) = \frac{1}{2}\,.$$

Међутим, вероватноће неких биграма су једнаке нули, нпр.:

$$P(\text{„доћи”}|\text{„ће”}) = 0\,, \quad (4.20)$$

што би требало да имплицира следеће закључке:

- вероватноћа јављања речи „доћи” после „ће” је нула,
- вероватноћа јављања било које реченице која садржи секвенцу речи „ће доћи” је нула (ово следи из формуле (4.13)),

за које искуствено знамо да нису коректни. Иако изгледа да је овај проблем узрокован искључиво премалим корпусом, он је актуелан чак и кад се користе већи језички корпуси. Слични проблем се јавља и кад су вероватноће биграма (а у општем случају н-грама) блиске нули, јер су тада вероватноће јављања реченица које садрже посматране биграме такође

неоправдано мале. То значи да формула за процену максималне изгледности није одговарајућа. У наредној секцији ћемо размотрити Лапласову корекцију ове формуле.

Ако накратко занемаримо проблем биграма са нултим вероватноћама, видећемо да ненулте вероватноће које смо израчунали у претходном примеру нису репрезентативне (тј., неоправдано су високе). То је последица тога што је језички корпус који смо разматрали екстремно мали. Реални корпуси су знатно већи. Као илустрација, неки детаљи у вези са Гугловим језичким корпусом текстова на енглеском језику су дати у табели 4.1.

Табела 4.1 Неки детаљи у вези са Гугловим језичким корпусом текстова на енглеском језику [2, 4].

Број токена:	1 024 908 267 229
Број реченица:	95 119 665 584
Број униграма:	13 588 391
Број биграма:	314 843 401
Број триграма:	977 069 902
Број тетраграма:	1 313 818 354
Број пентаграма:	1 176 470 663

4.2 Лапласова корекција

Лапласова корекција (енгл. *Laplace smoothing*, [6, 3]) формуле (4.17) је:

$$P_L(w_i|w_j) = \frac{C(w_j w_i)+1}{C(w_j)+|V|}\,, \tag{4.21}$$

где су:

- $C(w_j w_i)$ — број јављања секвенце $w_j w_i$,
- $C(w_j)$ — број јављања речи w_j,
- $|V|$ — број различитих речи у посматраном језичком корпусу.

У односу на формулу за процену максималне изгледности, Лапласова корекција увећава бројилац разломка за један (тј., $C(w_j w_i)+1$) и тиме спречава израчунавање нулте вероватноће. Пошто је $|V|$ број различитих речи у посматраном језичком корпусу, а број јављања сваке од њих је

сад увећан за један, именилац разломка је потребно увећати за $|V|$ (тј., $C(w_j)+|V|$), тј.:

$$\begin{aligned} P(w_i|w_j) &= \frac{C(w_j w_i)+1}{\sum\limits_{w_x \in V}(C(w_j w_x)+1)} \\ &= \frac{C(w_j w_i)+1}{\sum\limits_{w_x \in V} C(w_j w_x)+|V|} \\ &= \frac{C(w_j w_i)+1}{C(w_j)+|V|}\,. \end{aligned}$$

Пример 4.9. Израчунајмо вероватноће изабраних биграма из примера 4.8, за исти језички корпус, коришћењем кориговане формуле (4.21):

$$\begin{aligned} P(\text{„данас”}|<p>) &= \frac{2+1}{3+9} = \frac{3}{12}\,, \quad P(\text{„је”}|\text{„данас”}) = \frac{2+1}{3+9} = \frac{3}{12}\,, \\ P(\text{„доћи”}|<p>) &= \frac{1+1}{3+9} = \frac{2}{12}\,, \qquad P(\text{„леп”}|\text{„је”}) = \frac{1+1}{2+9} = \frac{2}{11}\,, \\ P(\text{„доћи”}|\text{„ће”}) &= \frac{0+1}{1+9} = \frac{1}{10}\,. \end{aligned} \tag{4.22}$$

У табели 4.2 су упоредно приказане вероватноће добијене проценом максималне изгледности у примеру 4.8, и коригованом формулом у овом примеру. Може се приметити да је Лапласова корекција решила проблем биграма чија је вероватноћа раније била нула, али је и значајно смањила вероватноће осталих биграма.

Табела 4.2 Упоредни приказ вероватноћа добијених проценом максималне изгледности, односно Лапласовом корекцијом ове формуле.

Вероватноћа	Процена максималне изгледности	Лапласова корекција
P(„данас”|<p>)	0.67	0.25
P(„доћи”|<p>)	0.33	0.17
P(„је”|„данас”)	0.67	0.25
P(„леп”|„је”)	0.5	0.18
P(„доћи”|„ће”)	0	0.1

Пример 4.10. Израчунајмо сад вероватноћу речнице „данас ће доћи“, користећи језички корпус из примера 4.8. Иако три од четири биграма који се јављају у овој реченици не постоје у изворном корпусу, употребом формула (4.13) и (4.21) можемо да проценимо вероватноћу реченице:

$$
\begin{aligned}
&P(\text{„данас ће доћи“}) \\
&= P(\text{„данас“}|<p>)P(\text{„ће“}|\text{данас})P(\text{„доћи“}|\text{„ће“})P(</p>|\text{„доћи“}) \qquad (4.23)\\
&= \frac{2+1}{3+9}\cdot\frac{0+1}{3+9}\cdot\frac{0+1}{1+9}\cdot\frac{0+1}{1+9}\,.
\end{aligned}
$$

Раније смо указалали на чињеницу да корпус који користимо у примерима у овом поглављу није репрезентативан, и да вероватноће које добијамо анализом овог коруса не рефлектују стварно стање. Зато их не израчунавамо до краја, већ само илуструјемо поступак израчунавања.

Иако Лапласова корекција није адекватно решење за модерне примене н-грама [5], она представља важни концепт на коме се заснивају напредније методе за корекцију вероватноћа добијених методом процене максималне изгледности, укључујући методе интерполације и контекстно зависног редуковања предисторије н-грама [7, 8, 3].

4.3 Моделовање речи ван речника

Н-граме смо до сад разматрали под претпоставком *затвореног речника*, тј., процењивали смо вероватноће секвенце речи под претпоставком да све речи припадају задатом језичком корпусу. Међутим, подаци за тестирање могу да садрже речи које нису присутне у корпусу.

Пример 4.11. Ако пробамо да израчунамо вероватноћу реченице „данас неће доћи“, користећи језички корпус из примера 4.8, видећемо да нам формуле (4.13) и (4.21) нису корисне, јер реченица садржи реч „неће“ која се не налази у корпусу.

Овакве речи називамо непознатим речима, или *речима ван речника* (енгл. *out-of-vocabulary words*). Имајући динамичку природу језика у виду, вероватноћа јављања речи ван унапред задатог речника није занемарљива, па се намеће питање како можемо да проценимо вероватноће јављања оваквих речи.

Системи са *отвореним речником* моделују речи ван речника на следећи начин:

1. Дефинише се речник (познатих речи).
2. У језичком корпусу се све речи ван речника замене токеном који означава непознате речи, нпр: $<\text{HP}>$.
3. Вероватноћа јављања токена $<\text{HP}>$ се рачуна на исти начин као и за све речи које припадају речнику.

Пример 4.12. Претпоставимо да смо за језички корпус из примера 4.8 дефинисали речник који садржи речи „данас“, „леп“, и „доћи“, док се све остале речи сматрају непознатим. Када се у посматраном корпусу све речи ван речника замене токеном који означава непознате речи добијамо:

$< p >$ данас $<$ НР $>$ леп $<$ НР $>$ $< /p >$
$< p >$ данас $<$ НР $>$ $<$ НР $>$ $<$ НР $>$ $< /p >$
$< p >$ доћи $<$ НР $>$ данас $< /p >$

Приметите да је сад број различитих речи у речнику $|V| = 6$. Вероватноћа реченице „данас неће доћи“, која садржи непознату реч „неће“, се рачуна на следећи начин:

$$P(\text{„данас неће доћи“}) \tag{4.24}$$
$$= P(\text{„данас“}|< p >)P(\text{„неће“}|\text{данас})P(\text{„доћи“}|\text{„неће“})P(< /p > |\text{„доћи“})$$
$$= P(\text{„данас“}|< p >)P(< \text{НР} > |\text{данас})P(\text{„доћи“}|< \text{НР} >)P(< /p > |\text{„доћи“})$$
$$= \frac{2+1}{3+6} \cdot \frac{2+1}{3+6} \cdot \frac{0+1}{6+6} \cdot \frac{0+1}{1+6}.$$

У претходним примерима смо видели да израчунавање вероватноће секвенце речи укључује множење вероватноћа појединачних биграма. За реалне језичке корпусе, вероватноће биграма су довољно мале да њихово множење може произвести вредности чије су апсолутне вредности толико мале да се не могу правилно рачунарски представити, па се заокружују на вредност 0. Овај ефекат се назива аритметичко поткорачење (енгл. *arithmetic underflow*), и може да узрокује грешке у извршавању програма. Да бисмо избегли овај ефекат и убрзали израчунавање, уместо вероватноће P се користи њена логаритмована вредност $\log P$, што операције множења вероватноћа у горњим формулама замењује операцијама сабирања логаритмованих вредности вероватноћа [6], тј.:

$$P_1 \cdot P_2 \cdot \ldots \cdot P_n = e^{\log P_1 + \log P_2 + \ldots + \log P_n}$$

4.4 Обучавање и тестирање језичких модела

У претходним примерима смо могли да уочимо да вероватноће појединачних секвенци речи зависе од изабраног језичког корпуса. То отвара питање процене адекватности добијеног језичког модела. Овде ћемо размотрити приступ који се често примењује за процену адекватности статистичких модела: параметри модела се формирају на основу задатог скупа

података, а потом се њихова адекватност процењује на другом скупу података [6]. У случају н-грама, ова идеја се своди на следећи поступак. Језички корпус се дели на:

- податке за обучавање модела — на основу којих се формирају вероватноће н-грама,
- податке за тестирање модела — на којима се оцењују вероватноће добијене у претходном кораку.

Овај методолошки приступ укључује и два потенцијална извора пристрасности (енгл. *bias*). Прво, ако подаци за обучавање и тестирање модела садрже исту реченицу, вероватноћа коју бисмо јој придружили приликом тестирања би била неоправдано велика, а тиме и неадекватна. Ова грешка се назива „обучавање на подацима за тестирање", и илуструје пристрасност која се у овом случају рефлектује у преувеличаним вероватноћама. Због тога, подаци за обучавање и тестирање језичких модела не смеју да садрже исте реченице.

Други извор пристрасности потиче из чињенице да уколико неки скуп података за тестирање користимо често, може се десити да почнемо да прилагођавамо наш модел његовим карактеристикама. Један од начина да се редукује оваква врста пристрасности је да се почетни језички корпус дели на три скупа:

- подаци за обучавање модела,
- подаци за развој модела — скуп података намењен за често тестирање параметара у процесу развоја модела, услед чега карактеристике овог скупа података нису сасвим непознате особи која развоја модел,
- подаци за тестирање модела — скуп „свежих" података за тестирање коначног модела, чије карактеристике нису познате особи која развија модел.

Приликом дељења језичког корпуса потребно је донети одлуку о величинама подскупова података. Са једне стране, велики скуп података за обучавање језичког модела омогућава адекватнију процену вероватноћа секвенци речи, и због тога тежимо да овај скуп буде што већи. Са друге стране, скуп података за тестирање не сме бити превише мали, јер би могао бити нерепрезентативан. Због тога, приликом дељења корпуса тежимо да изаберемо минимални скуп података за тестирање, који је ипак довољно велики да омогући задовољавајуће тестирање језичког модела (тј., да покаже статистички значајну разлику између два језичка модела). У практичним применама, подаци за тренинг често чине 80% језичког корпуса, а подаци за развој и тестирање по 10%.

У овом поглављу смо примарно разматрали н-граме на нивоу речи. Међутим, ови језички модели се могу посматрати, на концептуално

исти начин, и на нивоу појединачних знакова. Ово је урађено у поглављу 7.

4.5 Задаци

Задатак 4.1. Зашто је вероватније да се реч „дан" јави после секвенце речи „данас је леп" него само после речи „леп"?

Задатак 4.2. Нека су дати:

- корпус: „матрица трансформације садржи више мањих матрица",
- речник познатих термина: {матрица, трансформације}.

Користећи биграме, израчунајте вероватноћу реченице „ово није матрица трансформације", користећи формулу са Лапласовом корекцијом. Прикажите поступак.

Задатак 4.3. Нека су дати:

- корпус који садржи три реченице: „вода вода није кисела вода", „на петом спрату нема подземних вода", „кисела јабука на попусту",
- речник познатих термина: {вода, кисела, на}.

Користећи биграме, израчунајте вероватноћу реченице „вода на слици је кисела вода", користећи формулу са Лапласовом корекцијом. Прикажите поступак.

Задатак 4.4. Зашто Лапласова корекција смањује вредност ненултих вероватноћа?

Задатак 4.5. Нека су α и β два индексна термина. Објасните зашто је, у општем случају, језички модел који процењује да је вероватноћа биграма $P(\alpha|\beta)$ једнака нули неадекватан.

Задатак 4.6. Објасните зашто подаци за обучавање и тестирање н-грама не смеју да садрже исте реченице.

Задатак 4.7. Језички модел који у обзир узима две речи које непосредно претходе посматраној речи се назива *триграм*. Промените формуле (4.12), (4.16) и (4.17) тако да важе за триграме.

