عنوان زیرپروژه:

امکان‌سنجی تولید سیستم تشخیص‌دهنده متون تقریباً یکسان زبان فارسی

فهرست مطالب

عنوان **شماره صفحه**

عنوان **شماره صفحه**

1 متون تقریباً یکسان، مفاهیم، تعاریف و تاریخچه

داده‌های مشابه یا تقریباً یکسان داسده‌هایی هستند که گرچه تطابق بیت به بیت با یکدیگر ندارند اما شباهت زیادی به هم دارند. در تصویر، ویدئو، صدا و متن ممکن است این داده‌ها به وجود بیایند.

در تصویر ، انواع مختلف عملیات روی یک تصویر ممکن است منجر به تولید یک تصویر مشابه شود، مثلاً روتوش شدن تصویر یک فرد در عکاسی یا تغییر اندازه آن تصویر دیگری ایجاد می‌کنند که گرچه با تصویر اول مساوی نیست اما با آن مشابه است. یک منبع دیگر برای تصاویر مشابه یا تقریباً یکسان هرزنامه‌های تصویری هستند. فرستندگان این تصاویر معمولاً برای فرار از شناخته شدن توسط برنامه-های تحلیل‌گر ایمیل با تغییر و اضافه کردن نویز به یک تصویر پایه، تصاویر تقریباً یکسان تولید کرده و برای افراد مختلف ارسال می‌کنند.

در داده‌های ویدئویی هم تغییر شدت نور یا تغییر زاویه دوربین می‌تواند داده‌های تقریباً یکسان تولید کند. تشخیص داده‌های مشابه در بازیابی محتوایی تصاویر و ویدئو و استخراج اطلاعات زمانی و مکانی از آنها مورد توجه است.

در داده‌های متنی به دلیل سهولت تغییر متن، داده‌های تقریباً یکسان یا مشابه راحت‌تر و به وفور تولید می‌شوند. ویرایش متون و مستندات مربوط به سیستم‌های مختلف و ایجاد نسخه‌های جدید از آنها، هرزنامه‌هایی که برای افراد مختلف ارسال می‌شود ، mirroring در سرورهای وب و استفاده غیر مجاز از منابع و صفحات موجود در اینترنت از منابع ایجاد داده‌های مشابه و تقریباً یکسان متنی هستند.

شناسایی این داده‌ها کاربردهای فراوانی دارد، به همین دلیل مورد توجه است. به عنوان مثال مطالعات اخیر نشان داده است که حدود30% الی 45% از صفحات وب نسخه یا کپی صفحات دیگر هستند [1,14]. از این رو شناسایی این صفحات مورد توجه موتورهای جستجو است.

در این گزارش پس از بررسی مساله شناسایی متون تقریباً یکسان، کاربردها و روش‌های، پیش نیازهای تولید سیستم تشخیص‌دهنده متون تقریباً یکسان فارسی را بررسی خواهیم کرد.

در فصل اول این گزارش ضمن معرفی شناسایی متون تقریباً یکسان و تاریخچه آم برخی از پیش نیازهای لازم برای فصل‌های بعدی از جمله روش‌های پیش‌پردازش متن را بررسی خواهیم کرد.

در فصل دوم روش‌ها و تکنیک‌های ارائه شده برای شناسایی متون تقریباً یکسان را به تفصیل بررسی می‌کنیم.

در فصل سوم چالش‌های پیش روی تشخیص متون یکسان و در فصل پنجم سیستم شناسایی متون تقریبا یکسان فارسی را بررسی می‌کنیم.

1.1 کاربردهای شناسایی داده‌ها و متون تقریباً یکسان

موضوع شناسایی متون تقریباً یکسان به دلایل و با اهداف مختلفی مطرح و دنبال می‌شود. به عنوان مثال مطالعات اخیر نشان داده است که حدود30% الی 45% از صفحات وب نسخه یا کپی صفحات دیگر هستند [1, 14]. به علاوه در [14] نشان داده شده است که این صفحات معمولاً ماهیت استاتیک و پایایی دارند؛ یعنی دو صفحه‌ای که مشابه هم تشخیص داده شده‌اند، به احتمال زیاد با گذشت زمان و پس از 10 هفته باز هم مشابه خواهند بود. تشخیص این صفحات مشابه مورد توجه موتورهای جستجو است چرا که این دسته از صفحات به طرق مختلف روی کارایی موتورهای جستجو تاثیر منفی دارند: 1-فضای لازم برای ذخیره ایندکس را افزایش می‌دهند، 2-نتایج جستجو را با برگشت نتایج مشابه تحت تاثیر قرار می‌دهند و 3-برنامه‌های خزنده وب[1] را درگیر بررسی تکراری می‌کنند.

همانطور که اشاره شد موتورهای جستجو برای بالابردن کارایی خود و رضایت کاربرانشان به شناسایی متون تقریباً یکسان علاقه‌مند هستند. در صفحات وب موارد مختلفی ممکن است صفحات تقریباً یکسان را ایجاد کنند.

از جمله منابع ایجاد صفحات تقریباً یکسان در وب می‌توان به موارد زیر اشاره کرد :

سایت‌هایی که اصطلاحاً mirror می شوند

سایت‌هایی که با نام‌های مختلف و در نتیجه آدرس‌های متفاوتی در اختیار کاربران قرار می‌گیرند

سایت‌هایی که با استفاده از تکنیک‌هایی متفاوت کوکی یا Session ID کاربر را در Url قرار می‌دهند.

