Využití n-gramů pro odhalování plagiátů

Zdeněk Češka

Katedra informatiky a výpočetní techniky, Západočeská univerzita,
Univerzitní 8, 306 14 Plzeň, Česká republika
zceska@kiv.zcu.cz

Abstrakt *Stále rostoucí popularita Internetu a zvyšující se dostupnost různých dokumentů nám přináší i jisté problémy. Jedním z mnoha příkladů je množství pokusů o kopírování cizích prací, s vizí ulehčit si vlastní námahu. To s sebou přináší i rozvoj metod jak plagiátora identifikovat. Tento článek objasňuje metody pro detekci plagiátů a přibližuje náš výzkum v současnosti probíhající na ZČU. Zájem je věnován především metodě využívající n-gramy pro detekci překrývajících se částí dokumentů a odstranění problémů s posunem textu. K extrakci n-gramů vyšších řádů jsou na konci článku porovnány různé metody.*

1 Úvod

Se stále rostoucí popularitou Internetu se zvyšuje množství volně dostupných dokumentů. Pro uživatele Internetu je tak stále snadnější vyhledat vhodný dokument a místo pracného vymýšlení část dokumentu okopírovat a vydávat jej za vlastní práci. Ve školství se jedná o zvláště závažný problém, který je podpořen řadou serverů[1] nabízejících volně ke stažení různorodá témata. To samozřejmě působí nemalé škody, a proto se hledají různá řešení, jak tomu předcházet [1,2].

Ochrany znemožňující tisk a kopírování obsahu dokumentů prostřednictvím CTRL+C jsou nedostatečné. Často se podaří nalézt slabinu v zabezpečení a ochranu obejít. Navíc nejrůznější specializované ochrany obvykle vyžadují instalaci dodatečných nástrojů, což zabrání čtení příslušného dokumentu na počítačích, kde nejsou k dispozici administrátorská práva.

V boji proti plagiátorství je potřeba se vydat cestou vyhledávání dokumentů v rozsáhlých databázích. Vlastní myšlenka ochrany spočívá v psychologii, kdy si plagiátor kopírování rozmyslí, neboť by mohl být snadno odhalen při porovnání s databází již existujících dokumentů. Je zřejmé, že pro co nejvyšší účinnost musí databáze obsahovat velké množství existujících dokumentů. Faktorem úspěchu je též užití efektivních metod, které dokáží v krátkém čase a správně identifikovat plagiát.

Metoda relativních frekvencí, představená systémem SCAM [3], je založena na srovnání četností slov mezi dvěma dokumenty. Pozdější metody, jako je například hledání běžných trojic slov v systému Ferret [5], využívají kombinace slov pro nalezení částečných překryvů mezi dokumenty. Nejdále se dostávají metody využívající n-gramy vyšších řádů (5 a více), které identifikují překrývající se části textu a umožňují jejich vizualizaci. Metody postavené na n-gramech, stejně jako předešlé metody, jsou odolné vůči posunu textu uvnitř dokumentů. K němu dochází při mazání, vkládání nebo přepisování částí vět, což jsou obvyklé techniky pro zakrytí okopírovaného textu a znemožňují přesné porovnání textových řetězců.

[1] www.schoolsucks.com, www.cheathouse.com, www.seminarky.cz

2 Přehled metod pro detekci plagiátů

2.1 Metoda relativních frekvencí

Metoda založená na relativních frekvencích [3] vychází z tradičního vektorového modelu, který byl upraven pro nalezení podobnosti dvou dokumentů R a S. Do výpočtu jsou zahrnuta pouze slova, která splní stanovenou podmínku

$$\epsilon - \left(\frac{F_i(R)}{F_i(S)} + \frac{F_i(S)}{F_i(R)}\right) > 0, \qquad (1)$$

kde ϵ je konfigurovatelný parametr v intervalu $(2, \infty)$ a $F_i(D)$ je frekvence termu i pro dokument D. Pokud frekvence $F_i(R)$ nebo $F_i(S)$ je pro daný term nulová, podmínka není splněna.

Na základě podmínky (1) stanovíme množinu $c(R, S)$, která zahrnuje všechna běžná slova z obou dokumentů. Koeficient ϵ je tedy toleranční faktor určující zda bude dané slovo zahrnuto.