Литература

1. Biber D (1993) Representativeness in Corpus Design. Literary and Linguistic Computing 8(4): 243–257.

2. Brants T, Franz A (2006) Web 1T 5-gram Version 1 LDC2006T13. DVD. Philadelphia: Linguistic Data Consortium.
3. Chen SF, Goodman J (1998) An empirical study of smoothing techniques for language modeling. Tech. rep. TR-10-98, Computer Science Group, Harvard University.
4. Franz A, Brants T (2006) All Our N-gram are Belong to You. Google Research Blog https://research.googleblog.com/2006/08/all-our-n-gram-are-belong-to-you.html. Cited 28 Oct 2016.
5. Gale WA, Church KW (1994) What is wrong with adding one? In: Oostdijk N, de Haan P (eds.) Corpus-Based Research into Language, pp. 189–198, Rodopi.
6. Jurafsky D, Martin JH (2009) Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition (2ed.), Prentice Hall.
7. Katz S (1987) Estimation of probabilities from sparse data for the language model component of a speech recognizer. IEEE Transactions on Acoustics, Speech, and Signal Processing, 35(3), pp. 400–401.
8. Kneser R, Ney H (1995) Improved backing-off for M-gram language modeling. In: ICASSP-95, Vol. 1, pp. 181–184.
9. Sinclair J (2005) Corpus and Text — Basic Principles. In: Wynne M (ed) Developing Linguistic Corpora: a Guide to Good Practice. Oxford: Oxbow Books.
10. Tognini-Bonelli E (2001) Corpus Linguistics at Work. John Benjamins, Amsterdam.

Поглавље 5
Исправљање словних грешака у корисничким упитима

Кориснички упити нису увек исправно формулисани. Као илустрацију ове тврдње, Гугл је својевремено објавио[1] листу словних грешака у корисничким упитима, задатих у периоду од три месеца од барем два различита корисника, које је систем интерпретирао као *britney spears* (Бритни Спирс). Анализом листе се добија да је приближно 23% корисника (тј., 146258 од 635199 корисника) унело упит са словним грешкама. Листа укључује 592 различита упита са словним грешкама, при чему су три најучесталија била: *brittany spears* (40134 корисника), *brittney spears* (36315 корисника) и *britany spears* (24342 корисника). Треба истаћи да је оволики удео упита са словним грешкама није специфичан за тему претраге. Бритни Спирс је од значаја за ово излагање само по томе што је у време објављивања извештаја представљала популарну тему претраге.

Чињенице које морамо да узмемо у обзир приликом дизајнирања система за проналажење информација на вебу је да су словне грешке у значајној мери присутне у корисничким упитима, и да просечног корисника не треба условљавати да увек задаје исправно формулисане упите. Словне грешке у упиту могу бити последица немара или незнања, и корисник их не мора увек бити свестан. Са друге стране, корисник очекује корисне резултате. Ако бисмо дизајнирали системе који враћају корисне резултате само за исправно формулисане упите, ризиковали бисмо да корисници не прихвате систем. У претходном примеру, овакав ригидни систем би 23% корисника оставио без жељених резултата. Зато је један од захтева за системе за проналажење информација да могу да изврше аутоматско исправљање словних грешака у корисничким упитима.

Аутоматско исправљање словних грешака има широк контекст употребе, а у разматрањима која следе ћемо се примарно, и без губитка општости, фокусирати на област проналажења информација на вебу.

[1] По наводима других аутора [7], овај извештај је изворно био доступан на адреси http://www.google.com/jobs/britney.html. У време писања ове књиге, извештај више није био доступан на наведеној адреси, али су његове копије биле доступне на вебу, и заинтересовани читалац ће их лако пронаћи.

Основна идеја се може представити на следећи начин: задатак система је да термин у корисничком упиту који садржи словне грешке замени прикладно изабраним исправним термином из речника индексних термина који му стоји на располагању. Проналажење прикладног термина из речника се заснива на два принципа. Први принцип је да се неисправни термин замени исправним термином који му је *најсличнији*. Појам сличности између стрингова је интуитивно јасан, нпр., можемо рећи да је неисправном термину „*тран*“ сличнији придев „*стран*“ од именице „*транзистор*“, просто због тога што је релативна дужина заједничке секвенце знакова већа у првом случају (тј., секвенца знакова „*стран*“ чини 80% секвенце „*стран*“, и 40% секвенце „*транзистор*“). У општем случају, поређење стрингова није овако тривијално, а у секцији 5.1 ћемо детаљније размотрити једну меру сличности између стрингова. Други принцип је да ако у речнику индексних термина постоји више термина који су подједнако или приближно слични корисниковом неисправном термину, систем бира онај који процени као *највероватнији*. На пример, претпоставимо да за неисправни унос „*ершка*“ постоје два подједнако слична исправна термина у речнику: „*Грчка*“ и „*грешка*“. Систем би требало да изабере онај који процени као вероватнији у датом контексту претраге, што ћемо размотрити у секцији 5.2.

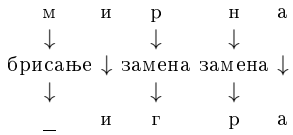
У претходном пасусу смо нагласили да систем прави разлику између исправних термина и термина са словним грешкама. Важно је да читалац не стекне погрешни утисак да систем заиста проверава правопис корисничких упита. Са становишта дизајна софтвера, разлика између исправних и неисправних термина је дефинисана формално: термини који се налазе у речнику индексних термина се сматрају исправним (чак и кад нису правописно исправни), док се термини који се налазе ван речника сматрају неисправним (чак и кад јесу правописно исправни). Овај прагматични приступ претпоставља да су документа у колекцији која се претражује правописно исправна у довољној мери да се може сматрати да речник индексних термина садржи исправне форме. Да се подсетите поступка генерисања речника индексних термина, погледајте поглавље 3.

5.1 Минимално растојање између стрингова

За дефинисање мере сличности стрингова можемо да искористимо општији појам минималног растојања између стрингова [5, 7, 11, 6]. Минимално растојање између два стринга (енгл. *minimum edit distance*) једнако је минималном броју операција уметања знака, брисања знака, или

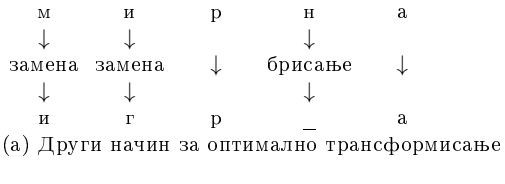
замене једног знака другим, потребних да се један стринг трансформише у други. Што је минимално растојање између два стринга мање, стрингови су сличнији.

Пример 5.1. Размотримо растојање између стрингова „мирна“ и „игра“. Минимално растојање између ова два стринга је 3, јер су потребне три операције да би се један стринг трансформисао у други (в. сл. 5.1).

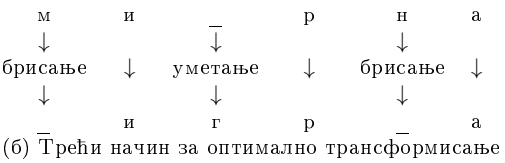


Слика 5.1 Пример оптималног трансформисања једног стринга у други.

У општем случају, постоји више начина да се један стринг трансформише у други, користећи исти број операција. Сл. 5.2 приказује још два начина да се стринг „мирна“ преведе у стринг „игра“ користећи минимални број операција.



(а) Други начин за оптимално трансформисање



(б) Трећи начин за оптимално трансформисање

Слика 5.2 Више оптималних начина да се један стринг трансформише у други.

Минимално растојање између стрингова које смо разматрали у претходном примеру се назива и Левенштајново растојање (енгл. *Levenshtein distance*, [6]). Левенштајново растојање је специјални случај растојања између стрингова, у ком се подразумева да операције уметања, брисања и замене знакова подједнако доприносе растојању између стрингова, и да је довољно само утврдити минимални број различитих операција потребних да се један стринг трансформише у други. У општем случају, овим операцијама се могу доделите произвољне „тежине“, што ћемо касније и објаснити, а засад ћемо се фокусирати на Левенштајново растојање. Два питања су нам од интереса: Како се одређује минимални број операција за трансформацију једног стринга у други? Како се проналазе све могуће трансформације са минималним бројем операција?

Као одговор на прво питање, дефинисаћемо алгоритам који израчунава Левенштајново растојање између два дата стринга. Улазни аргументи за овај алгоритам су:

- стринг s_1, дужине m знакова,
- стринг s_2, дужине n знакова.

Стринг s_1 ћемо звати изворни стринг, а s_2 одредишни стринг. Резултат алгоритма је минимални број операција потребних да се изворни стринг трансформише у одредишни. За дате стрингове се креира матрица растојања D, димензија $(m+1)\times(n+1)$. Ова матрица садржи по једну врсту за сваки знак изворног стринга s_1, и по једну колону за сваки знак одредишног стринга s_2. Прва врста и прва колона матрице растојања се иницијализују у складу са формулом:

$$(\forall\, 0 \le i \le m)\; D(i,0) = i\,, \tag{5.1}$$

$$(\forall\, 0 \le j \le n)\; D(0,j) = j\,. \tag{5.2}$$

Остали елементи матрице растојања се израчунавају по следећој рекурзивној формули:

$$(\forall\, 1 \le i \le m,\; 1 \le j \le m) \tag{5.3}$$

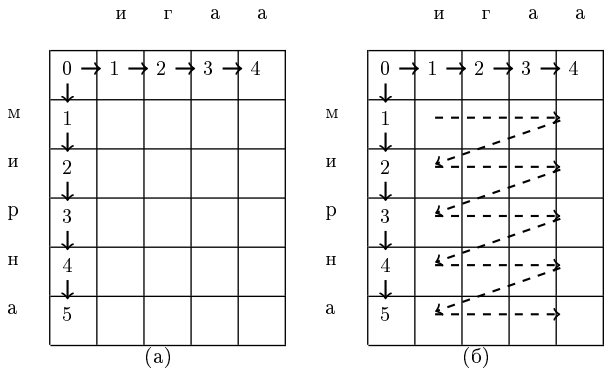
$$D(i,j) = min \begin{cases} D(i-1,j)+1 \\ D(i,j-1)+1 \\ D(i-1,j-1)+ \begin{cases} 1, & s_1(i) \neq s_2(j) \\ 0, & s_1(i) = s_2(j) \end{cases} \end{cases}.$$

Вредност елемента $D(m,n)$ је Левенштајново растојање између стрингова s_1 и s_2.

Пример 5.2. Илуструјмо алгоритам за израчунавање Левенштајновог растојања између стрингова „мирна“ и „игра“. За дате стрингове, креира се празна матрица растојања као на сл. 5.3(а). Прва врста и прва колона се попуне сукцесивним целим бројевима, почевши од 0. Унете бројеве ћемо повезати стрелицама које означавају прелазе између ћелија, а ускоро ћемо видети да ови прелази означавају операције над знаковима. Остатак матрице се попуњава слева надесно, и одозго надоле, по рекурзивној формули (5.3). Смер попуњавања је представљен испрекиданим стрелицама на сл. 5.3(б).

Претпоставимо да желимо да израчунамо вредност елемента x који се налази на пресеку врсте додељене знаку $s_1(i)$ и колоне додељене знаку $s_2(j)$. Његова вредност се дефинише као најмања од три вредности које се израчунавају на основу три суседна елемента матрице:

- g — који се налази непосредно изнад x,
- l — који се налази са леве стране x,
- d — који се налази дијагонално лево изнад x (в. сл. 5.4(а))



Слика 5.3 (а) Иницијализација и (б) смер попуњавања матрице растојања.

Пошто се матрица попуњава слева надесно, и одозго надоле, то значи да су вредности елемената g,d,l већ претходно израчунате. Вредност x се израчунава по рекурзивној формули (5.3). Ако важи $s_1(i) = s_2(j)$ (нпр., в. сл. 5.4(б)), онда је

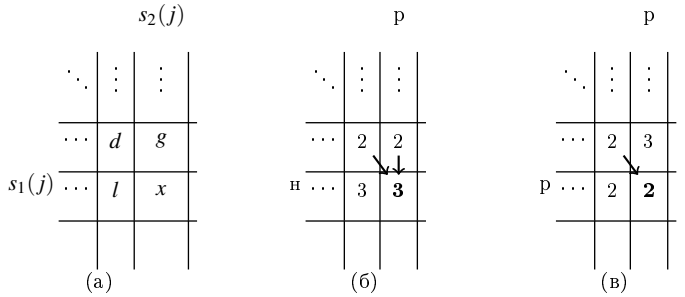
$$x = min \begin{cases} g+1 \\ l+1 \\ d \end{cases}, \tag{5.4}$$

а у супротном (нпр., в. сл. 5.4(в)):

$$x = min \begin{cases} g+1 \\ l+1 \\ d+1 \end{cases}. \tag{5.5}$$

За потребе анализе извршавања алгоритма, осим што израчунавамо вредност x, евидентираћемо и који од елемената g,d,l су учествовали у генерисању минималне вредности. У примеру приказаном на сл. 5.4(б), минимална вредност се добија из два елемента, g и d, а ови прелази су евидентирани стрелицама. У примеру приказаном на сл. 5.4(в), минимална вредност се добија из само једног елемента, d, што је такође адекватно евидентирано.

Попуњена матрица растојања, са назначеним прелазима, је дата на сл. 5.5(а). Последњи елемент матрице представља Левенштајново растојање између стрингова „мирна“ и „игра“. Пошто смо одредили растојање између стрингова, питање које нам је сад од интереса је како да одредимо минималну секвенцу операција која ће први стринг да трансформише у други. Посматрајући евидентиране прелазе у матрици, можемо да пронађемо три путање које воде од горњег левог елемента матрице до доњег десног елемента (в. сл. 5.5(б), (в), (г)). Свака од ових путања представља једну трансформацију стринга „мирна“ у стринг „игра“, а сваки прелаз



Слика 5.4 Израчунавање елемента матрице растојања.