[1] Web Crawlers

در تمام این موارد به جز آدرس‌های مورد ارجاع لینک‌ها، تصاویر و دیگر منابع موجود در صفحه ، مواردی که به صورت پویا یا اصطلاحاً داینامیک در صفحه بارگذاری می‌شوند، مثلاً تبلیغات، نیز باعث می‌شوند که 2 صفحه کاملاً یکسان و مشابه نباشند. دسته دیگری از صفحاتِ مورد علاقه، سایت‌های مختلفی هستند که محتوای یکسانی را در معرض نمایش قرار می‌دهند. سایت‌های خبری که اخبار را از منابع یکسانی دریافت و منتشر می‌کنند نمونه خوبی برای این دسته از سایت‌ها هستند. بسیاری از مقالاتی که در این زمینه منتشر شده‌اند و کارهایی که انجام شده‌اند به این کاربرد پرداخته‌اند. از آن جمله می‌توان [8, 22, 23] را نام برد.

یکی از کاربردهای دیگر شناسایی متون تقریباً یکسان، تشخیص متون ویرایش شده است که می‌تواند در مدیریت مجموعه مستندات یک سازمان به کار برود. در این زمینه نیز کارهای زیادی انجام شده است از جمله در [2, 19, 20, 24].

کاربرد دیگری که تشخیص متون یکسان در آن اهمیت دارد؛ شناسایی متون تقلبی یا به اصطلاح دزدی ادبی[1] است. استفاده از یک متن و تغییر برخی از جنبه های آن تا اصلی و غیر تقلبی به نظر برسد کار نسبتاً ساده ای است.

موضوع شناسایی متون تقریباً یکسان به صورت محدودتری همراه با موضوع کلی تر شناسایی رکورد های مشابه یا تقریباً مشابه در بانک‌های اطلاعاتی هم مطرح شده است و سابقه نسبتاً قدیمی‌تری نسبت به بحث مطرح فعلی دارد. این موضوع در بانک‌های اطلاعاتی با عناوینی همچون Record Linkage ، Set Similarity Join و Approximate Join مطرح است. برخی از روش‌های ارائه شده در این زمینه با روش‌های مطرح فعلی در شناسایی متون تقریباً یکسان شباهت‌هایی دارند.

بنابراین کاربردهای تشخیص متون تقریباً یکسان را می‌توان در موارد زیر دسته بندی کرد:

- حذف متون کپی یا تقریباً کپی هم [7, 8, 14]
- پیدا کردن و برگرداندن متون تقریباً کپی [9]
- تشخیص تقلب در کد نرم افزار [10]
- تشخیص متون تقلبی (دزدی علمی و ادبی) و ویرایش شده [2]

[1] Plagiarism

- اصلاح اشتباهات نوشتاری
- پیشنهاد عبارات جایگزین در موتورهای جستجو[1] [36]

1.2 متون تقریباً یکسان

شناسایی متون تقریباً یکسان در مقالات با عناوینی چون "Near Duplicate Detection"، "Record Linkage"، "Merge-Purge"، "Data Duplication" و "Name Matching" مطرح شده است.

متون تقریباً یکسان نیز در میان مقالات مختلف با عناوین و نام های متفاوتی مورد اشاره قرار گرفته اند. "Roughly The Same"عنوانی است که اولین بار توسط Broder و در [22] از آن استفاده شده است. این عبارت در دیگر مقالات کمتر مورد استفاده قرار گرفته است. " Near Duplicate" و "Duplicate" و "Inexact Duplicate" عبارات دیگری است به وفور در مقالات دیگر مثل [3, 4, 6, 8, 15] استفاده شده است.

عبارات دیگری مثل "Versioned" یا "Co-derivative" هم موارد دیگری هستند که بنا بر کاربرد مورد بررسی استفاده شده‌اند. این تنوع در نام بی‌دلیل نیست بلکه نشاندهنده گستردگی مفهوم و کاربرد است. در این گزارش از عبارت "**تقریباً یکسان**" که به هر دو عبارت "Roughly The Same" و " Near Duplicate" نزدیک است استفاده شده است.

در اغلب کاربردها دو متن را در صورتی "Duplicate" یا "Near Duplicate" می دانند که درصد بالایی از "اجزای آنها" با هم اشتراک داشته باشند. در مورد اجزای متن در بخش پیش‌پردازش متن به تفصیل بحث خواهیم کرد. اما در این فصل می‌توانیم فرض کنیم منظور از اجزای متن کلمات یا جملات هستند. این تعریف در کاربردهای شناسایی صفحات وبِ تقریباً یکسان بیشتر مورد توجه است.

در اغلب کاربردهایی که از عناوین "Versioned" یا "Co-Derivative"" استفاده شده است درصد شباهت مابین اجزای متن می‌تواند پایین‌تر باشد. در واقع درصد شباهت است که در مفهوم و کاربرد تفاوت ایجاد می کند. با وجود این گستردگی مفهوم و کاربرد به دلیل اشتراک روش‌های مورد استفاده، همه آنها را تحت عنوان روش‌های تشخیص متون تقریباً یکسان مورد بررسی قرار می‌دهیم.

[1] Query Refinement For Web Search

اینکه در چه صورتی 2 متن تقریباً یکسان هستند مساله مبهمی است و تعریف دقیقی برای آن ارائه نشده است. برخی از مقالات سعی در فرمولبندی و ارائه تعریف دقیق برای این مفهوم کرده‌اند. که در زیر می آید.

تعریف - متون تقریبا یکسان *- اگر فرض کنیم j تابعی باشد که شباهت دو متن را در بازه $[0,1]$ اندازه گیری می‌کند در این صورت دو متن A و B در صورتی تقریباً یکسان هستند که*

$$j(A,B) \geq 1-e, 0 < e << 1 \qquad (1)$$

در این تعریف j تابعی است که شباهت میان دو متن را محاسبه می‌کند. به عنوان مثال j می‌تواند رابطه کسینوسی، رابطه Jaccard یا عکس فاصله همینگ و ... باشد. این روابط در بخش بعدی همین فصل بررسی می‌شوند.