Mezi dokumenty R a S určíme dvě asymetrické míry podobnosti $subset(R, S)$ a $subset(S, R)$. V případě $subset(R, S)$ uvažujeme na kolik procent je dokument R podmnožinou druhého dokumentu S, kde

$$subset(R, S) = \frac{\sum_{w_i \in c(R,S)} \alpha_i^2 \cdot F_i(R) \cdot F_i(S)}{\sum_{i=1}^{N} \alpha_i^2 \cdot F_i^2(R)}. \qquad (2)$$

α_i představuje váhu asociovanou s výskytem termu i. Podobně odvodíme $subset(S, R)$. Výsledná podobnost je maximum z obou asymetrických měr

$$sim(R, S) = \max\{subset(R, S), subset(S, R))\}. \qquad (3)$$

Po výpočtu je nutné ověřit, zda $sim(R, S)$ není větší než jedna a hodnotu oříznout na interval $\langle 0, 1 \rangle$.

2.2 Hledání běžných trojic

Tato metoda, prezentovaná na systému Ferret [5], počítá podobnost dvou dokumentů z celkového počtu společných trojic slov. Aplikací Jaccard-Tanimoto koeficientu lze podobnost mezi dokumenty R a S vyjádřit vztahem

$$sim(R,S) = \frac{|triplet(R) \bigcap triplet(S)|}{|triplet(R) \bigcup triplet(S)|}, \qquad (4)$$

kde výstupem funkce $triplet(D)$ je množina všech extrahovaných trojic slov z dokumentu D.

2.3 Karp-Rabin algoritmus

U tohoto algoritmu [4], který je určen pro porovnání textových řetězců, si lze všimnout prvního náznaku užití n-gramů. K urychlení výpočtu se využívá hašovací funkce, jejíž výsledné hodnoty se porovnávají namísto znaků. Pro stanovení minimální shodné délky řetězců, která bude algoritmem registrována, zvolíme okénko velikosti k. Řetězec je reprezentován hašovací funkcí

$$H(c_1 \ldots c_k) = c_1 \cdot b^{k-1} + c_2 \cdot b^{k-2} + \ldots + c_{k-1} \cdot b + c_k, \quad (5)$$

kde c_i představuje ordinální hodnotu znaku a b je libovolná zvolená báze pro výpočet vhodné hašovací funkce. Výpočet následující hodnoty $c_2, c_3, \ldots, c_{k+1}$ lze urychlit použitím předchozích hodnot

$$H(c_2 \ldots c_{k+1}) = (H(c_1 \ldots c_k) - c_1 \cdot b^{k-1}) \cdot b + c_{k+1}. \quad (6)$$

Výslednou podobnost dokumentů R a S určíme z počtu stejných hašů při postupném porovnání všech podřetězců délky k mezi dokumenty R a S.

3 Užití n-gramů pro detekci plagiátů

Jak bylo naznačeno v úvodu, pro odhalení plagiátu je nutné určit společné části obou porovnávaných dokumentů. Na základě společných částí lze vyvodit procentuální podobnost a stanovením prahové hodnoty rozhodnout, zda se jedná či nejedná o plagiát. Často je vhodné ponechat konečné rozhodnutí na uživateli a jako nápovědu zobrazit překrývající se části dokumentů.

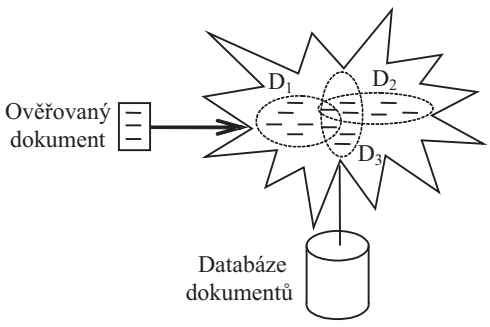
Protože práce s celým textem najednou je komplikovaná, metody hledající překryv společných částí, dělí text na množství malých kousků, které lze mezi sebou snadněji porovnat. Ze shody například 300 kousků textu z 1000 lze vyvodit, že překryv mezi dokumenty je 30%.