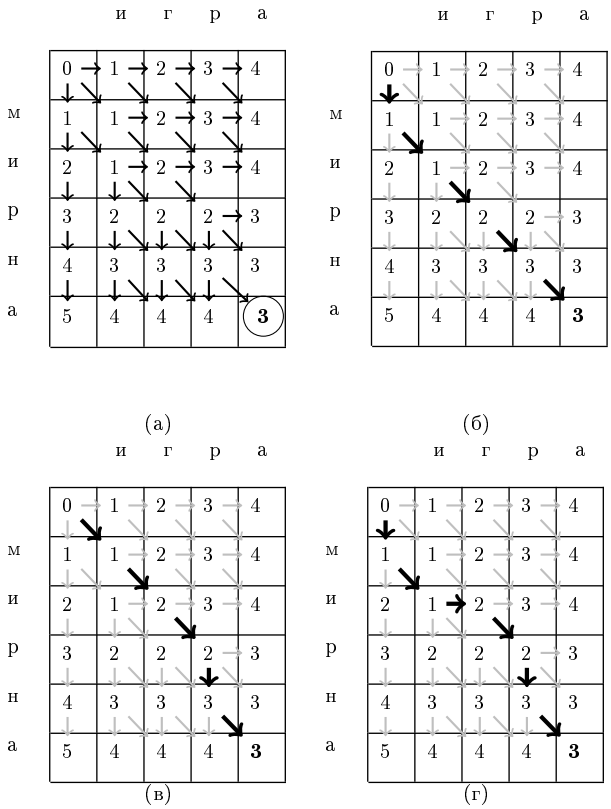
између две ћелије различитих вредности представља једну операцију. Интерпретирање појединачних прелаза се врши на следећи начин. Нека дата стрелица завршава у ћелији c. Тада важи:

- стрелица надоле ($\downarrow$) означава брисање знака из првог стринга коме припада врста у којој се налази ћелија c,
- стрелица надесно ($\rightarrow$) означава уметање знака из другог стринга коме припада колона у којој се налази ћелија c,
- дијагонална стрелица ($\searrow$) означава да се знак из првог стринга коме припада врста у којој се налази ћелија c мења знаком из другог стринга коме припада колона у којој се налази ћелија c. Приметите да ово укључује и случајеве када се знак изворног стринга мења истим знаком одредишног стринга. У том случају се операција замене знакова не извршава.

Ако интерпретирамо путање приказане на сл. 5.5(б), 5.5(в) и 5.5(г), видећемо да одговарају трансформацијама приказаним на сл. 5.1, 5.2(а) и 5.2(б), респективно. Све путање садрже минимални број операција.

Алгоритам за рачунање Левенштајновог растојања између стрингова припада класи алгоритама који користе приступ динамичког програмирања [2]. Основна идеја динамичког програмирања је да се проблем декомпонује на мање проблеме истог типа, да би се комбиновањем њихових решења добило решење полазног проблема. У овом конкретном алгоритму, приступ динамичког програмирања се рефлектује кроз:

- концептуализацију да вредност ћелије матрице $[i, j]$ представља Левенштајново растојање између првих i знакова изворног стринга и првих j знакова одредишног стринга,
- рекурзивно правило дато формулом (5.3), по коме се вредност ћелије матрице $[i, j]$ рачуна као минимална вредност три могуће путање које воде до дате ћелије.



Слика 5.5 (а) Попуњена матрица растојања, и (б), (в), (г) три путање које воде од првог до последњег елемента матрице.

5.1.1 Тежине операција

Већ смо напоменули да израчунавање Левенштајновог растојања између стрингова подразумева да операције уметања, брисања и замене знакова подједнако доприносе растојању између стрингова, и да је довољно само утврдити минимални број различитих операција потребних да се један стринг трансформише у други. Ова претпоставка се у конкретном алгоритму рефлектује кроз чињеницу да је свим операцијама придружена иста „тежина“ (тј., вредност 1, в. алгоритам (5.3)). Међутим, тежине операција се могу мењати у складу са контекстом примене алгоритма.

Пример 5.3. Операција замене знака c_1 знаком c_2 се састоји од брисања знака c_1 и уметања знака c_2. Због тога би се тежина операције замене могла дефинисати као збир тежина операција брисања и уметања.

Пример 5.4. Тежине операција брисања, уметања и замене се могу дефинисати засебно за сваки симбол, односно пар симбола. Нпр., због распореда тастера на тастатури, вероватноћа да ће корисник грешком притиснути знак M уместо знака N је већа него да ће притиснути знак Q уместо знака N. Зато би тежина замене знакова M и N требало да буде мања од тежине замене знакова Q и N.

Увођењем тежина операција се уопштава Левенштајнов алгоритам. У општем случају, рекурзивно правило за израчунавање минималног растојања између стрингова s_1 и s_2 се формулише на следећи начин.

- Иницијализација матрице растојања:

$$D[0,0] = 0, \tag{5.6}$$

$$(\forall\, 0 < i \leq m)\; D(i,0) = D(i-1,0) + \texttt{тежина брисања знака}\, s_1[i]\,, \tag{5.7}$$

$$(\forall\, 0 < j \leq n)\; D(0,j) = D(0,j-1) + \texttt{тежина уметања знака}\, s_2[j]\,. \tag{5.8}$$

- Рекурзивно правило за попуњавање матрице растојања:

$$(\forall\, 1 \leq i \leq m,\, 1 \leq j \leq m) \tag{5.9}$$

$$D(i,j) = min \begin{cases} D(i-1,j) + \texttt{тежина брисања знака}\, s_1[i] \\ D(i,j-1) + \texttt{тежина уметања знака}\, s_2[j] \\ D(i-1,j-1) + \texttt{тежина замене знака}\, s_1[i]\, \texttt{знаком}\, s_2[j] \end{cases}.$$

Ова рекурзивна дефиниција се може представити у итеративном маниру, а одговарајући алгоритам у псеудокоду је дат на сл. 5.6. Овај алгоритам попуњава матрицу растојања и израчунава минимално растојање између датих стрингова, али не евидентира експлицитно прелазе између ћелија матрице, тј., потенцијалне операције над знаковима. Читаоцу се оставља да модификује алгоритам тако да укључује и ову функционалност (в. задатак 5.5).

Рачунарски алгоритам, који за дату матрицу растојања са евидентираним прелазима проналази све могуће путање, врши претрагу од последњег елемента матрице (тј., доњег десног елемента) ка првом елементу матрице (тј., горњем левом елементу). Овакав начин претраживања се назива претраживање уназад (енгл. *backtracking*).

5.1.2 Нормализовано Левенштајново растојање

Подсетимо се да минимално растојање између стрингова користимо као меру сличности између стрингова. Другим речима, што је минимално

```
function minimalno_rastojanje(s1,s2)
  m = dužina(s1)
  n = dužina(s2)
  D = matrica dimenzija (m + 1)×(n + 1)
  D[0,0] = 0
  for i = 1 to m do
    D[i,0] = D[i−1, 0] + težina_brisanja(s1[i])
  for j = 1 to n do
    D[0,j] = D[0,j−1] + težina_umetanja(s2[j])
  for i = 1 to m do
    for j = 1 to n do
      D[i,j] = min(D[i−1,j] + težina_brisanja(s1[i]),
                   D[i,j−1] + težina_umetanja(s2[j]),
                   D[i−1,j−1] + težina_zamene(s1[i],s2[j]))
  return D[m,n];
```

Слика 5.6 Итеративни алгоритам за израчунавање минималног растојања између стрингова.

растојање између два стринга мање, стрингови су сличнији. Имајући то на уму, размотримо следећи пример.

Пример 5.5. Левенштајново растојање између стрингова „ти“ и „он“ је 2, а између стрингова „оториноларингологија“ и „оториноларинголог“ је 3. Ако бисмо посматрали само апсолутни број операција потребних да се један стринг преведе у други, испало би да су стрингови у првом пару сличнији од стрингова у другом пару. Ипак, јасно је да су стрингови из другог пара сличнији, и да добијена растојања не рефлектују ту чињеницу.

Један од начина да се превазиђе очигледни недостатак уочен у претходном примеру је да се изврши нормализација Левенштајновог растојања између стрингова. Нека су:

- s_1 — стринг дужине m знакова,
- s_2 — стринг дужине n знакова,
- $d(s_1, s_2)$ — Левенштајново растојање између стрингова s_1 и s_2.

Нормализовано Левенштајново растојање можемо да дефинишемо као:

$$d_N(s_1, s_2) = \frac{d(s_1, s_2)}{m+n} \,. \tag{5.10}$$

Пример 5.6. Нормализовано Левенштајново растојање између стрингова „ти“ и „он“ је:

$$d_N(\text{ти}, \text{он}) = \frac{d(\text{ти}, \text{он})}{2+2} = \frac{1}{2} \,, \tag{5.11}$$

а између стрингова „оториноларингологија“ и „оториноларинголог“:

$$d_N(\text{оториноларингологија}, \text{оториноларинголог}) = \frac{d(\text{оториноларингологија}, \text{оториноларинголог})}{20+17} = \frac{3}{37}. \tag{5.12}$$

Нормализовано растојање између другог пара стрингова је сад мање ($\frac{3}{37} < \frac{1}{2}$), што је у складу са чињеницом да је овај пар стрингова сличнији. Ипак, треба напоменути да, иако је формула 5.10 интуитивна за нормализацију Левенштајновог растојања, она није адекватна за нормализацију минималног растојања између стрингова у општем случају који дозвољава произвољне тежине операција.

5.1.3 Жакаров коефицијент сличности

На концептуалном нивоу, систем за проналажење информација може да користи минимално растојање између стрингова да би за термин у корисничком упиту који садржи словне грешке пронашао исправни термин из речника индексних термина који му је најсличнији. Међутим, израчунавање минималних растојања за сваки термин у великом речнику индексних термина је временски захтевно, и због тога није практично. Да би се превазишао овај проблем, претрага термина по сличности се врши у две фазе:

- У првој фази се издвајају они термини из речника који су „потенцијално“ слични неисправном термину који је задао корисник. Алгоритам који се користи за иницијалну процену сличности стрингова представља компромис између квалитета процене сличности и временске ефикасности. Другим речима, иницијална процена сличности би требало да се извршава брже од рачунања минималног растојања између стрингова, чак и ако даје мање прецизну процену сличности.
- У другој фази се израчунавање минималног растојања извршава на подскупу индексних термина издвојеном у првој фази. Ово поређење стрингова је прецизније, али и временски захтевније од иницијалног поређења у првој фази. Временска захтевност поређења појединачних парова стрингова се компензује мањим скупом стрингова над којима се поређење врши.

За временски ефикасну, иницијалну процену сличности два термина може се применити Жакаров (Jaccard) коефицијент сличности два скупа [4], A и B, који се дефинише:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}, \tag{5.13}$$

тј., као однос броја елемената пресека скупова A и B, и броја елемената њихове уније. Да би се Жакаров коефицијент употребио за процену сличности термина, сваки термин ћемо представити као скуп свих *биграма*[2] које садржи, тј., скуп свих секвенци два суседна знака које су садржане у датом термину.

Пример 5.7. Термин „авокадо“ садржи следеће биграме $A = \{\text{ав}, \text{во}, \text{ок}, \text{ка}, \text{ад}, \text{до}\}$, а термин „адвокат“ $B = \{\text{ад}, \text{дв}, \text{во}, \text{ок}, \text{ка}, \text{ат}\}$. Жакаров коефицијент сличности ова два термина је:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|\{\text{ад}, \text{во}, \text{ок}, \text{ка}\}|}{|\{\text{ав}, \text{во}, \text{ок}, \text{ка}, \text{ад}, \text{до}, \text{дв}, \text{ат}\}|} = \frac{4}{8}\,, \tag{5.14}$$

Пример 5.8. Приликом представљања термина као скупова биграма, треба имати у виду да биграми у термину могу да се понављају, и да скуповна представа укључује само различите биграме. Нпр., стринг „лала“ садржи 3 биграма, од којих су само 2 различита: $\{\text{ла}, \text{ал}\}$ (в. задатак 5.7).

Жакаров коефицијент се увек налази у опсегу $[0,1]$, без обзира на скупове који се пореде. Уколико су скупови исти, коефицијент њихове сличности је 1, а уколико су дисјунктни, коефицијент сличности је 0. Ова особина нам допушта да формализујемо поступак редуковања речника на подскуп термина који су „потенцијално“ слични неисправном термину који је задао корисник. За дати корисников термин, из речника се издвајају само они термини за које важи да је вредност Жакаровог коефицијента сличности са корисниковим термином већи од унапред задате вредности $t \in (0,1)$. Минимално растојање између стрингова се потом израчунава само за термине из издвојеног подскупа.

5.2 Контекстно зависно исправљање словних грешака

До сада смо разматрали растојање између стрингова као једину меру њихове сличности. Међутим, језички контекст у коме се врши исправљање термина са словним грешкама такође утиче на процену сличности стрингова. На пример, претпоставимо да за неисправни унос „*гршка*“ постоје два подједнако удаљена термина у речнику: „*Грчка*“ и „*грешка*“. Систем би могао да изабере онај који процени као вероватнији у датом контексту претраге. У општем случају, исправљање словне грешке у термину s_x се може свести на проналажење термина $\hat{s}$ из речника индексних термина V за који је *највероватније* да представља исправни запис [8, 9], тј.:

$$\hat{s} = \underset{s \in V}{\operatorname{argmax}}\, P(s|s_x)\,. \tag{5.15}$$

[2] Појам биграма смо детаљније разматрали у поглављу 4.