در اینجا ذکر این نکته لازم است که این تعریف به تعریف ارائه شده برای $e-approximate$ در نزدیکترین همسایگی تقریبی بسیار نزدیک است. از همین رو برخی از مهمترین روش‌های شناسایی متون تقریباً یکسان که تحت عنوان کلی روشهای تقریبی آنها را بررسی خواهیم کرد از روش‌های حل مساله نزدیکترین همسایگی تقریبی در ابعاد بالا است. چرا که یکی از ویژگی های مهم متن، ابعاد بالای داده‌های متنی است.

1.3 روابط موجود جهت بررسی شباهت

از روابط مختلفی می‌توان شباهت دو رشته یا دو متن را محاسبه کرد. در این بخش مهمترین آنها را بررسی می‌کنیم. در این بخش پس از بررسی روابط شباهت خواهیم دید که برخی از این روابط قابل تبدیل به یکدیگر هستند.

1.3.1 ضریب Jaccard

اگر فرض کنیم A و B دو مجموعه باشند در این صورت ضریب Jaccard برای این دو مجموعه به صورت (2) تعریف می‌شود :

$$J(A,B)=\frac{|A\cap B|}{|A\cup B|} \qquad (2)$$

برای استفاده از این رابطه برای اندازه گیری شباهت دو متن لازم است که هر متن را به صورت یک مجموعه نمایش دهیم. این مساله را در بخش روش‌های پیش پردازش متن مطالعه خواهیم کرد.

1.3.2 رابطه شباهت Cosine

اگر $\vec{A}$ و $\vec{B}$ دو بردار باشند دراین صورت رابطه کسینوسی بین این دو بردار به صورت زیر تعریف می‌شود

$$C(\vec{A},\vec{B})=\frac{\vec{A}.\vec{B}}{\|\vec{A}\|.\|\vec{B}\|} \qquad (3)$$

در واقع $C(\vec{A},\vec{B})$ کسینوس زاویه بین دو بردار $\vec{A}$ و $\vec{B}$ است. برای استفاده از این رابطه برای محاسبه شباهت مابین دو متن باید هر دو متن را به صورت بردار نمایش دهیم. اگر a_i نماینده عنصر i ام بردار $\vec{A}$ باشد و b_i هم به همین ترتیب نماینده عنصر i ام بردار $\vec{B}$ باشد؛ در این صورت $C(\vec{A},\vec{B})$ را به صورت رابطه (4) می‌توانیم بنویسیم.

$$C(\vec{A},\vec{B})=\frac{\sum_i a_i b_i}{\sqrt{|\vec{A}|}.\sqrt{|\vec{B}|}} \qquad (4)$$

و اگر بردارهای $\vec{A}$ و $\vec{B}$ نرمال شده باشند، یعنی طول آنها واحد باشد، $C(\vec{A},\vec{B})$ برابر است با ضرب داخلی دو بردار $\vec{A}$ و $\vec{B}$:

$$C(\vec{A},\vec{B})=\vec{A}.\vec{B} \qquad (5)$$

(6)

1.3.3 همپوشانی

اگر فرض کنیم A و B دو مجموعه باشند در این صورت رابطه همپوشانی برای این دو مجموعه به صورت رابطه (6) تعریف می‌شود:

$$O(A,B) = |A \cap B| \qquad (7)$$

مشابه با رابطه Jaccard در این رابطه نیز لازم است دو متن به صورت مجموعه نمایش داده شوند.

اگر A و B بردار باشند ضریب همپوشانی را می‌توانیم از رابطه (7) محاسبه کنیم.

$$O(\mathbf{A},\mathbf{B}) = \frac{\mathbf{A}.\mathbf{B}}{\min(|\mathbf{A}|,|\mathbf{B}|)} \qquad (8)$$

1.3.4 فاصله همینگ

اگر A و B دو مجموعه باشند فاصله همینگ آنها به صورت اندازه تفاضل متقارن آنها تعریف می‌شود :

$$H(A,B) = |(A-B) \cap (B-A)| \qquad (9)$$

اگر دو متن را به صورت دو مجموعه نمایش دهیم، شباهت بین آن دو متن با فاصله آنها نسبت معکوس خواهد داشت.

1.3.5 فاصله ویرایش

فاصله ویرایش[1] با فاصله Levenshtein بین دو رشته برابر است با تعداد کمینه عملگرهای ویرایشی که یکی از رشته ها را به دیگری تبدیل کند. که منظور از عملگر ویرایش عملیات درج، حذف و جایگزینی کاراکترهاست.

[1] Edit Distance

1.3.6 رابطه شباهت همانی

رابطه شباهت‌همانی[1] که در [2] و توسط Zobel و Hoad ارائه شده است بر این اساس کار می‌کند که کلماتی که به طور مشترک در 2 متن آمده‌اند می‌توانند نماینده این باشند که آیا 2 متن "هم ریشه"[2] هستند یا خیر. در این مقاله از اصطلاح هم ریشه برای متن‌هایی استفاده شده است که به طور مجاز یا غیرمجاز از هم یا متن دیگری گرفته شده‌اند. یعنی متون ویرایش شده به دست آمده از یک متن. بر این اساس گفته شده است کلمات مشترک در 2 متن هم ریشه به تعداد یکسانی ظهور می‌کنند؛ خصوصاً اگر کلمات پر استفاده‌ای نباشند[3] یا دیکته آنها اشتباه باشد.