3.1 Dělení textu

Postupy pro dělení textu na menší části [6] lze shrnout se všemi výhodami i nevýhodami do následujících třech řešení:

1. Dokument je rozdělen dle vět. Nevýhodou je příliš veliký délkový rozptyl, který lze částečně kompenzovat dělením souvětí, ale i tak mohou být některé věty velmi krátké nebo naopak dlouhé.
2. Dokument je rozdělen podle předem definovaného slova nebo funkční hodnoty hašovací funkce, která může být společná pro více slov. Výhodou hašovací funkce je redukce potřebného paměťového prostoru. V tomto případě již nedochází k velkým rozptylům jako u předchozího řešení. Nevýhodou je jazyková závislost, která muže vést v nedosažitelnost požadované dělící hodnoty.
3. Dokument je rozdělen na pravidelné části po n slovech, což zajistí stejnou délku všech vytvářených částí textu. Pro úsporu paměťového prostoru lze opět využít hašovací funkce. Rozdělení textu *"Měna během posledního dne zažila na trhu drastický propad"* na pravidelné části po třech slovech bude vypadat následovně:
   - *"Měna během posledního"*
   - *"dne zažila na"*
   - *"trhu drastický propad"*

Po rozdělení dokumentů na menší části můžeme porovnávat jednotlivé části ověřovaného dokumentu vůči částem dokumentů uložených v databázi. Uchovávány jsou pouze unikátní části textu, které jsou sdíleny příslušnými dokumenty. Při vyhledávání se pro maximální urychlení využívá prostředků databáze a její schopnosti indexovat texty. Schéma příslušné databáze je zobrazeno na obrázku 1.



Obrázek 1. Databáze pro vyhledávání plagiátů

3.2 Odstranění problémů s posuvy textu

Při kopírování dokumentu nikdo nekopíruje celý dokument 1:1. Obvyklé se jedná o kopírování několika

vět nebo odstavců. Důslednější plagiátoři pak přeformulují část věty nebo alespoň nahradí některá slova. Vkládáním nebo mazáním textu se v dokumentu různě posouvá text. Tato změna ovlivní i tvorbu rozkouskovaného textu. Dojde například ke stavu, kdy v databázi budou uloženy části *"Měna během posledního"* a *"dne zažila na"*, kdežto náš vyhledávaný text bude *"během posledního dne"*. Při vyhledávání se jednotlivé části nepřekryjí a daná část nebude do výpočtu zahrnuta.

Řešením zmíněného problému je vytvořit z textu n-gramy. Při generování n-gramů řádu n, který se shoduje s n užitým pro dělení textu, získáme pro každou část textu všechny možné varianty posunu. Nevýhodou jsou zvýšené nároky na paměť, protože se počet částí textu zvýší n-krát. N-gramy stačí generovat pouze pro jednu ze stran, tzv. pro ověřovaný dokument nebo dokumenty uložené v databázi. Na opačné straně lze text rozdělit po n slovech.

Pro příklad použijeme předchozí text *"Měna během posledního dne zažila na trhu drastický propad"*, který pro úsporu místa převedeme na posloupnost čísel `1 2 3 4 5 6 7 8 9`. Posloupnost rozdělíme na pravidelné části po třech slovech ($n = 3$) a uložíme do databáze `1 2 3 | 4 5 6 | 7 8 9`.

V případě, že text někdo okopíruje a pokusí se o jeho zakrytí, může provést následující tři operace:

a) Odstranění slova z textu
- *"Měna během posledního dne zažila na trhu propad"*
- 3-gramy: `1 2 3 | 2 3 4 | 3 4 5 | 4 5 6 | 5 6 7 | 6 7 9`
- Shoda: `1 2 3` | `4 5 6` | ~~`7 8 9`~~

b) Vložení nového slova do textu
- *"Měna během posledního dne zažila na českém trhu drastický propad"*
- 3-gramy: `1 2 3 | 2 3 4 | 3 4 5 | 4 5 6 | 5 6` **`10`** `| 6` **`10`** `7 |` **`10`** `7 8 | 7 8 9`
- Shoda: `1 2 3` | `4 5 6` | `7 8 9`

c) Záměna slova v textu
- *"Měna během posledního dne prodělala na trhu drastický propad"*
- 3-gramy: `1 2 3 | 2 3 4 | 3 4` **`5a`** `| 4` **`5a`** `6 |` **`5a`** `6 7 | 6 7 8 | 7 8 9`
- Shoda: `1 2 3` | ~~`4 5 6`~~ | `7 8 9`

Z příkladu je vidět, že přidání, odebrání nebo záměna jednoho slova ve větě způsobí maximálně jednu neshodu při porovnávání n-gramů s rozkouskovaným textem. Z toho lze vyvodit, že při dělení dokumentu po n slovech, kde n je stupeň užitých n-gramů, by musela být provedena záměna nejméně každého n-tého slova pro kompletní zmatení algoritmu. V případě delších dokumentů je takováto záměna bez kompletního přepsání téměř nemožná.