Међутим, вероватноћу $P(s|s_x)$ није лако директно проценити. На пример, претпоставимо да за неисправни унос $s_x =$ *„профодија“* постоје два слична исправна термина у речнику: $s_1 =$ *„прозодија“* и $s_2 =$ *„професија“*. Да бисмо проценили која је од вероватноћа $P(s_1|s_x)$ и $P(s_2|s_x)$ већа, морамо узети у обзир два фактора. Први фактор се односи на минамално растојање између стрингова. Левенштајново растојање између стрингова s_1 и s_x је мање од Левенштајновог растојања између стрингова s_2 и s_x, што би могло да упути на закључак да термин s_1 представља вероватнијег кандидата, тј., $P(s_1|s_x) > P(s_2|s_x)$. Други фактор се односи на учесталост јављања речи у језику. Термин s_2 се знатно чешће употребљава од термина s_1, па би се могло закључити да је већа вероватноћа да је корисник намеравао да зада термин s_2, тј., $P(s_1|s_x) < P(s_2|s_x)$. За адекватно процењивање вероватноће, потребно је комбиновати ова два контрадикторна фактора, што се може постићи применом Бајесове теореме (Bayes, [1]). Нека су A и B случајни догађаји, при чему важи $P(B) \neq 0$. По Бајесовој теореми, важи:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}\,, \tag{5.16}$$

где су:

- $P(A)$ — вероватноћа реализације догађаја A,
- $P(B)$ — вероватноћа реализације догађаја B,
- $P(A|B)$ — вероватноћа реализације догађаја A, под условом да се реализовао догађај B,
- $P(B|A)$ — вероватноћа реализације догађаја B, под условом да се реализовао догађај A.

Применом Бајесове теореме на формулу (5.15) се добија:

$$\hat{s} = \underset{s \in V}{\operatorname{argmax}}\, P(s|s_x) = \underset{s \in V}{\operatorname{argmax}}\, \frac{P(s_x|s)P(s)}{P(s_x)}\,. \tag{5.17}$$

Анализирајмо формулу (5.17):

- Вероватноћа $P(s)$, тј., вероватноћа термина s, се најједноставније може проценити као учесталост термина s у датом језику. Ако желимо да проценимо вероватноћу јављања термина s у контексту конкретног корисничког упита, можемо употребити н-граме, као што је објашњено у поглављу 4.
- Вероватноћа $P(s_x|s)$, тј., вероватноћа да корисик грешком зада термин s_x уместо термина s, се може проценити на основу корпуса словних грешака (нпр., [10]). Уколико корпус словних грешака није доступан за дати језик, ова вероватноћа се може интуитивно интерпретирати као обрнуто пропорционална минималном растојању између стрингова s_x и s.

- Вероватноћа $P(s_x)$, тј., вероватноћа погрешног термина s_x, се не може проценити (в. задатак 5.8), али то није ни потребно. У формули (5.17) се тражи термин s из речника за који је вредност $\frac{P(s_x|s)P(s)}{P(s_x)}$ максимална. Вероватноћа $P(s_x)$ није позната, али је константна, без обзира на избор термина s, што значи да је довољно наћи термин s за који је вредност $P(s_x|s)P(s)$ максимална.

Формулу (5.17) сада можемо да упростимо:

$$\hat{s} = \underset{s \in V}{\operatorname{argmax}} P(s_x|s)P(s) \,. \tag{5.18}$$

при чему све вероватноће са десне стране знака једнакости можемо сматрати познатим. Ова идеја представља основ за комплексније приступе исправљању словних грешака у корисничким уносима [12, 3].

5.3 Задаци

Задатак 5.1. За дате стрингове „село“ и „осло“ израчунајте Левенштајново растојање и пронађите све могуће секвенце операција које трансформишу први стринг у други. Прикажите матрицу растојања са евидентираним прелазима између ћелија.

Задатак 5.2. Покажите да Левенштајново растојање између стрингова s_1, дужине m знакова, и s_2, дужине n знакова, не може бити веће од $max\{m,n\}$.

Задатак 5.3. Претпоставите да операција замене знакова има тежину 2, а да операције брисања и уметања знакова имају тежину 1. Израчунајте минимално растојање између стрингова из задатка 5.1, и пронађите све могуће секвенце операција које трансформишу први стринг у други. Прикажите матрицу растојања са евидентираним прелазима између ћелија.

Задатак 5.4. Претпоставите да су изворни и одредишни стринг заменили места. (а) Да ли минимално растојање између стрингова остаје исто? (б) Како се мења матрица растојања између два стригна? (в) Како се мењају секвенце операција које преводе један стринг у други.

Задатак 5.5. Модификујте алгоритам дат на сл. 5.6 тако да укључује и функционалност евидентирања прелаза између ћелија матрице.

Задатак 5.6. За матрицу Левенштајновог растојања приказану на сл. 5.7, пронађите све путање почевши од последњег елемента матрице.

Задатак 5.7. Израчунајте Жакарков коефицијент сличности стрингова „веверица“ и „царица“.

		т	а	ч	к	а
	0	1	2	3	4	5
з	1	1	2	3	4	5
н	2	2	2	3	4	5
а	3	3	2	3	4	4
ч	4	4	3	2	3	4
к	5	5	4	3	2	3
а	6	6	5	4	3	2

Слика 5.7 Матрица растојања у задатку 5.6

Задатак 5.8. Зашто у општем случају не можемо да проценимо вероватноћу $P(s_x)$, тј., вероватноћу погрешног термина s_x, у формули (5.17)?

Литература

1. Bayes T (1764) An Essay Toward Solving a Problem in the Doctrine of Chances. Philosophical Transactions of the Royal Society of London 53, pp. 370–418.
2. Bellman R (1957). Dynamic Programming. Princeton University Press, Princeton, NJ.
3. Choudhury M, Thomas M, Mukherjee A, Basu A, Ganguly N (2007) How Difficult is it to Develop a Perfect Spell-checker? A Cross-linguistic Analysis through Complex Network Approach. In: TextGraphs-2: Graph-Based Algorithms for Natural Language Processing, pp. 81–88, Rochester, April 2007.
4. Jaccard P (1912) The Distribution of the Flora in the Alpine Zone, New Phytologist 11:37–50. Original in Revue geiiérale des Sciences, 15th December, 1907, pp. 961–967.
5. Jurafsky D, Martin JH (2009) Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition (2ed.), Prentice Hall.
6. Levenshtein VI (1966) Binary codes capable of correcting deletions, insertions, and reversals. Cybernetics and Control Theory, 10(8): 707—710. Original in Doklady Akademii Nauk SSSR 163(4): 845—848, 1965.
7. Manning CD, Raghavan P, Schütze H (2008) Introduction to Information Retrieval, Cambridge University Press.
8. Norvig P (2016) How to Write a Spelling Corrector http://norvig.com/spell-correct.html. Cited 20 Oct 2016.
9. Norvig P (2009) Natural Language Corpus Data. In: Segaran T, Hammerbacher J (eds) Beautiful Data. The Stories Behind Elegant Data Solutions, O'Reilly Media.
10. Roger M (1985) Birkbeck spelling error corpus. University of Oxford Text Archive http://ota.ox.ac.uk/headers/0643.xml. Cited 20 Oct 2016.
11. Wagner RA, Fischer MJ (1974) The string-to-string correction problem. Journal of the Association for Computing Machinery, 21, 168–173.
12. Whitelaw C, Hutchinson B, Chung GY, Ellis G (2009) Using the Web for Language Independent Spellchecking and Autocorrection. In: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pp. 890-–899, Singapore, 6–7 August 2009.

Поглавље 6
Рангирање резултата претраге

У поглављу 2 смо истакли следеће недостатке линеарног претраживања великих колекција докумената [7]:

- не омогућава ефикасну претрагу великих колекција,
- није погодно за обраду упита са флексибилним условима за претрагу,
- није погодно за рангирање пронађених докумената у односу на то колико су релевантни за корисника.

Прва два недостатка су превазиђена увођењем инвертованог индекса, за чију конструкцију смо користили сортиране листе, због ефикасног извршавања скуповних операција. Међутим, описани приступ не решава проблем неадекватног рангирања резултата претраге. Наиме, код овакве организације инвертованог индекса, крајњи резултат обраде упита је такође сортирана листа. То значи да је резултујућа листа докумената која се приказује кориснику сортирана према шифрама докумената. Чест је случај да је број резултујућих докумената знатан, што значи да не можемо очекивати да ће корисник прегледати све резултате. Зато је један од основних захтева за системе за проналажење информација на вебу да резултате рангирају према релевантности, тј., прво најрелевантније, а потом мање релевантне.

У реалним применама, постоји мноштво фактора који могу да утичу на релевантност појединачних веб-страница. У овом поглављу ћемо размотрити један од њих, који се заснива на анализи графа који представља скуп веб-страница.

6.1 Изабрани појмови из теорије графова

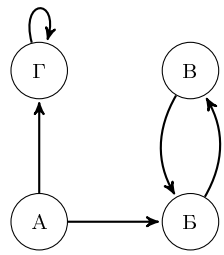
Пре него што пређемо на питање рангирања резултата претраге, кроз примере ћемо се подсетити основних појмова из теорије графова који су од значаја за разумевање концепата који следе [2]. Формално, граф G

се дефинише као уређени пар (V,E), где су: V непразни скуп, E бинарна релација над V ($E \subseteq V \times V$). Интуитивно, граф чине скуп чворова (тј., елементи скупа V) и скуп грана (тј., елементи скупа E) које повезују чворове.

Пример 6.1. Посматрајмо граф $G_1 = (V_1, E_1)$, где су:

- чворови $V_1 = \{\text{А}, \text{Б}, \text{В}, \text{Г}\}$,
- гране $E_1 = \{(\text{А}, \text{Б}), (\text{А}, \text{Г}), (\text{Б}, \text{В}), (\text{В}, \text{Б}), (\text{Г}, \text{Г})\}$.

Пример визуелног представљања овог графа дат је на сл. 6.1. Приказани граф је *усмерен* — свака грана има свој смер, који је назначен стрелицом. У овом примеру, чворови А и Б су повезани у смеру од А ка Б (не и обрнуто), док су чворови Б и В повезани у оба смера, итд.



Слика 6.1 Пример визуелног представљања усмереног графа.

Пример 6.2. Код *неусмереног* графа, смер грана није назначен, и подразумева се да свака грана повезује чворове у оба смера. Неусмерени граф $G_2 = (V_2, E_2)$ представљен на сл. 6.2(а) је само наизглед исти као граф $G_1 = (V_1, E_1)$ на сл. 6.1. Иако су им скупови чворова исти, тј.,

$$V_1 = V_2 \,,$$

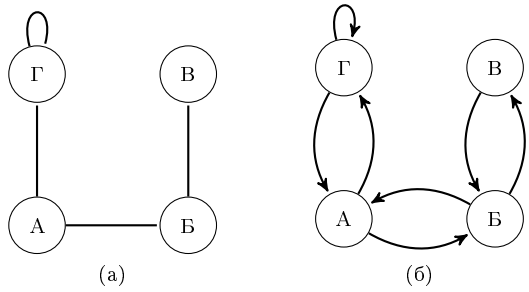
скупови грана им се разликују:

$$E_1 \subset E_2 = \{(\text{А}, \text{Б}), (\text{Б}, \text{А}), (\text{А}, \text{Г}), (\text{Г}, \text{А}), (\text{Б}, \text{В}), (\text{В}, \text{Б}), (\text{Г}, \text{Г})\} \,.$$

Алтернативни приказ графа G_2 је дат на сл. 6.2(б). Због лакше репрезентације, неусмерене графове ћемо представљати на први начин.

Пример 6.3. Да ли користимо усмерене или неусмерене графове зависи од природе релација које желимо да представимо. Да бисмо ово илустровали посматрајмо следеће примере:

- Нека чворови графа представљају веб-странице, а гране хипервезе између њих. Хипервеза представља асиметричну релацију — чињеница



Слика 6.2 Примери визуелног представљања неусмереног графа.

да страница А указује на страницу Б не имплицира да страница Б указује на страницу А. Зато користимо усмерени граф за представљање хипервеза између веб-страница. Нпр., ако страница А садржи хипервезу ка страници Б, смер гране која их повезује ће бити од А ка Б. У овом контексту, граф приказан на сл. 6.1 се интерпретира на следећи начин:

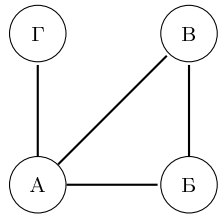
- страница А садржи хипервезе ка страницама Б и Г,
- странице Б и В садрже хипервезе једна ка другој,
- страница Г садржи хипервезу ка себи самој.

- Нека чворови графа представљају налоге на Фејсбуку, а гране везе „пријатељства“ између њих. Везе „пријатељства“ на Фејсбуку су симетричне релације — ако је налог А „пријатељ“ налогу Б, онда важи и да је Б „пријатељ“ А. Зато можемо да користимо неусмерени граф за представљање ових веза.
- Нека чворови графа представљају налоге на Твитеру, а гране везе „праћења“ између њих. Веза „праћења“ на Твитеру представља асиметричну релацију — чињеница да налог А „прати“ налог Б не имплицира да налог Б „прати“ налог А. Због тога можемо да користимо усмерени граф за представљање ових веза.

Пример 6.4. Степен чвора је једнак броју грана са којима је чвор повезан. Степени чворова за неусмерени граф су илустровани на сл. 6.3. За усмерени граф се дефинишу излазни и улазни степени чвора, као бројеви грана које излазе из чвора, односно улазе у њега. Ово је илустровано на сл. 6.4

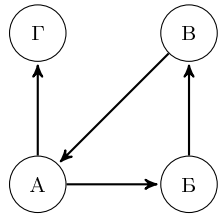
Пример 6.5. Још један начин за представљање графа $G = (V, E)$ је помоћу матрице повезаности $A_{G_{|V|\times|V|}}$.

$$A_G[i,j] = \begin{cases} 1, & (v_i, v_j) \in E \\ 0, & \text{у супротном} \end{cases}. \tag{6.1}$$



Чвор	Степен
А	3
Б	2
В	2
Г	1

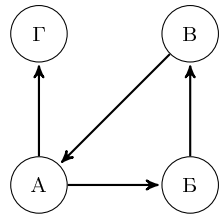
Слика 6.3 Степени чворова неусмереног графа.