1.3.7 ضریب Dice

اگر A و B دو بردار باشند، ضریب Dice به صورت رابطه (9) محاسبه می‌شود :

$$D(\vec{A},\vec{B}) = \frac{2|\vec{A}.\vec{B}|}{|\vec{A}|+|\vec{B}|} \qquad (10)$$

1.3.8 تبدیل روابط شباهت به یکدیگر

تنوع روابطی که می‌توان برای محاسبه شباهت متون از آنها استفاده کرد ممکن است این سوال را به وجود بیاورد که از کدامیک از این روابط می‌بایست استفاده کرد. در پاسخ به این سوال باید گفت این مساله بستگی به کاربرد و روش دارد. با این حال ذکر این نکته لازم است که بسیاری از این روابط قابل تبدیل به یکدیگر هستند. این حقیقت باعث می شود به راحتی بتوان در برخی موارد سیستمی که از یکی از این روابط استفاده می‌کند را طوری تغییر داد که از رابطه دیگری در محاسبه شباهت استفاده کند. در این بخش برخی از این روابط را بررسی می‌کنیم.

---

[1] Identity
[2] Co-Derivative
[3] Rare Words

فرض کنیم برای تشخیص تقریباً یکسان بودن دو متن A و B از ضریب Jaccard با حد آستانه t استفاده کرده‌ایم؛ در این صورت اگر بخواهیم از همپوشانی برای محاسبه شباهت استفاده کنیم چه حد آستانه ای را باید درنظر بگیریم؟

می‌توان نشان داد که

$$J(A,B) \geq t \Leftrightarrow O(A,B) \geq a = \frac{t}{1+t}(|A|+|B|) \qquad (11)$$

به همین ترتیب در مورد همپوشانی و فاصله همینگ می‌توان گفت:

$$O(A,B) \geq a \Leftrightarrow H(A,B) \leq |A|+|B|-2a \qquad (12)$$

1.4 تاریخچه شناسایی متون تقریباً یکسان

تشخیص داده‌های تقریباً یکسان خصوصاً در حوزه بانک‌های اطلاعاتی مدتهاست که مورد توجه است. از کارهایی که در این حوزه در باره این مساله انجام شده است می‌توان به [35,36,37] اشاره کرد.

اولین بار Manber در 1994 در [17] برای پیدا کردن فایل‌های تکراری در سیستم فایلی، Brin و دیگران در 1995 در [20] برای اجرا قانون کپی رایت، و Broder و همکارانش در 1997 برای کلاستر کردن نحوی وب [22] مساله شناسایی متن و فایل‌های تقریباً یکسان را بررسی کردند. این روش‌ها اغلب روش‌های تقریبی هستند.

در 2002، Charikar ضمن اشاره به شباهت این مساله با مساله نزدیکترین همسایگی، یکی از روش‌های نزدیکترین همسایگی در ابعاد بالا را برای این مساله پیشنهاد داد.

Lyon و همکاران در 2001 در [18] برای تشخیص تقلب، Conrad و همکارانش در 2003 در [19] مساله مدیریت نسخه‌های ویرایش شده و Bernstein و Zobel در 2005 در [21] برای جستجو در وب ، از روش‌های شناسایی متون تقریباً یکسان استفاده کردند.

1.5 پیش‌پردازش داده‌ها

پردازش متن به صورت خام امکانپذیر نیست و لازم است با انجام چند مرحله پیش‌پردازش متن را برای انجام پردازش‌های لازم روی آن آماده کرد. در فصل اول، دیدیم که تمام روابط محاسبه شباهت مورد استفاده نیازمند این هستند که متن به صورت خاصی مثل مجموعه یا بردار نمایش داده شود. در این بخش روش‌های مختلف پیش‌پردازش متن و تبدیل آن به نمایش‌های مختلف را بررسی می‌کنیم. در تمام روش‌های نمایش متن لازم است ابتدا متن به مجموعه‌ای از اجزا تبدیل شود. به همین دلیل قبل از اینکه نمایش‌های مختلف متن را بررسی کنیم، روش‌های استخراج اجزا متن را بررسی می کنیم.

1.5.1 استخراج اجزا یا تکه‌های متن

همانطور که اشاره شد در انواع فرمت‌های نمایش متن نیاز است که متن ابتدا به اجزا یا تکه‌هایی[1] تقسیم شود. در این بخش روش‌های مختلف استخراج تکه‌های متن را بررسی می‌کنیم. این روش‌ها را می توان در دو دسته تکه‌های همپوشان و غیرهمپوشان بررسی کرد. قبل از اینکه این روشها را بررسی کنیم لازم است در رابطه با اصطلاح توکن بحث کنیم.

توکن در واقع کلمه‌ای است که از کامپایلر وارد این بحث شده است و به هر مجموعه از کاراکترهای معنی دار متن توکن گفته می‌شود. قبل از هر گونه پیش‌پردازش لازم است توکن‌های متن استخراج شود یا به اصطلاح متن tokenize شود.

1.5.1.1 تکه‌های غیرهمپوشان

در برخی از کاربردها توکن‌ها، جمله‌ها و یا حتی زیر رشته‌های غیرهمپوشان با طول مشخص به عنوان تکه های متن در نظر گرفته شده اند. به عنوان مثال اگر متن "a rose is a rose.is a rose" را در نظر بگیریم پس از tokenize کردن آن خواهیم داشت :

(a,rose,is,a,rose,is,a,rose)

[1] Chunk

1.5.1.2 تکه‌های همپوشان

هر زیررشته nتایی از توکن‌ها را اصطلاحاً یک n-gram می‌گویند. در [22,23] نویسندگان مقاله به آنها shingle نام داده‌اند. اگر همان متن "a rose is a rose.is a rose" را در نظر بگیریم. تمام 4-gram ها یا shingle های با اندازه 4 در این متن عبارتند از :

{ (a,rose,is,a), (rose,is,a,rose), (is,a,rose,is), (a,rose,is,a), (rose,is,a,rose)}

گاهی در برخی از کاربردها n-gram ها از توکن‌ها تشکیل نشده اند بلکه از کاراکترهای متوالی در متن و یا حتی جمله‌های متولی تشکیل شده‌اند.