3.3 Porovnávání dokumentů s databází

V předchozí sekci byla objasněna metoda porovnávající rozkouskovaný text s n-gramy. Celkem lze odvodit čtyři různé varianty, kde každá má jisté výhody i nevýhody. Tabulka 1 zobrazuje všechny možné varianty.

Varianta	Přístup ověř. dok.	Přístup databáze	Odolnost vůči posunu	Rychlost vyhledávání	Paměťové nároky
1:1	Dělený text	Dělený text	Ne	Vysoká	Nízké
1:N	Dělený text	N-gramy	Ano	Vyšší	Vysoké
N:1	N-gramy	Dělený text	Ano	Nižší	Nízké
N:N	N-gramy	N-gramy	Ano	Nižší	Vysoké

Tabulka 1. Metody porovnávání dokumentů s databází

Z uvedených přístupů je pro reálné nasazení a svou úspornost nejvhodnější varianta N:1. Varianta 1:N je vhodná v případě, že preferujeme maximální rychlost, bez ohledu na dostupný paměťový prostor. Poslední varianta N:N je spíše pro experimentální účely, kde se předpokládá hlubší zpracování textu a vyhledávání závislostí mezi dokumenty. Naproti tomu varianta 1:1 je z praktického hlediska nepoužitelná, protože není odolná vůči posunům textu, čímž je identifikace plagiátu značně znesnadněna.

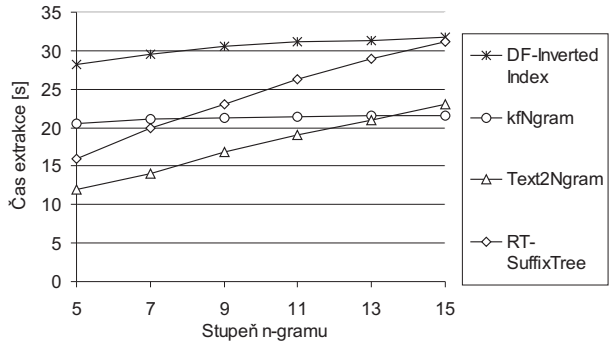
4 Porovnání metod extrakce n-gramů

Náš další výzkum je zaměřen na výběr významných n-gramů, které budou použity pro zvýšení přesnosti detekce plagiátů a zúžení množiny porovnávaných dokumentů. K tomuto účelu jsme provedli srovnání metod pro extrakci různých stupňů n-gramů a výpočet jejich četností.

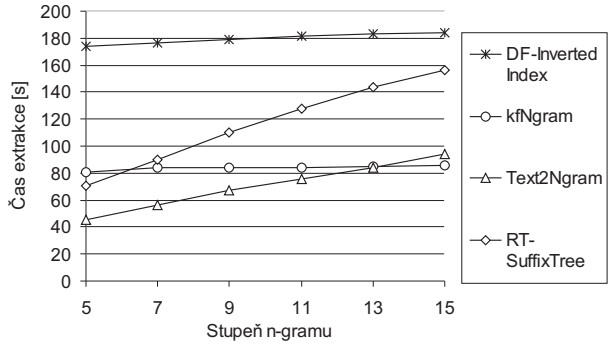
Do porovnání jsme zařadili aplikaci Text2Ngram [9] implementující metodu Suffix Tree, kfNgram [8] implementující metodu Suffix Array, vlastní implementaci Suffix Tree pojmenovanou RT-SuffixTree [7] a implementaci invertovaného indexu pojmenovanou DF-InvertedIndex [7]. Extrahovány byly n-gramy řádu 5, 7, 9, 11, 13 a 15 z testovacích korpusů o velikosti 5MB a 20MB. Testovací korpusy byly vytvořeny náhodným výběrem vět z korpusu Reuters Volume 1.