Чвор	Улазни степен	Излазни степен
А	1	2
Б	1	1
В	1	1
Г	1	0

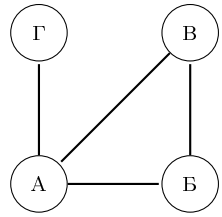
Слика 6.4 Степени чворова усмереног графа.

Матрица повезаности садржи по једну врсту и једну колону за сваки чвор графа. Уколико постоји грана од чвора v_i ка чвору v_j, елемент матрице који се налази на пресеку i-те врсте и j-те колоне добија вредност 1; у супротном 0. Сл. 6.5 и 6.6 илуструју матрице повезаности за усмерени, односно неусмерени граф. Може се уочити да је матрица повезаности за неусмерени граф симетрична у односу на главну осу.



$$A_G = \begin{array}{c} \\ \text{А} \\ \text{Б} \\ \text{В} \\ \text{Г} \end{array} \begin{array}{c} \begin{array}{cccc} \text{А} & \text{Б} & \text{В} & \text{Г} \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Слика 6.5 Матрица повезаности за усмерени граф.



$$A_G = \begin{array}{c} \\ \text{А} \\ \text{Б} \\ \text{В} \\ \text{Г} \end{array} \begin{array}{c} \begin{array}{cccc} \text{А} & \text{Б} & \text{В} & \text{Г} \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Слика 6.6 Матрица повезаности за неусмерени граф.

6.2 Пристрасност обиласка неусмереног графа на случајни начин

Обилазак графа на случајни начин (енгл. *random walk*) представља основу за разумевање критеријума за рангирање резултата претраге представљеног у овом поглављу [8, 3]. Да бисмо поједноставили овај концепт, прво ћемо размотрити обилазак неусмереног графа на случајни начин [6].

Нека је $G = (V,E)$ неусмерени, повезани граф. Обилазак графа G на случајни начин се може описати на следећи начин:

- Почетни чвор обиласка се бира према датој расподели вероватноћа. За сваки чвор v_i посматраног графа се дефинише вероватноћа да обилазак графа почиње од њега. Означимо ову вероватноћу са $P_0(v_i)$. Расподелу вероватноћа за избор почетног чвора ћемо представљати матрицом P_0, која садржи једну колону, и по једну врсту за сваки чвор у посматраном графу:

$$P_0 = \begin{bmatrix} P_0(v_1) \\ P_0(v_2) \\ \dots \\ P_0(v_n) \end{bmatrix} . \tag{6.2}$$

  Збир свих вероватноћа у матрици P_0 је једнак 1, тј.:

$$\sum_{v \in V} P_0(v) = 1 . \tag{6.3}$$

- Ако се налазимо у чвору v_i, следећи чвор се бира на случајни начин из скупа суседних чворова. Вероватноћа (директног) преласка из чвора v_i у чвор v_j се дефинише:

$$P_{v_i,v_j} = \begin{cases} \frac{1}{d(v_i)}, & (v_i,v_j) \in E \\ 0, & \text{у супротном,} \end{cases} \tag{6.4}$$

  где је: $d(v_i)$ — степен чвора v_i.

Пример 6.6. За неусмерени граф дат на сл. 6.3, општа форма матрице P_0 је дата на сл. 6.7(а). Различите вредности елемената ове матрице дефинишу различите расподеле вероватноћа за избор почетног чвора (в. сл. 6.7(б, в, г, д)):

- P_0^I — А је почетни чвор,
- P_0^{II} — Б је почетни чвор,
- P_0^{III} — вероватноћа да је А почетни чвор је $\frac{3}{4}$, вероватноћа да је Б почетни чвор је $\frac{1}{4}$,
- P_0^{IV} — сви чворови имају једнаке вероватноће ($\frac{1}{4}$) да буду почетни чворови.

$$P_0 = \begin{bmatrix} P_0(\text{А}) \\ P_0(\text{Б}) \\ P_0(\text{В}) \\ P_0(\text{Г}) \end{bmatrix} \quad P_0^I = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad P_0^{II} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad P_0^{III} = \begin{bmatrix} \frac{3}{4} \\ \frac{1}{4} \\ 0 \\ 0 \end{bmatrix} \quad P_0^{IV} = \begin{bmatrix} \frac{1}{4} \\ \frac{1}{4} \\ \frac{1}{4} \\ \frac{1}{4} \end{bmatrix}$$

(а) (б) (в) (г) (д)

Слика 6.7 Расподела вероватноћа за избор почетног чвора.

Пример 6.7. У складу са изразом (6.4), израчунајмо неке вероватноће прелаза за неусмерени граф приказан на сл. 6.3:

- ако смо у чвору А, са једнаким вероватноћама ($\frac{1}{3}$) прелазимо у Б, В и Г, тј.,
$$P_{\text{А,Б}} = P_{\text{А,В}} = P_{\text{А,Г}} = \frac{1}{3} ,$$
- ако смо у чвору Б, са једнаким вероватноћама ($\frac{1}{2}$) прелазимо у А и В, док директни прелаз у Г није могућ, тј.,
$$P_{\text{Б,А}} = P_{\text{Б,В}} = \frac{1}{2} , \quad P_{\text{Б,Г}} = 0 .$$

По аналогији са дефиницијом матрице P_0, нека матрица P_t садржи вероватноће да ће се алгоритам након t корака ($t > 0$) обиласка графа на случајни начин наћи у одређеном чвору:

$$P_t = \begin{matrix} \text{А} \\ \text{Б} \\ \text{В} \\ \text{Г} \end{matrix} \begin{bmatrix} P_t(\text{А}) \\ P_t(\text{Б}) \\ P_t(\text{В}) \\ P_t(\text{Г}) \end{bmatrix} . \tag{6.5}$$

За ову матрицу важи:

$$P_{t+1} = M^T \times P_t , \tag{6.6}$$

тј.,

$$P_t = (M^T)^t \times P_0 , \tag{6.7}$$

где су:

- $M = D \times A_G$ — матрица прелаза,
- M^T — транспонована матрица M,
- A_G — матрица повезаности,
- D — дијагонална матрица: $D_{[v_i,v_j]} = \begin{cases} \frac{1}{d(v_i)}, & v_i = v_j \\ 0, & v_i \neq v_j \end{cases}$,
- $d(v_i)$ — степен чвора v_i,
- P_0 — почетна расподела вероватноћа,

- t — број прелазака из чвора у чвор при обиласку графа на случајни начин.

Транспоновањем произвољне матрице A, димензија $m \times n$, се добија нова матрица A^T, димензија $n \times m$, код које су врсте и колоне замениле места у односу на изворну матрицу, тј.:

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}, \quad A^T = \begin{bmatrix} a_{11} & a_{21} & \dots & a_{m1} \\ a_{12} & a_{22} & \dots & a_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & \dots & a_{mn} \end{bmatrix}.$$

Пример 6.8. Демонстрирајмо ову формулу за неусмерени граф на сл. 6.3. Дијагонална матрица D за овај граф је:

$$D = \begin{bmatrix} \frac{1}{3} & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{6.8}$$

док је матрица повезаности A_G већ дата на сл. 6.6. Сада можемо да одредимо и матрицу прелаза M:

$$M = D \times A_G = \begin{bmatrix} \frac{1}{3} & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}, \tag{6.9}$$

тј.,

$$M^T = \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 1 \\ \frac{1}{3} & 0 & \frac{1}{2} & 0 \\ \frac{1}{3} & \frac{1}{2} & 0 & 0 \\ \frac{1}{3} & 0 & 0 & 0 \end{bmatrix}. \tag{6.10}$$

Претпоставимо да је почетна расподела вероватноћа дата матрицом:

$$P_0^I = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{6.11}$$

За прва два корака обиласка графа на случајни начин, расподеле вероватноћа по чворовима су:

$$P_1 = M^T \times P_0 = \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 1 \\ \frac{1}{3} & 0 & \frac{1}{2} & 0 \\ \frac{1}{3} & \frac{1}{2} & 0 & 0 \\ \frac{1}{3} & 0 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{3} \end{bmatrix} ,$$

$$P_2 = (M^T)^2 \times P_0 = M^T \times P_1 = \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 1 \\ \frac{1}{3} & 0 & \frac{1}{2} & 0 \\ \frac{1}{3} & \frac{1}{2} & 0 & 0 \\ \frac{1}{3} & 0 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} 0 \\ \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{3} \end{bmatrix} = \begin{bmatrix} \frac{2}{3} \\ \frac{1}{6} \\ \frac{1}{6} \\ 0 \end{bmatrix} .$$

Претпоставимо да смо почетну расподелу вероватноћа P_0 изабрали тако да важи $P_0 = M^T \times P_0$. Тада, из израза (6.7) следи да важи $P_t = P_0$, за $t > 0$. Другим речима, вероватноћа да ћемо се наћи на чвору v је иста у сваком кораку обиласка графа:

$$(\forall\, t > 0)\, P_t(v) = P_0(v) .$$

Овакав обилазак називамо *стационарним*, а почетну расподелу вероватноћа P_0 — стационарном. За неусмерени граф $G = (V, E)$, стационарна расподела вероватноћа се рачуна на следећи начин:

$$P = \begin{matrix} v_1 \\ v_2 \\ \cdots \\ v_i \\ \cdots \\ v_n \end{matrix} \begin{bmatrix} \frac{d(v_1)}{2|E|} \\ \frac{d(v_2)}{2|E|} \\ \cdots \\ \frac{d(v_i)}{2|E|} \\ \cdots \\ \frac{d(v_n)}{2|E|} \end{bmatrix} , \tag{6.12}$$

где су: $d(v_i)$ — степен чвора v_i, $|E|$ — број грана у графу.

Пример 6.9. За неусмерени граф дат на сл. 6.3, стационарна расподела вероватноћа је:

$$P_s = \begin{bmatrix} \frac{3}{8} \\ \frac{2}{8} \\ \frac{2}{8} \\ \frac{1}{8} \end{bmatrix} . \tag{6.13}$$

Матрицу M за овај граф смо већ израчунали (в. израз (6.9)), па провером можемо утврдити да за ову расподелу важи:

$$P_s = M^T \times P_s ,$$

тј.,

$$P_t = P_s, \text{ за } t \geq 0 .$$

Стационарна расподела вероватноћа се може и овако интерпретирати: за обилазак повезаног и апериодичног[1] графа $G = (V, E)$ на случајни начин, вероватноћа да ћемо се наћи на датом чвору v временом конвергира ка вредности:

$$p_v = \frac{d(v)}{2|E|} . \tag{6.14}$$

Ова вероватноћа је сразмерна степену чвора v, $p_v \sim d(v)$, што значи да је обилазак графа на случајни начин *пристрасан* (енгл. *biased*). Нпр., ако је степен чвора v_i два пута већи од степена чвора v_j ($d(v_i) = 2d(v_j)$), то значи да ће, приликом обиласка графа на случајни начин, број посета чвору v_i бити два пута већи од броја посета чвору v_j. Из особине пристрасности произлази следећа идеја:

- чворови графа се могу рангирати према вероватноћама стационарне расподеле вероватноћа за дати граф,
- ранг чвора је сразмеран броју грана са којима је чвор повезан.

Критеријум за рангирање веб-страница који се заснива на овој идеји је размотрен у следећој секцији.

6.3 Рангирање веб-страница

У контексту веб-претраге, релевантност веб-странице (тј., чвора који представља пронађени документ) би се могла дефинисати као сразмерна броју хипервеза које на њу упућују (тј., броју улазних грана). Замислимо хипотетичког корисника који обилази веб на случајни начин. Пошто је овакав обилазак пристрасан, корисник ће чешће посећивати странице на које упућује већи број хипервеза. Међутим, критеријум који овде разматрамо проширује ову концептуализацију [8, 3]. Ранг веб-странице v не зависи само од броја хипервеза које на њу упућују, већ и од њиховог „квалитета“, при чему се сматра да је „квалитет“ хипервезе:

- сразмеран рангу странице са које потиче,
- обрнуто сразмеран броју хипервеза на страници са које потиче.

Ранг странице v се формално може дефинисати на следећи начин:

$$R(v) = \sum_{v_i \in L_v} \frac{R(v_i)}{n_i} , \tag{6.15}$$

где су:

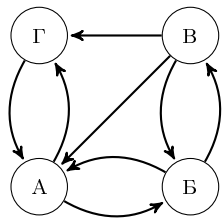
- v — веб-страница,

[1] Граф је апериодичан ако је највећи заједнички делилац дужина свих циклуса у графу једнак 1.

- $R(v)$ — ранг веб-странице v,
- $L_v = \{v_1, v_2, ..., v_n\}$ — скуп страница које имају хипервезу ка страници v,
- n_i — број хипервеза које садржи страница $v_i \in L_v$,

тј., ранг странице је једнак суми нормализованих рангова страница које је референцирају, при чему се нормализовани ранг странице добија када се ранг странице подели бројем хипервеза које страница садржи.

Пример 6.10. Посматрајмо усмерени граф који представља подскуп веб-страница и веза између њих, представљен на сл. 6.8. Применом израза



Слика 6.8 Усмерени граф који представља подскуп веб-страница и веза између њих.