1.5.2 انتخاب تکه‌های متن

برخی از روش‌ها از تمام تکه‌های به دست آمده برای نمایش یک متن استفاده می‌کنند و برخی دیگر از آنها انواع تکنیک‌ها را برای انتخاب زیرمجموعه‌ای از این تکه‌ها به کار می‌برند. در زیر به بررسی این روش‌ها می‌پردازیم:

نمونه‌برداری بر اساس مکان: در این روش از هر m تکه متوالی یکی از تکه ها در مجموعه نماینده متن قرار می‌گیرد و بقیه حذف می‌شوند.

نمونه‌برداری بر اساس تابع درهم‌ساز[1] در این روش تکه‌های متن با استفاده از توابع درهم‌ساز به اعداد صحیح نگاشته می‌شوند و تنها تکه هایی در مجموعه نماینده متن قرار می‌گیرند که عدد صحیح اختصاص داده شده به آن بر m بخش‌پذیر باشند.

روش‌های مبتنی بر anchor : در این روش یک مجموعه از کلمات یا تکه‌های متن به عنوان anchor انتخاب می‌شوند و پس از آن برای انتخاب تکه‌های متن از این مجموعه استفاده می‌شود. به عنوان مثال در [17] تمام shingle هایی که با یکی از کلمات anchor شروع می‌شوند به عنوان نماینده متن انتخاب شده اند.

[1] Hash Functions

1.5.3 روش‌های مختلف نمایش متن

در این بخش دو مورد از روش‌های نمایش متن را که در الگوریتم‌های فصل سوم و چهارم استفاده شده-اند را بررسی می‌کنیم.

1.5.3.1 Bag-of-words

یک روش بسیار ساده در پردازش متن است. در این مدل متن به صورت مجموعه‌های نامرتب از تکه‌های متن نمایش داده می شود. با نمایش متن به این روش ترتیب قرار گرفتن کلمات در متن و اطلاعات گرامری آن از دست می‌رود.

1.5.3.2 Vector-Space-model

این روش یک روش جبری برای نمایش متن است و به آن روش term-vector-model نیز گفته می‌شود. در این روش متن به عنوان یک بردار نمایش داده‌می شود. هر بعد در این بردار منطبق بر یکی از تکه‌های متن است. تعداد ابعاد بردار متن به تعداد تمام تکه‌های مجزای موجود در مجموعه متن است. به ازای هر تکه متن اگر تکه مورد نظر در آن متن وجود نداشته باشد در موقعیت منطبق بر آن در بردار متن مقدار صفر قرار داده می شود و در غیر اینصورت مقدار غیر صفر. این مقدار بسته به کاربرد می تواند یک باشد که تنها نماینده وجود تکه مورد نظر در متن است یا می تواند مقداری باشد که نماینده وزن آن تکه در متن مورد نظر است. وزنی که به هر تکه در هر متن اختصاص داده می شود معمولاً تابعی از تعداد تکرار آن تکه در متن مورد نظر است. از تکنیک‌های وزن‌دهی برای این منظور می‌توان به tf و tf-idf اشاره کرد.

هر یک از کارهای مطرح در شناسایی متون تقریباً یکسان از یکی از این روش‌ها استفاده کرده‌اند. در [22]، نویسندگان هر کلمه را به عنوان یک توکن در نظر گرفته اند. در [27] هم پاراگراف‌ها به عنوان توکن در نظر گرفته شده است.

یک مساله مورد توجه در روش‌هایی که از shingle ها یا n-gram ها استفاده می‌کنند مقدار n است. در [8]، نویسندگان مقاله هر 5 کلمه متوالی را به عنوان یک shingle در نظر گرفته اند. در [2]،Zobel و

Hoad با بررسی تکه‌های با سایزهای متفاوت نشان داده‌اند که انتخاب هر 3 تا 5 کلمه به عنوان یک تکه بهترین نتیجه را خواهد داشت. البته باید توجه داشت که این عدد به زبان متون هم بستگی دارد. تمام متون مورد بررسی در این مقالات متون به زبان انگلیسی بوده‌اند و در مورد متون به زبان‌های دیگر این پارامترها باید در سیستم با توجه به آن زبان تنظیم شوند.

روش‌های متفاوتی برای انتخاب این اجزاء یا به اصطلاح تکه‌ها استفاده شده است. در [2]، Zobel و Hoad ضمن اشاره به نقش مهمی که مکانیزم انتخاب این تکه‌ها در نتایج به دست آمده دارند؛ ضمن بررسی انواع متدها نشان داده‌اند که روشهای anchor-based بهتر از دیگر روشها هستند.

2 روش‌های شناسایی متون تقریباً یکسان

اندازه‌گیری شباهت بین دو متن با استفاده از روابط شباهتی که در فصل قبل مطرح شدند به راحتی امکانپذیر است؛ در فصل قبل دیدیم که هریک از روابط شباهت با استفاده از نمایش خاصی مثل مجموعه یا بردار، شباهت بین دو متن را اندازه می‌گیرند. بنابراین کافی است با استفاده از روش‌های پیش‌پردازش مناسب هر متن را در قالب مناسبِ رابطه شباهت نمایش دهیم و سپس با استفاده از آن رابطه، شباهت متون را اندازه گیری کنیم.