Na obrázcích 2 a 3 jsou zobrazeny výsledky našeho porovnání. K tomu byl použit počítač Intel Core 2 Duo E6600, 2GB RAM a 1TB HDD (2x500GB RAID-0) s instalovaným operačním systémem Windows XP Professional SP2.

Z provedených testů je vidět, že metoda Suffix Tree (aplikace Text2Ngram a RT-SuffixTree) je pro výpočet vyšší řádů nevhodná. Její výpočetní čas lineárně roste pro zvyšující se řád n-gramů. Naproti tomu metoda Suffix Array (aplikace kfNgram) má téměř konstantí



Obrázek 2. Extrakce n-gramů z 5MB korpusu



Obrázek 3. Extrakce n-gramů z 20MB korpusu

průběh pro vyšší řády. Metoda invertovaného indexu (aplikace DF-InvertedIndex) má podobný průběh, ale je několikanásobně pomalejší než Suffix Array.

5 Budoucí práce

Současnou metodu využívající n-gramy pro detekci plagiátů zamýšlíme zpřesnit bonifikací významných n-gramů během výpočtu. Kromě toho plánujeme provést testování vlivu různých délek n-gramů a jejich kombinací v prostředí českého jazyka.

Pro detekci plagiátu je nutné provést porovnání se všemi dokumenty v databázi. Našim dalším cílem je tvorba otisků dokumentů, které by zúžily množinu porovnávaných dat. Též předzpracování, omezující množství dokumentů na témata vztahují se k ověřovanému dokumentu, by zrychlilo výpočet.

6 Závěr

V tomto článku jsme nastínili problematiku plagiátorství a využití n-gramů pro odhalování plagiátů. Zároveň byly objasněny různé varianty aplikace n-gramů na straně ověřovaného dokumentu a databáze, včetně jejich výhod i nevýhod. Pro extrakci vyšších řádů n-gramů a výpočet jejich četností jsme provedli srovnání metod Suffix Tree, Suffix Array a invertovaného indexu. Z výsledků je patrné, že metoda Suffix Array dosahuje téměř konstatního času pro různé řády, a proto je lepší volbou pro naše experimentování. Náš další výzkum se ubírá směrem k bonifikaci významných n-gramů během výpočtu. Pro rychlejší detekci plagiátu zamýšlíme tvorbou vhodných otisků omezit množinu porovnávaných dat.

Tato práce byla částečně podporována z prostředků Národního Programu Výzkumu II, projekt 2C06009 (COT-SEWing).

Reference

1. Clough, P.: Plagiarism in natural and programming languages: An overview of current tools and technologies. Internal Report CS-00-05, Department of Computer Science, University of Sheffield, 2000.
2. Clough, P.: Old and new challenges in automatic plagiarism detection. Plagiarism Advisory Service, Volumen 10, 2003.
3. Shivakumar, N.,Garcia-Molina, H.: SCAM: A copy detection mechanism for digital documents. In Proceedings of 2nd International Conference in Theory and Practice of Digital Libraries, Austin, 1995.
4. Charras, C., Lecroq, T.: Handbook of Exact String-Matching Algorithms. King's College London Publications, 2004. ISBN 0-9543006-5-3.
5. Lane, P., Lyon, C., Malcolm, J.: Demonstration of the Ferret Plagiarism Detector. In Proceedings of the 2nd International Plagiarism Conference, NewcastleGateshead, 2006.
6. Pataki, M.: Plagiarism Detection and Document Chunking Methods. The Twelfth International Word Wide Web Conference, Budapest, 2003.
7. Tesar, R., Fiala, D., Rousselot, F., Jezek, K.: A comparison of two algorithms for discovering repeated word sequences. The 6th International Conference on Data Mining, Text Mining and their Business Applications, Skiathos, Greece, 2005. ISBN 1-84564-017-9.
8. Fletcher, W.: kfNgram. `http://www.kwicfinder.com/kfNgram/`
9. Zhang, L.: Text2Ngram. `http://homepages.inf.ed.ac.uk/s0450736/ngram.html`

Annotation

N-gram utilization for plagiarism detection
Growing popularity of Internet has brought the possibility to download a lot of different documents. This paper describes some common methods relating to the widely spread plagiarism. Employing n-grams is discussed in detail to detect overlapping documents and to avoid issues caused by text shifting. At the end of this paper we compare various methods for higher n-gram sizes extraction.