(6.15), за овај граф се може дефинисати следећи систем једначина:

$$\begin{cases} R(\text{А}) = \frac{R(\text{Б})}{2} + \frac{R(\text{В})}{3} + \frac{R(\Gamma)}{1} \\ R(\text{Б}) = \frac{R(\text{А})}{2} + \frac{R(\text{В})}{3} \\ R(\text{В}) = \frac{R(\text{Б})}{2} \\ R(\Gamma) = \frac{R(\text{А})}{2} + \frac{R(\text{В})}{3} \end{cases} . \tag{6.16}$$

Овај систем једначина се може решити аналитички. Међутим, пошто постоји више решења, изабраћемо оно решење које задовољава и додатни услов: $R(\text{А}) + R(\text{Б}) + R(\text{В}) + R(\Gamma) = 1$. Ово решење се назива *сопствени вектор* (нем. *Eigenvektor*), и у овом примеру је:

$$\begin{bmatrix} R(\text{А}) \\ R(\text{Б}) \\ R(\text{В}) \\ R(\Gamma) \end{bmatrix} = \begin{bmatrix} \frac{10}{25} \\ \frac{6}{25} \\ \frac{3}{25} \\ \frac{6}{25} \end{bmatrix} = \begin{bmatrix} 0,4 \\ 0,24 \\ 0,12 \\ 0,24 \end{bmatrix} . \tag{6.17}$$

Систем једначина (6.16) се може представити и матрично:

$$\begin{bmatrix} R(\text{А}) \\ R(\text{Б}) \\ R(\text{В}) \\ R(\Gamma) \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{3} & 1 \\ \frac{1}{2} & 0 & \frac{1}{3} & 0 \\ 0 & \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{3} & 0 \end{bmatrix}}_{M_t} \times \begin{bmatrix} R(\text{А}) \\ R(\text{Б}) \\ R(\text{В}) \\ R(\Gamma) \end{bmatrix} , \tag{6.18}$$

где је M_t тежинска матрица прелаза. Уколико је граф који представља веб повезан, тада се, за произвољну матрицу почетне расподеле вероватноћа P_0, сопствени вектор P добија формулом [4, 5]:

$$P = \lim_{k\to\infty} M_t^k \times P_0 \,, \tag{6.19}$$

тј., низ матрица $P_0, M_t \times P_0, M_t^2 \times P_0, \ldots$ конвергира ка сопственом вектору. Практично, израчунавање сопственог вектора се своди на следеће кораке:

- изабере се почетна матрица расподеле вероватноћа P_0,
- потом се израчунавају елементи низа $M_t \times P_0, M_t^2 \times P_0, \ldots$, све док апсолутна разлика између вредности два узастопна елемента низа није мања од унапред задате вредности Δ.

Уколико је вредност Δ адекватно изабрана, елемент низа који је последњи израчунат је приближно једнак сопственом вектору.

Приметите да i-ти елемент низа $M_t \times P_0, M_t^2 \times P_0, \ldots$ представља расподелу вероватноћа по чворовима у i-том кораку обиласка графа на случајни начин.

Пример 6.11. Решимо систем једначина (6.16) итеративним приступом. Као почетну расподелу вероватноћа изабраћемо ону која свим чворовима графа додељује једнаке вероватноће $(\frac{1}{4})$ да буду почетни чворови, тј:

$$P_0 = \begin{bmatrix} \frac{1}{4} \\ \frac{1}{4} \\ \frac{1}{4} \\ \frac{1}{4} \end{bmatrix} \,. \tag{6.20}$$

Табела 6.1 садржи вредности елемената изабраних матрица које представљају расподеле вероватноћа на старту, после првог прелаза, после другог прелаза, итд. За $\Delta = 0,001$, приближна вредност сопственог вектора се добија у 49. итерацији. За $\Delta = 0,0001$, приближна вредност сопственог вектора се добија у 73. итерацији. Може се приметити да вредности конвергирају ка тачном решењу система, датом изразом 6.17.

У практичним применама, величине графова који се обрађују на овај начин су знатне. Због тога се рангови страница израчунавају унапред, а не у реалном времену. А кад корисник зада упит, релевантност пронађених страница се процењује у реалном времену, узимајући у обзир претходно израчунате рангове, али и додатне факторе који могу да утичу на релевантност резултата (попут информација о упиту).

Табела 6.1 Илустрација итеративне методе за израчунавање сопственог вектора.

	P_i	P_i(А)	P_i(Б)	P_i(В)	P_i(Г)	
старт,	P_0	0,25	0,25	0,25	0,25	
итерација 1.,	$P_1 = M_t \times P_0$	0,458333	0,208333	0,125	0,208333	
итерација 2.,	$P_2 = M_t^2 \times P_0$	0,354167	0,270833	0,104167	0,270833	
		⋮				
итерација 49.,	$P_{49} = M_t^{49} \times P_0$	0,400489	0,239663	0,120186	0,239663	$\leftarrow \Delta = 0{,}001$
		⋮				
итерација 73.,	$P_{73} = M_t^{73} \times P_0$	0,400049	0,239967	0,120019	0,239967	$\leftarrow \Delta = 0{,}0001$
тачно решење:			0,4	0,24	0,12	0,24

Рангирање веб-страница представљено у овој секцији зависи само од структуре графа који представља подскуп веб-страница и веза између њих. Ради се о важној концептуализацији релевантности веб-странице, на којој се делом заснива и један од најпознатијих алгоритама за рангирање страница — ПејџРенк (енгл. PageRank [8, 3]). Овај алгоритам је био веома важна компонента система Гугл у првим годинама његовог постојања, а тренутно је само једна од преко 200 променљивих које систем узима у обзир приликом рангирања страница, укључујући: језичка обележја упита (нпр., фразе, синониме, словне грешке), временска обележја упита (нпр., информације о времену индексирања докумената садржаних у резултату обраде упита), и информације о кориснику (нпр., информације о претходно задатим упитима, итд.) [4].

6.4 Метрополис-Хејстингсов алгоритам

Анализа садржаја на вебу често укључује примену алгоритама за обилазак графа. Због велике количине доступних података на вебу, потребно је изабрати подскуп веб-страница на коме се може спровести анализа. Алгоритми за обилазак графа се примењују у поступку изабирања страница за анализу. Притом, да би резултати анализе могли да се генерализују, потребно је да подскуп веб-страница на коме је извршена анализа буде репрезентативан[2].

[2] У поглављу 4 (секција 4.1.1) смо истакли да се појам репрезентативности језичког корпуса односи на обим у којем корпус укључује целокупни опсег разноврсности у оквиру датог језика [1]. Слично важи и за репрезентативност подскупа веб-страница.

У претходном тексту смо видели да је обилазак графа на случајни начин пристрасан, и да се на овој особини заснива један од критеријума за рангирање веб-страница. Са друге стране, основни алгоритам за обилазак графа на случајни начин није, у општем случају, адекватан за селектовање подскупа веб-страница, јер пристрасност обиласка доводи до тога да узорак страница није репрезентативан — тј., веб-странице које референцира већи број хипервеза ће бити фаворизоване приликом избора.

Метрополис-Хејстингсов алгоритам за обилазак графа на случајни начин (енгл. *Metropolis-Hastings Random Walk*, [6]) коригује пристрасност према чворовима са већим степенима. Овај алгоритам ћемо илустровати на неусмереном графу. Претпоставимо да се налазимо на чвору v_i, и да је v_j један од његових суседних чворова. У основном алгоритму за обилазак графа на случајни начин, вероватноћа преласка са чвора v_i на чвор v_j зависи само од степена чвора v_i, тј:

$$p_{v_i,v_j} = \frac{1}{d(v_i)}\,, \tag{6.21}$$

где је $d(v_i)$ степен чвора v_i. У Метрополис-Хејстингсовом алгоритму, вероватноћа прелаза са чвора v_i на чвор v_j зависи од степена оба чвора, и дефинише се на следећи начин:

$$p_{v_i,v_j} = \begin{cases} \frac{1}{d(v_i)},\ d(v_i) \geq d(v_j) \\ \frac{1}{d(v_j)},\ d(v_i) < d(v_j) \end{cases} = \frac{1}{d(v_i)} \cdot \min\{1, \frac{d(v_i)}{d(v_j)}\}\,. \tag{6.22}$$

Ако је степен чвора v_j већи од степена чвора v_i, вероватноћа преласка са чвора v_i на чвор v_j је мања него код основног алгоритма за обилазак графа на случајни начин, чиме се коригује пристрасност обиласка.

У општем случају, вероватноћа прелаза са чвора v_i на чвор v_j, при чему v_i и v_j нису обавезно суседни чворови, се дефинише:

$$p_{v_i,v_j} = \begin{cases} \frac{1}{d(v_i)} \cdot \min\{1, \frac{d(v_i)}{d(v_j)}\}, & \text{ако су } v_i \text{ и } v_j \text{ суседни чворови,} \\ 1 - \sum\limits_{v \neq v_i} p_{v_i,v}, & \text{ако је } v_i = v_j, \\ 0, & \text{у супротном.} \end{cases} \tag{6.23}$$

Из израза (6.22) такође следи $p_{v_i,v_j} = p_{v_j,v_i}$, па је матрица прелаза која се генерише на основу израза (6.23) симетрична, тј., $M_{MH} = M_{MH}^T$.

Пример 6.12. За неусмерени граф приказан на сл. 6.6, матрица прелаза која се добија након Метрополис-Хејстингсове корекције пристрасности је:

Она се односи на обим заступљености и узајамне балансираности феномена који представљају предмет анализе.

$$M_{MH} = \begin{array}{c} \\ \text{А} \\ \text{Б} \\ \text{В} \\ \text{Г} \end{array} \begin{array}{c} \begin{array}{cccc} \text{А} & \text{Б} & \text{В} & \text{Г} \end{array} \\ \begin{bmatrix} 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{6} & \frac{1}{2} & 0 \\ \frac{1}{3} & \frac{1}{2} & \frac{1}{6} & 0 \\ \frac{1}{3} & 0 & 0 & \frac{2}{3} \end{bmatrix} \end{array} . \tag{6.24}$$

Размотримо израчунавање ове матрице за врсту која одговара чвору В. Вероватноће $p_{\text{В,А}}$, $p_{\text{В,Б}}$ и $p_{\text{В,Г}}$ се могу израчунати директно:

$$p_{\text{В,А}} = \min\{\frac{1}{d(B)}, \frac{1}{d(\text{А})}\} = \min\{\frac{1}{2}, \frac{1}{3}\} = \frac{1}{3} ,$$
$$p_{\text{В,Б}} = \min\{\frac{1}{d(B)}, \frac{1}{d(\text{Б})}\} = \min\{\frac{1}{2}, \frac{1}{2}\} = \frac{1}{2} ,$$
$$p_{\text{В,Г}} = 0, \text{ јер чворови В и Г нису суседни.}$$

Вероватноћа $p_{\text{В,В}}$ се рачуна посредно:

$$p_{\text{В,В}} = 1 - (p_{\text{В,А}} + p_{\text{В,Б}} + p_{\text{В,Г}}) = \frac{1}{6} .$$

Корекција пристрасности Метрополис-Хејстингсовим алгоритмом се може илустровати на следећи начин. За произвољну почетну расподелу вероватноћа P_0, низ матрица $M_{MH} \times P_0, M_{MH}^2 \times P_0, \ldots$ конвергира ка униформној расподели вероватноћа:

$$\begin{bmatrix} \frac{1}{n} \\ \frac{1}{n} \\ \vdots \\ \frac{1}{n} \end{bmatrix} , \tag{6.25}$$

где је n број чворова у графу.

Пример 6.13. Као почетну расподелу вероватноћа за неусмерени граф приказан на сл. 6.6 изаберимо ону која одређује А као почетни чвор, тј:

$$P_0 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} .$$

Слично као у примеру 6.11, да бисмо приближно одредили матрицу којој конвергира низ $M_{MH} \times P_0, M_{MH}^2 \times P_0, \ldots$, можемо да израчунавамо елементе низа, све док апсолутна разлика између вредности два узастопна елемента низа није мања од унапред задате вредности Δ. Табела 6.2 садржи вредности елемената матрица које представљају расподеле вероватноћа на старту, после првог прелаза, после другог прелаза, итд. За $\Delta = 0,001$,

приближна вредност униформне расподеле се добија у 7. итерацији. За $\Delta = 0,0001$, приближна вредност униформне расподеле се добија у 9. итерацији.

Табела 6.2 Илустрација Метрополис-Хејстингсовог алгоритма.

	P_i	P_i(А)	P_i(Б)	P_i(В)	P_i(Г)	
старт,	P_0	1	0	0	0	
итерација 1.,	$P_1 = M_{MH} \times P_0$	0	0,333333	0,333333	0,333333	
итерација 2.,	$P_2 = M_{MH}^2 \times P_0$	0,333333	0,222222	0,222222	0,222222	
итерација 3.,	$P_3 = M_{MH}^3 \times P_0$	0,222222	0,259259	0,259259	0,259259	
итерација 4.,	$P_4 = M_{MH}^4 \times P_0$	0,259259	0,246914	0,246914	0,246914	
итерација 5.,	$P_5 = M_{MH}^5 \times P_0$	0,246914	0,251029	0,251029	0,251026	
итерација 6.,	$P_6 = M_{MH}^6 \times P_0$	0,251029	0,249657	0,249657	0,249657	
итерација 7.,	$P_7 = M_{MH}^7 \times P_0$	0,249657	0,250114	0,250114	0,250114	$\leftarrow \Delta = 0,001$
итерација 8.,	$P_8 = M_{MH}^8 \times P_0$	0,250114	0,249962	0,249962	0,249962	
итерација 9.,	$P_9 = M_{MH}^9 \times P_0$	0,249962	0,250013	0,250013	0,250013	$\leftarrow \Delta = 0,0001$
униформна расподела:		0,25	0,25	0,25	0,25	

6.5 Задаци

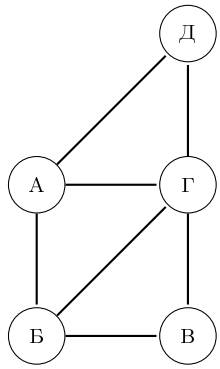
Задатак 6.1. За неусмерени граф приказан на сл. 6.9, израчунајте матрицу повезаности, дијагоналну матрицу, матрицу прелаза и стационарну расподелу вероватноћа за обилазак графа на случајни начин. За дату почетну расподелу вероватноћа:

$$P_0 = \begin{matrix} \text{А} \\ \text{Б} \\ \text{В} \\ \text{Г} \\ \text{Д} \end{matrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix},$$

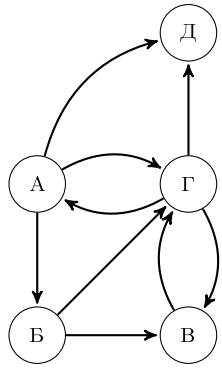
израчунајте расподеле вероватноћа након првог, односно другог прелаза.