پس از محاسبه شباهت بین دو متن در چه صورتی می‌توانیم دو متن را تقریباً یکسان بدانیم؛ یا به عبارت دیگر چه حد آستانه‌ای برای تقریباً یکسان بودن باید در نظر بگیریم؟ از کدامیک از روابط شباهت باید استفاده کنیم؟ تبدیل متن به نمایش‌های مورد انتظار در روابط شباهت چه تاثیری در نتایج خواهد داشت؟

اینها تمام ابهاماتی هستند که در محاسبه شباهت بین تنها دو متن و تشخیص تقریباً یکسان بودن باید به آنها پاسخ دهیم.

اگر مجموعه بزرگی از متون در اختیار داشته باشیم و به دنبال تقریباً یکسان‌ها در آن باشیم با چالش‌ها و مسائل دیگری مواجه خواهیم بود که مهمترین آنها زمان است. در اغلب کاربردها نیاز است که متون تقریباً یکسان مجموعه در کمترین زمان ممکن شناسایی شوند.

در این فصل ابتدا دسته‌بندی کلی روش‌های شناسایی متون تقریباً یکسان را بررسی می‌کنیم. پس از آن هر یک از این روشها را به طور جزئیتر مورد بررسی قرار می‌دهیم.

2.1 روش‌های مطرح در شناسایی متون تقریباً یکسان

روش‌های شناسایی متون تقریباً یکسان را می‌توانیم از لحاظ شکل جستجو به روش‌های یک‌به چند و چندبه چند تقسیم کنیم.

به علاوه استفاده از روشهای معنایی یا آماری و ساختاری در محاسبه شباهت متون هم جنبه دیگری است که می‌توان روشهای شناسایی را از یکدیگر متمایز کرد. روشهای معنایی بیشتر در کاربرد شناسایی

تقلب و در کنار روشهای آماری و ساختاری استفاده شده‌اند و نسبت به روشهای ساختاری و آماری کمتر استفاده شده‌اند. از اینرو در این گزارش بیشتر به روشهای ساختاری و آماری می‌پردازیم و در انتهای این فصل، در بخشی جداگانه روشهای معنایی را بررسی می‌کنیم.

همانطور که اشاره شد، در مقابل روشهای معنایی، روشهای آماری و ساختاری قرار دارند که از جنبه های ساختاری و آماری متن برای محاسبه شباهت استفاده می‌کنند. در این فصل این روشها را به تفصیل بررسی می‌کنیم. این روشها را می توان از لحاظ نحوه محاسبه میزان شباهت متون به روش‌های دقیق و تقریبی تقسیم کرد.

2.1.1 روش‌های یک‌به چند و روش‌های چندبه چند

همانطور که گفته شد، به طور کلی و بدون اینکه وارد جزئیات روش‌های پیاده سازی شده برای این کار بشویم می توانیم از یک زاویه این روش‌ها را در 2 دسته کلی روش‌های یک‌به چند[۱] و روشهای چندبه چند[۲] قرار دهیم. این دسته‌بندی از نظر نحوه تعریف مساله انجام می‌شود. روش‌های یک‌به چند مساله را به صورت پیدا کردن تمام متن‌های مشابه و تقریباً یکسان با متن مورد پرسش[۳] تعریف می‌کنند. روشهای چندبه چند نیز تمام متون مشابه با هم و تقریباً یکسان را در یک کلاستر قرار می‌دهند.

یک روش ابتدایی برای رسیدن به روش‌های چندبه چند این است که روش یک‌به چند را به تعداد متن-های مجموعه مورد بررسی تکرار کنیم. اشکال این روش در زمان لازم برای انجام آن است و روش های چندبه چند نیز از این دید اهمیت دارند. در [22]، Broder و همکاران با استفاده از روشی که ارائه کرده-اند توانسته‌اند در زمان O(n log n) وب را اصطلاحاً به صورت نحوی کلاستر کنند و صفحات مشابه با هم را در یک کلاستر قرار دهند. نویسندگان در این مقاله به این نکته اشاره کرده‌اند که رابطه تقریباً یکسانی و مشابه بودن یک رابطه تراگذاری نیست و از این رو یک صفحه می‌تواند در چندین کلاستر قرار بگیرد. در [27,28] نیز نویسندگان مجموعه داده‌های خود را به کلاسترهایی از متون مرتبط کلاستربندی کرده-اند. در [28] نیز چند نوع ارتباط مابین متون با استفاده از درصد شباهت آنها و نیز ساختار مجموعه داده

[۱] 1-To-N

[۲] N-To-N

[۳] Query Document

مورد بررسی تعریف شده است و پس از آن با استفاده از این اطلاعات نوعی کلاسترینگ با شبه نظارت[1] انجام شده است.

2.1.2روش‌های دقیق و روش‌های تقریبی

علاوه بر دسته‌بندی روش‌ها به روش‌های یک‌به چند و چندبه چند، روشهای موجود برای شناسایی متون تقریباً یکسان را می‌توان به دو دسته کلی روش‌های دقیق و روش‌های تقریبی تقسیم کرد. این تقسیم-بندی بر اساس محاسبه دقیق یا تقریبی شباهت متون توسط آنها انجام شده است.

روش‌های دقیق معمولاً بر اساس نمایه‌های معکوس،ایندکس معکوس، کار می‌کنند و نام روش‌های دقیق بر روی آنها به دلیل این نهاده شده است که این روش‌ها مقدار دقیق و واقعی شباهت بین دو متن را محاسبه و اندازه گیری می‌کنند و بر مبنای آن عمل می‌کنند. این روش‌ها به دلیل اینکه مقدار شباهت بین دو متن را به طور دقیق محاسبه می‌کنند نسبتاً کند هستند. برای مقابله با این کندی اکثر این روش‌ها با استفاده از نمایه‌های معکوس و روش‌های فیلترینگ مناسب سعی می‌کنند و برای هر متن تعداد متن‌هایی که باید با آن مقایسه شوند را کاهش دهند.