Задатак 6.2. Да ли је могуће директно одредити матрицу прелаза дату у примеру 6.8 (тј., без множења дијагоналне матрице и матрице повезаности)? Објасните.

Задатак 6.3. За усмерени граф приказан на сл. 6.10, поставите систем једначина користећи израз (6.15), и одредите аналитички сопствени вектор који задовољава овај систем.



Слика 6.9 Слика уз задатке 6.1 и 6.5.



Слика 6.10 Слика уз задатке 6.3 и 6.4.

Задатак 6.4. За усмерени граф приказан на сл. 6.10, одредите тежинску матрицу прелаза. Као почетну расподелу вероватноћа изаберите ону која свим чворовима графа додељује једнаке вероватноће, и израчунајте расподеле вероватноћа након првог, односно другог прелаза.

Задатак 6.5. За неусмерени граф приказан на сл. 6.9, израчунајте матрицу прелаза користећи Матрополис-Хејстингсов алгоритам. Као почетну расподелу вероватноћа изаберите ону која чворовима А и Г додељује вероватноће 0,5, и израчунајте расподеле вероватноћа након првог, односно другог прелаза.

Задатак 6.6. Нека су чворови v_j и v_k суседи чвора v_i, и нека је степен чвора v_j већи од степена чвора v_k, тј. $d(v_j) > d(v_k)$. Полазећи од израза (6.22), покажите да у Метрополис-Хејстингсовом алгоритму важи $p_{v_i,v_j} \leq p_{v_i,v_k}$. Од чега зависи да ли важи $p_{v_i,v_j} = p_{v_i,v_k}$ или $p_{v_i,v_j} < p_{v_i,v_k}$?

Литература

1. Biber D (1993) Representativeness in Corpus Design. Literary and Linguistic Computing 8(4): 243–257.
2. Bondy A, Murty USR (2008) Graph Theory. Springer-Verlag London.
3. Brin S, Page L (1998) The anatomy of a large-scale hypertextual web search engine. Computer Networks and ISDN Systems 30(1–7), pp. 107–117.
4. Ceri S, Bozzon A, Brambilla M, Della Valle E, Fraternali P, Quarteroni S (2013) Web Information Retrieval, Springer-Verlag Berlin Heidelberg.
5. Langville A, Meyer C (2008) Google's Page Rank and Beyond: the Science of Search Engine Rankings, Princeton University Press, Princeton.
6. Lovász L (1993) Random Walks on Graphs: A Survey, Combinatorics, Paul Erdős is Eighty (Volume 2), Keszthely (Hungary), pp. 1–46.
7. Manning CD, Raghavan P, Schütze H (2008) Introduction to Information Retrieval, Cambridge University Press.
8. Page L, Brin S, Motwani R, Winograd T (1999) The PageRank citation ranking: bringing order to the web, Technical report, Stanford InfoLab.

Поглавље 7
Класификација текста

Класификација текста је важни аспект проналажења информација. Претпоставимо да хипотетички корисник жели да прати развој у области дијалошких система. Да би пронашао релевантне текстове на вебу, овај корисник би, у терминима буловске претраге, могао да зада следећи упит:

$$\text{дијалошки} \wedge \text{систем},$$

којим налаже систему да пронађе сва документа која садрже термине „дијалошки“ и „систем“. Међутим, пошто се за дијалошке системе често користи израз „конверзациони агенти“, овај упит се може допунити, нпр.:

$$(\text{дијалошки} \vee \text{конверзациони}) \wedge (\text{систем} \vee \text{агент}).$$

Да би се постигао већи одзив, овај упит се може додатно проширивати, тако да обухвата и друге термине релевантне за тему претраге, попут „системи за обраду природних језика“, „језички интерфејси“, итд. Сваком допуном упита, његова комплексност расте. Уместо генерисања комплексних упита, адекватније би било кад би корисник могао да зада упит систему да пронађе документа која тематски припадају области развоја дијалошких система. Оваква функционалност система је пример класификације текста. Међутим, класификација текста није ограничена само на аутоматску детекцију теме текста. У општем случају, класификацију текста можемо посматрати на следећи начин — за коначни скуп класа $\{c_1, c_2, ..., c_n\}$, потребно је одредити којој класи (или којим класама) припада дати документ d [3]. Разне примене класификације текста укључују:

- аутоматску детекцију нежељених електронских порука,
- аутоматску детекцију неприкладних садржаја,
- аутоматску детекцију емоционалне обојености текста,
- аутоматско идентификовање аутора текста, итд.

У овом поглављу ћемо размотрити основе два приступа класификацији текста. Први приступ се заснива на Бајесовој теореми (в. секцију 7.1), и његова примена ће бити илустрована за аутоматску детекцију нежељених електронских порука. Други приступ се заснива на биграмима на

нивоу знакова (в. секцију 7.2), и његова примена ће бити илустрована за аутоматско идентификовање аутора текста.

7.1 Бајесовска класификација

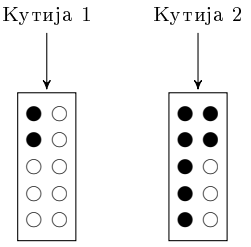
У поглављу 5 смо већ користили Бајесову теорему [1] за контекстно зависно исправљање грешака у корисничким упитима. Овде ћемо размотрити њену примену у класификацији текста. Због комплетности излагања, навешћемо опет ову теорему. Нека су A и B случајни догађаји, при чему важи $P(B) \neq 0$. Тада важи:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}\,, \tag{7.1}$$

где су:

- $P(A)$ — вероватноћа реализације догађаја A,
- $P(B)$ — вероватноћа реализације догађаја B,
- $P(A|B)$ — вероватноћа реализације догађаја A, под условом да се реализовао догађај B,
- $P(B|A)$ — вероватноћа реализације догађаја B, под условом да се реализовао догађај A.

Пример 7.1. Претпоставимо да имамо две кутије — у првој кутији (K_1) се налази 8 белих куглица и 2 црне куглице, а у другој кутији (K_2) се налази 7 црних куглица и 3 беле куглице (в. сл. 7.1). На случајни начин бирамо једну од кутија, а потом из ње на случајни начин бирамо куглицу. Ако је извучена црна куглица (●), колика је вероватноћа да смо изабрали прву кутију, $P(K_1|●)$? Ова вероватноћа се не може израчунати директно, већ



Слика 7.1 Илустрација Бајесове теореме.

посредно, применом Бајесове теореме:

$$P(K_1|●) = \frac{P(●|K_1)P(K_1)}{P(●)}\,. \tag{7.2}$$

Вероватноће које се јављају са десне стране знака једнакости у изразу (7.2) можемо да израчунамо директно. Из поставке примера, познате су нам следеће вероватноће:

- вероватноћа да је изабрана прва кутија $P(K_1) = \frac{1}{2}$,
- вероватноћа да је изабрана друга кутија $P(K_2) = \frac{1}{2}$,
- вероватноћа да се из прве кутије на случајни начин извуче црна куглица $P(\bullet|K_1) = \frac{2}{10}$,
- вероватноћа да се из друге кутије на случајни начин извуче црна куглица $P(\bullet|K_2) = \frac{7}{10}$.

Недостаје нам вероватноћа да је извучена црна куглица, $P(\bullet)$. Ова вероватноћа је једнака збиру вероватноћа следећих догађаја:

- изабрали смо прву кутију, па из ње извукли црну куглицу,
- изабрали смо другу кутију, па из ње извукли црну куглицу,

тј.:

$$\begin{aligned} P(\bullet) &= P(K_1)P(\bullet|K_1) + P(K_2)P(\bullet|K_2) \\ &= \frac{1}{2} \cdot \frac{2}{10} + \frac{1}{2} \cdot \frac{7}{10} = \frac{9}{20}\,. \end{aligned}$$

Сад можемо да израчунамо тражену вероватноћу:

$$\begin{aligned} P(K_1|\bullet) &= \frac{P(\bullet|K_1)P(K_1)}{P(\bullet)} \\ &= \frac{P(\bullet|K_1)P(K_1)}{P(K_1)P(\bullet|K_1) + P(K_2)P(\bullet|K_2)} \\ &= \frac{\frac{2}{10} \cdot \frac{1}{2}}{\frac{9}{20}} \\ &= \frac{2}{9}\,. \end{aligned}$$

Пример 7.2. Претпоставимо да располажемо корпусом који садржи 1000 електронских порука, на основу ког желимо да обучимо софтвер да детектује нежељене поруке (тзв. *спем*) на основу термина садржаних у поруци. Анализом корпуса смо утврдили да 300 порука спада у групу нежељених, док су остале поруке оцењене као коректне. Такође, утврдили смо да се термин $t =$ „награда“ налази у 25% нежељених порука, и у 5% коректних порука. Ако непозната електронска порука (тј., порука за коју се не зна унапред да ли је коректна или не) садржи термин „награда“, како можемо да проценимо да ли је посматрана порука коректна или нежељена?

Из поставке примера, познате су нам следеће вероватноће:

- вероватноћа да је електронска порука нежељена је $P(S) = \frac{300}{1000} = 0,3$,

- вероватноћа да је електронска порука коректна је $P(C) = \frac{1000-300}{1000} = 0,7$,
- вероватноћа да нежељена порука садржи термин $t =$ „награда“ је $P(t|S) = 0,25$,
- вероватноћа да коректна порука садржи термин $t =$ „награда“ је $P(t|C) = 0,05$.

Вероватноћа да је порука која садржи термин $t =$ „награда“ нежељена, $P(S|t)$, се може израчунати применом Бајесове теореме, слично као у претходном примеру:

$$\begin{aligned} P(S|t) &= \frac{P(t|S)P(S)}{P(t)} \\ &= \frac{P(t|S)P(S)}{P(t|S)P(S)+P(t|C)P(C)} \\ &\approx 0,68\,. \end{aligned}$$

Вероватноћа да је посматрана порука коректна, $P(C|t)$ се такође може израчунати применом Бајесове теореме. Међутим, пошто у овом примеру имамо само две категорије (нежељена порука и коректна порука), вероватноћу $P(C|t)$ можемо да израчунамо и на следећи начин:

$$P(C|t) = 1 - P(S|t) \approx 0,32\,.$$

Добили смо да важи $P(S|t) > P(C|t)$, тј., вероватније је да је посматрана порука нежељена.

У општем случају бајесовске класификације [3], можемо сматрати да су нам познати:

- скуп класа $C = \{c_1, c_2, ..., c_n\}$, који се одређује у зависности од конкретног проблема који се решава,
- колекција докумената D, при чему је за сваки документ $d \in D$ назначена класа $c \in C$ којој припада.

Основни задатак класификације докумената је да се на основу информација о терминима садржаним у документу одреди класа којој документ припада. Посматрајмо најједноставнији случај — ако је познато да документ d садржи термин t, одредимо којој класи документ највероватније припада. Слично као у примеру 7.2, за све расположиве класе $c \in C$ се израчунају вероватноће $P(c|t)$, и класа $\hat{c}$ за коју се добије максимална вероватноћа се бира као највероватније решење, тј.:

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}}\, P(c|t)\,. \tag{7.3}$$

За израчунавање вероватноће $P(c|t)$ користимо Бајесову теорему:

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}}\, \frac{P(t|c)P(c)}{P(t)}\,. \tag{7.4}$$

Анализирајмо вероватноће у изразу (7.4):

- $P(c)$ означава вероватноћу класе c. Ова вероватноћа се може проценити на следећи начин:
$$P(c) = \frac{N_c}{N}\,, \tag{7.5}$$
где су: N_c — број докумената у колекцији који припадају класи c, N — укупни број докумената у колекцији.
- $P(t|c)$ означава вероватноћу да документ који припада класи c садржи термин t. Ова вероватноћа се може концептуално представити као релативна учесталост термина t у документима који припадају класи c, тј.:
$$P(t|c) = \frac{T_{ct}}{\sum\limits_{t' \in V} T_{ct'}}\,, \tag{7.6}$$
где су: T_{ct} — број јављања термина t у свим документима класе c, V — речник индексних термина. Формула (7.6) представља процену максималне изгледности, коју смо већ разматрали у поглављу 4. Треба приметити да је ова формула такође подложна Лапласовој корекцији (разлози су већ изнети у поглављу 4). Коригована формула гласи:
$$P(t|c) = \frac{T_{ct}+1}{(\sum\limits_{t' \in V} T_{ct'}) + |V|}\,. \tag{7.7}$$
- $P(t)$ означава вероватноћу јављања термина t, и њу не морамо да рачунамо. Вероватноћа $P(t)$ је константна, па се проблем одређивања класе која максимизује вредност $\frac{P(t|c)P(c)}{P(t)}$ (в. израз (7.4)) може свести на одређивање класе за коју је вредност $P(t|c)P(c)$ максимална.