روش‌های تقریبی در مقابل روش‌های دقیق قرار می‌گیرند. این روش‌ها برای کاهش زمان جستجو و بالا بردن سرعت خود با استفاده از روش‌هایی که در فصل چهار خواهیم دید میزان شباهت دو متن را تخمین می‌زنند. و با استفاده از مقدار تقریبی محاسبه شده در مورد تقریباً یکسان بودن دو متن قضاوت می‌کنند. این روشها نسبت به روش‌های دقیق سریعتر هستند اما به دلیل استفاده از مقدار تقریبی شباهت متون، نرخ خطای مثبت در آنها بالا تر است.

شناسایی متون تقریباً یکسان در واقع حالت خاصی از مساله نزدیکترین همسایگی است. برای مساله نزدیکترین همسایگی در ابعاد بالا روشها و الگوریتم های تقریبی متفاوتی ارائه شده‌اند. این روش‌ها اغلب سعی در کاهش ابعاد و اندازه بردار ویژگی داده‌ها دارند. روش‌های تقریبی شناسایی متون تقریباً یکسان بر مبنای روش‌های ارائه شده برای نزدیکترین همسایگی در ابعاد بالا هستند.

[1] Semi-Supervised Clustering

2.2 روش‌های دقیق

روش‌های دقیق معمولاً بر اساس نمایه‌های معکوس[1] کار می‌کنند و نام روش‌های دقیق بر روی آنها به دلیل این نهاده شده است که این روش‌ها مقدار دقیق و واقعی شباهت بین دو متن را محاسبه و اندازه گیری می‌کنند و بر مبنای آن عمل می‌کنند. این روش‌ها به دلیل اینکه مقدار شباهت بین دو متن را به طور دقیق محاسبه می‌کنند نسبتاً کند هستند و روشهای مطرح در این زمینه اغلب برای مقابله با این کندی با استفاده از نمایه‌های معکوس و روشهای فیلترینگ مناسب سعی در بهبود کارایی این روشها دارند. البته استفاده از نمایه‌های معکوس شاخص اصلی این الگوریتم‌ها نیست و برخی از این الگوریتم‌ها از روشهای دیگری استفاده کرده‌اند.

همانطور که قبلاً هم اشاره شد بحث شناسایی تقریباً یکسان‌ها در بانک‌های اطلاعاتی با عناوینی چون data cleaning و record linkage مطرح است. ذکر این نکته لازم است که اکثر روش‌های مطرح شده در این فصل از روش‌های مطرح شده در حیطه بانک‌های اطلاعاتی هستند. در این الگوریتم‌ها معمولاً داده‌ها رکوردهای متنی هستند. به همین دلیل در این بخش اصطلاحات "متن" و "رکورد" به طور مکرر به جای یکدیگر به کار رفته‌اند. در مباحث بانک‌های اطلاعاتی این الگوریتم‌ها اغلب با نام‌هایی چون set Similarity، Similarity Join و Approximation Join مطرح هستند.

روش‌های دقیق را می‌توان به 2 دسته کلی تقسیم کرد :

- روش‌های مبتنی بر نمایه‌های معکوس
- روش‌های فیلترینگ پیشوندی و پسوندی

روش‌های مبتنی بر فیلترینگ پیشوندی و پسوندی در واقع توسعه روشهای مبتنی بر نمایه‌های معکوس هستند و با استفاده از اصول فیلترینگ پیشوندی و پسوندی سعی در کاهش اندازه نمایه معکوس و بهبود کارایی دارند.

[1] Inverted Index

2.2.1روش‌های مبتنی بر نمایه‌های معکوس

نمایه معکوس اصطلاحی است که در بازیابی اطلاعات مطرح است. این نمایه هر کلمه را به لیستی از رکوردها یا متن‌هایی که کلمه مورد نظر در آنها آمده است می‌نگارد. در این بخش الگوریتم Probe-Count [35] بسط‌هایی که بر روی آن انجام شده‌اند را به عنوان نمونه‌ای از روش‌های مبتنی بر نمایه معکوس مورد بررسی قرار می‌دهیم.

Probe-Count 2.2.1.1

الگوریتم Probe-Count اولین الگوریتمی است که بر پایه نمایه معکوس کار می‌کند [35]. در این الگوریتم برای پیدا کردن تمام متن[1] هایی که همپوشانی آنها از حد آستانه t بیشتر است به روش زیر عمل می‌شود:

ابتدا و با یکبار اسکن کردن تمام متن‌ها نمایه معکوس ساخته می‌شود.

برای هر متن، برای هر کلمه موجود در آن، با استفاده از نمایه معکوس لیستی از متن‌هایی به دست می‌آید که با متن مورد نظر در کلمه مورد نظر اشتراک دارند. به این ترتیب برای هر متن و برای هر کلمه، لیستی از متن‌هایی به دست می‌آیند که با متن مورد نظر اشتراک دارند.

با ادغام این لیست‌ها و شمارش تعداد کلمات مشترک بین متن مورد نظر و متن‌های بدست آمده می‌توان مجموعه متن‌هایی که همپوشانی آنها با متن مورد نظر از حد آستانه t بیشتر است را به دست آورد.

در مراحل بالا تعداد کلمات مشترک هر دو متن شمرده می‌شود. برای بسط این روش می‌توان فرض کرد به هر کلمه در مجموعه متون وزنی اختصاص داده شده است و به جای شمارش تعداد کلمات مشترک، مجموع وزن کلمات مشترک ملاک قرار بگیرد.

ادغام لیست متن‌ها زمانبرترین بخش الگوریتم است. برای تسریع در این بخش می‌توانیم لیست کلمات در نمایه معکوس را به صورت مرتب شده نگهداری کنیم. و باز هم برای سریع تر کردن مرحله سوم می-

[1] در [35] از کلمه "رکورد" برای اشاره به رکوردهای متنی استفاده شده است. در اینجا برای هماهنگی با سایر فصل ها از کلمه "متن" استفاده شده است.