Другим речима, формулу (7.4) можемо да упростимо:

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}}\, [P(t|c)P(c)]\,. \tag{7.8}$$

До сада смо разматрали случај у ком се класификација документа врши на основу информације о присутности једног термина. У општем случају бајесовске класификације, документ d садржи скуп термина $\{t_1, t_2, ..., t_n\}$. Формула (7.8) се тада уопштава на следећи начин:

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}}\, [P(c) \cdot \prod_{1 \le k \le n} P(t_k|c)]\,. \tag{7.9}$$

За реалне језичке корпусе, вредности вероватноћа које разматрамо су толико мале да се не могу правилно рачунарски представити (в. поглавље 4). Да би се избегло аритметичко поткорачење, уместо ве-

роватноће P се користи њена логаритмована вредност $\log P$, што операције множења вероватноћа у горњим формулама замењује операцијама сабирања логаритмованих вероватноћа. Формула (7.9) се тиме преводи у следећи облик [3]:

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} \left[\log P(c) + \sum_{1 \leq k \leq n} \log P(t_k|c) \right] . \tag{7.10}$$

Пример 7.3. Табела 7.1 приказује минималистичке[1] корпусе за обучавање и тестирање класификатора електронских порука. Корпус за обучавање садржи скупове индексних термина који описују четири електронске поруке (d_1, d_2, d_3, d_4) за које је назначено да ли припадају класи коректних (C) или нежељених (S) порука. Корпус за тестирање класификатора садржи скуп индексних термина који описују поруку d_5 за коју треба одредити класу којој припада. Класу поруке d_5 ћемо одредити применом Бајесове теореме и Лапласове корекције процене максималне изгледности.

Табела 7.1 Корпуси за обучавање и тестирање класификатора електронских порука.

	шифра документа	индексни термини	класа
корпус за обучавање	d_1	награда милион долар награда	S
	d_2	писац награда	C
	d_3	награда корисник награда	S
	d_4	милион корисник	S
корпус за тестирање	d_5	писац корисник награда	?

Речник генерисан из корпуса за обучавање класификатора садржи пет индексних термина:

$$V = \{\text{„долар“}, \text{„корисник“}, \text{„милион“}, \text{„награда“}, \text{„писац“}\} .$$

У корпусу за обучавање класификатора, три од четири поруке су означене као нежељене, па применом формуле (7.5) добијамо вероватноћу да је порука нежељена, $P(S) = \frac{3}{4}$. Слично се добија вероватноћа да је порука коректна, $P(C) = \frac{1}{4}$.

За индексне термине који описују поруку d_5 рачунамо условне вероватноће користећи формулу (7.7):

[1] Корпуси у овом примеру су намерно нереално мали, да би поступак рачунања био лакши.

$$P(\text{„писац“}|S) = \frac{0+1}{9+5} = \frac{1}{14}\,,$$
$$P(\text{„писац“}|C) = \frac{1+1}{2+5} = \frac{2}{7}\,,$$
$$P(\text{„корисник“}|S) = \frac{2+1}{9+5} = \frac{3}{14}\,,$$
$$P(\text{„корисник“}|C) = \frac{0+1}{2+5} = \frac{1}{7}\,,$$
$$P(\text{„награда“}|S) = \frac{4+1}{9+5} = \frac{5}{14}\,,$$
$$P(\text{„награда“}|C) = \frac{1+1}{2+5} = \frac{2}{7}\,.$$

Сада можемо да израчунамо вероватноће које се јављају у формули (7.9):

$$\begin{aligned} P_S &= P(S)\cdot P(\text{„писац“}|S)\cdot P(\text{„корисник“}|S)\cdot P(\text{„награда“}|S) \qquad (7.11)\\ &= \frac{3}{4}\cdot\frac{1}{14}\cdot\frac{3}{14}\cdot\frac{5}{14}\\ &\approx 0,004\,,\\ P_C &= P(C)\cdot P(\text{„писац“}|C)\cdot P(\text{„корисник“}|C)\cdot P(\text{„награда“}|C) \qquad (7.12)\\ &= \frac{1}{4}\cdot\frac{2}{7}\cdot\frac{1}{7}\cdot\frac{2}{7}\\ &\approx 0,003\,. \end{aligned}$$

Добили смо да важи $P_S > P_C$, па је, по формули (7.9), порука d_5 класификована као нежељена.

7.2 Класификација на основу биграма знакова

Други приступ класификацији текста који разматрамо у овом поглављу се заснива на биграмима на нивоу знакова. Његову примену ћемо илустровати за аутоматско идентификовање аутора текста. Нека је дат скуп аутора познатог идентитета, и нека је за сваког од њих доступан скуп текстова које су написали. Нека је T текст непознатог аутора. Под идентификовањем аутора подразумевамо да се из датог скупа аутора познатог идентитета одреди највероватнији аутор текста T.

Прво аутоматско идентификовање аутора текста је примењено на „Федералистичким списима“, колекцији 85 новинских чланака које су под истим псеудонимом објавили Александер Хамилтон, Џејмс Ме-

дисон и Џон Џеј у периоду 1787–1788. За неке од ових текстова није било познато да ли су их написали Хамилтон или Медисон [4]. Тренутно се аутоматско идентификовање аутора практично примењује за идентификовање аутора електронских порука, детекцију плагијаризма, форензичка испитивања, итд.

Алгоритми за идентификовање аутора текста се заснивају на чињеници да текст рефлектује ауторов стил писања на различитим нивоима: семантичком, синтаксном, лексикографском, ортографском, морфолошком [5]. Аутор углавном несвесно примењује свој стил писања, тако да обележја стила представљају корисну основу за утврђивање ауторства. Уобичајени приступи анализи стила писања укључују два корака: издвајање обележја стила, и класификацију. Издвајање обележја стила укључује језичку анализу (означавање делова текста, парсирање, морфолошка анализа, итд.), а класификација подразумева избор обележја стила (нпр., применом статистичких метода) која су релеватна за идентификовање аутора. Овакви приступи имају следеће недостатке:

- Технике за издвајање обележја стила писања у великој мери зависе од језика — нпр., парсер који се користи за енглески језик се не може употребити за српски језик.
- Избор релевантних обележја стила није тривијални задатак — нпр., погрешно је априорно закључити да обележја која се ретко јављају нису важна за идентификовање аутора, јер могу да имају значајни кумулативни ефекат.
- Анализа стила се често врши на нивоу речи, чиме се запостављају морфолошка обележја стила. Поред тога, проблем сегментације речи у неким језицима, попут кинеског и јапанског, није тривијалан.

Ови недостаци се могу превазићи применом н-грама на нивоу знакова. Предност примене н-грама је што анализа стила не зависи од језика на ком је текст написан, информација о знаковима који се користе за раздвајање речи (нпр., празни знак, нови ред), употребе великих и малих слова, итд. Основни кораци овог приступа су:

- одабирање оптималног скупа н-грама на нивоу знакова који чине профил аутора,
- израчунавање сличности датих профила.

У приступу који разматрамо у овој секцији, профил аутора се дефинише као скуп података о учесталости јављања биграма на нивоу знакова у текстовима које је написао [5].

Пример 7.4. Посматрајмо азбуку $\Theta = \{а, б, в, г\}$, и текст „габабав“. У датом тексту постоји 6 биграма — биграми „га“ и „ав“ се јављају по једанпут,

а биграми „аб“ и „ба“ по двапут. За дату азбуку постоји $|\Theta|^2 = 4^2 = 16$ различитих биграма, и за сваки можемо да забележимо учесталост јављања у датом тексту, где је:

$$\text{учесталост биграма } (s_i s_j) = \frac{\text{број јављања биграма } (s_i s_j) \text{ у тексту}}{\text{укупни број биграма у тексту}} . \tag{7.13}$$

Подаци о учесталости јављања биграма у датом тексту чине профил текста. Профил текста посматраног у овом примеру је дат у табели 7.2. На слични начин се формира и профил аутора — забележе се учесталости биграма који се јављају у свим текстовима које је написао.

Табела 7.2 Профил текста.

биграм	учесталост
аа	0
ба	$\frac{2}{6}$
ва	0
га	$\frac{1}{6}$
аб	$\frac{2}{6}$
бб	0
вб	0
гб	0
ав	$\frac{1}{6}$
бв	0
вв	0
гв	0
аг	0
бг	0
вг	0
гг	0

Претпоставимо да имамо скуп профила познатих аутора, генерисаних на основу доступних текстова које су написали. Нека је T текст непознатог аутора. У општем случају, ауторство текста T се придружује оном аутору из скупа познатих аутора чији је профил *најсличнији* профилу текста T.

Једна од основних формула за израчунавање сличности профила је [2]:

$$\text{растојање}(P_1, P_2) = \sum_{x,y \in \Theta} (f_1(x,y) - f_2(x,y))^2 , \tag{7.14}$$

где су:

- P_1 и P_2 — профили који се пореде,
- Θ — азбука над којом су написани текстови,
- $f_1(x,y)$ — учесталост јављања биграма „xy“ у профилу P_1,

- $f_2(x,y)$ — учесталост јављања биграма „xy“ у профилу P_2.

Веће растојање између профила означава мању сличност профила, док мање растојање означава већу сличност профила.

Пример 7.5. Нека је дата азбука: {а, б, в, г}, и нека су аутор$_1$, аутор$_2$ и аутор$_3$ написали текстове: „аббг“, „вббаг“ и „авб“, респективно. Одредимо аутора текста „вабб“. Профили аутора и текста непознатог аутора су дати у табели 7.3. Применом формуле (7.14) можемо да проценимо сличност

Табела 7.3 Профили аутора и текста непознатог аутора.

биграм	учесталост			
	аутор$_1$	аутор$_2$	аутор$_3$	аутор$_?$
аа	0	0	0	0
ба	0	$\frac{1}{4}$	0	0
ва	0	0	0	$\frac{1}{3}$
га	0	0	0	0
аб	$\frac{1}{3}$	0	0	$\frac{1}{3}$
бб	$\frac{1}{3}$	$\frac{1}{4}$	0	$\frac{1}{3}$
вб	0	$\frac{1}{4}$	$\frac{1}{2}$	0
гб	0	0	0	0
ав	0	0	$\frac{1}{2}$	0
бв	0	0	0	0
вв	0	0	0	0
гв	0	0	0	0
аг	0	$\frac{1}{4}$	0	0
бг	$\frac{1}{3}$	0	0	0
вг	0	0	0	0
гг	0	0	0	0

профила сваког од познатих аутора са профилом текста непознатог аутора. Добијамо:

- растојање(аутор$_1$, аутор$_?$)$=\frac{2}{9}$,
- растојање(аутор$_2$, аутор$_?$)$=\frac{5}{12}$,
- растојање(аутор$_3$, аутор$_?$)$=\frac{5}{6}$.

Растојање између профила првог аутора и профила текста непознатог аутора је најмање ($\frac{2}{9}$), тј., профил првог аутора је најсличнији профилу текста непознатог аутора.

Формула (7.14) има један недостатак. У текстовима на основу којих се формирају профили аутора, учесталости различитих биграма могу знатно да варирају. Кад у обзир узмемо само апсолутну разлику њихових учесталости, као што је учињено у формули (7.14), биграми који се чешће јављају у тексту ће бити наглашенији у формирању профила,

јер су апсолутне разлике њихових учесталости у различитим текстовима веће. Да би сви биграми имали равномерне улоге у формирању профила, потребно је извршити нормализацију апсолутне разлике учесталости биграма [5]:

$$\text{растојање}(P_1,P_2)=\sum_{x,y\in\Theta}\Big(\frac{2\cdot(f_1(x,y)-f_2(x,y))}{f_1(x,y)+f_2(x,y)}\Big)^2. \tag{7.15}$$

Пример 7.6. Нека су учесталости биграма „s_1s_2“ и „s_3s_4“ у профилима P_1 и P_2 дате у табели 7.4. Иако су апсолутне разлике учесталости ових биграма у датим профилима једнаке, применом израза (7.15) се већа тежина приписује другом биграму.

Табела 7.4 Учесталости биграма у два профила.

биграм	профил P_1	профил P_2
„s_1s_2“	$f_1(s_1,s_2)=0,8$	$f_2(s_1,s_2)=0,7$
„s_3s_4“	$f_1(s_3,s_4)=0,1$	$f_2(s_3,s_4)=0,2$

Примена н-грама није ограничена на идентификовање аутора текста. Н-грами на нивоу речи и знакова се могу применити за класификовање и-мејл порука (нпр., коректне и нежељене поруке), тематско класификовање текста (нпр., спорт, економија, политика, култура), детектовање емоционалне обојености текста (нпр., позитивне и негативне рецензије производа), итд. Са методолошког аспекта, важно је приметити да се тренутно најпопуларнији аутоматски класификатори заснивају на софистицираним статистичким методама, али да су језички једноставни, тј., не захтевају дубоке увиде у природу језика [6].

7.3 Задаци

Задатак 7.1. Објасните Лапласову корекцију формуле за процену максималне изгледности, представљене изразом (7.7).

Задатак 7.2. Зашто збир вероватноћа (7.11) и (7.12) у примеру 7.1 није једнак 1?

Задатак 7.3. Применом израза (7.15), идентификујте непознатог аутора у примеру 7.5.

Задатак 7.4. Зашто су апсолутне и нормализоване разлике учесталости биграма у формулама (7.14) и (7.15) квадриране?

Задатак 7.5. Објасните зашто је описани приступ идентификовању аутора текста базиран на биграмима независан од језика.

Литература

1. Bayes T (1764) An Essay Toward Solving a Problem in the Doctrine of Chances. Philosophical Transactions of the Royal Society of London 53, pp. 370–418.
2. Bennett WR (1976) Scientific and engineering problem-solving with the computer. Prentice-Hall, Inc., Englewood Cliffs, New Jersey.
3. Manning CD, Raghavan P, Schütze H (2008) Introduction to Information Retrieval, Cambridge University Press.
4. Holmes D, Forsyth R (1995) The federalist revisited: New directions in authorship attribution. Literary and Linguistic Computing, 10:111–127.
5. Kešelj V, Peng F, Cercone N, Thomas C (2003) N-gram-based Author Profiles For Authorship Attribution, In Proceedings of the Conference Pacific Association for Computational Linguistics, PACLING'03, Dalhousie University, Halifax, Nova Scotia, Canada, pp. 255–264.
6. Norvig P (2009) Natural Language Corpus Data. In: Segaran T, Hammerbacher J (eds) Beautiful Data. The Stories Behind Elegant Data Solutions, O'Reilly Media.

Индекс појмова