توان از ساختار داده heap استفاده کرد. برای هر لیست اشاره‌گری در نظر گرفته می‌شود که به ابتدای آن اشاره می‌کند. این اشاره‌گرها همه در یک heap درج می‌شوند. پس از آن در pop های متوالی و تا وقتی که اشاره‌گرهای pop شده به متن یکسانی اشاره می‌کنند وزن‌های کلماتِ لیستی که اشاره‌گر مربوط به آن pop شده است جمع می‌شوند و با حد آستانه مورد نظر مقایسه می‌شوند.

اگر هر متن به طور متوسط شامل n کلمه باشد و طول لیست کلمه w را در نمایه معکوس n_w بنامیم در این صورت با استفاده از heap پیچیدگی زمانی این الگوریتم از مرتبه $O\left(\sum_w n_w^2 \log(n)\right)$ خواهد بود.

در [35] الگوریتم دیگری به نام pair-count ارائه شده است. این الگوریتم مشابه الگوریتم probe-count با یکبار اسکن کردن مجموعه متون، نمایه معکوس را می‌سازد. در مرحله بعد با استفاد از لیست‌های کلمات در نمایه معکوس، تمام جفت صفحاتی که در آن کلمه با هم اشتراک دارند ساخته می‌شوند. اگر تعداد متون لیست مربوط به کلمه w را l_w بنامیم و طول آن را n_w در این صورت برای هر کلمه $\frac{n_w(n_w-1)}{2}$ جفت متن به دست می‌آید. در جفت متن‌های به دست آمده جفت‌های تکراری به وفور وجود خواهند داشت؛ چرا که متن‌هایی وجود دارند که در بیش از یک کلمه با هم اشتراک دارند. در مرحله بعد و برای به دست آوردن مجموع وزن‌های کلمات مشترک جفت متن، با استفاده از تابع در هم-سازی شناسه متن‌ها در هم‌سازی می‌شوند و مجموع وزن‌ها محاسبه می‌شود.

در ارزیابی‌های انجام شده در [35] الگوریتم اشاره شده pair-count به حافظه بسیار زیادی احتیاج دارد. به عنوان مثال با مجموعه داده‌ای با 20000 متن، و در سیستمی با 1 GB حافظه اصلی اجرای این الگوریتم به دلیل کمبود حافظه متوقف شده است. در حالی که الگوریتم Probe-Count برای اجرا شدن بر روی مجموعه داده ای با 50000 متن و حد آستانه 80% نیاز به حدود 90 دقیقه زمان دارد.

2.2.1.2 Probe-stopWords

پراکندگی کلمات در بسیاری از مجموعه داده‌های واقعی بسیار نامتوازن است و بیشتر زمان صرف شده در مرحله ادغام در الگوریتم probe-count در لیست‌های مربوط به چند کلمه است که بسیار طولانی است. اینها در واقع کلماتی هستند که به دلیل کاربرد زیادشان در بسیاری از متن‌های مجموعه داده وجود دارند و به همین دلیل لیست مربوط به آنها در نمایه معکوس طولانی و بلند است. این کلمات در بازیابی اطلاعات با نام stop word یا noise word شناخته می‌شوند و در ساختن نمایه معکوس حذف می‌شوند و هیچ لیستی برای آنها در نمایه معکوس ساخته نمی‌شود. با استفاده از همین ایده می‌توان

الگوریتم probe-count را بهبود داد. در [35] از این ایده به این صورت استفاده شده است که پس از حذف پرتکرارترین کلمات از ایندکس معکوس؛ در محاسبه همپوشانی برای هر متن حد آستانه مربوط به آن متن متناسب با تعداد stop word های آن متن کاهش داده شده است. الگوریتم تغییر یافته به این روش probe-stopWord نامیده شده است.

2.2.1.3 Probe-optMerge

الگوریتم probe-stopWord را می‌توان باز هم بهبود داد. این بهبود با یک تغییر ساده در مرحله ادغام بدست می‌آید. برای این کار کافی است توجه کرد که برای تعیین اینکه آیا همپوشانی متنی که در لیست‌های کلمات مربوط به یک متن آمده با آن متن از حد آستانه بیشتر است یا خیر لازم نیست تمام لیست‌ها بررسی شوند. بلکه لیست کلمات هر متن را تنها تا جایی بررسی می کنیم که مطمئن شویم این دو متن استراک لازم با یکدیگر را ندارند.

فرض کنیم $A=l_1l_2...l_t$ لیست‌های مربوط به کلمات، $w_1, w_2, ...w_t$ یک متن هستند که به ترتیب صعودی مرتب شده‌اند. و نیز فرض کنیم.

(13)

$$Colmulative Wt(l_i)=\sum_{j=1}^{i} weight(w_j)$$

و نیز فرض کنیم $L=l_1l_2...l_k$ که در آن $k=\max\arg_n(Wt(l_n)<t)$ و S= A-L. در این صورت کافی است تنها لیست‌های موجود در S را ادغام کنیم. در حین ادغام و پس از محاسبه مجموع وزن مشترک متن مورد بررسی، اگر مجوع وزن‌ها و $Colmulativ\ eWt(lk)$ از حد آستانه کمتر بود نیازی به بررسی لیست‌های موجود در L نیست، چرا که حتی اگر متن مورد نظر در تمام لیست‌های L آمده باشد هم مجموع وزن‌ها به حد آستانه نخواهد رسید. در شبه کد شکل 1 الگوریتمِ مرحله ادغام بهبود یافته آمده است.

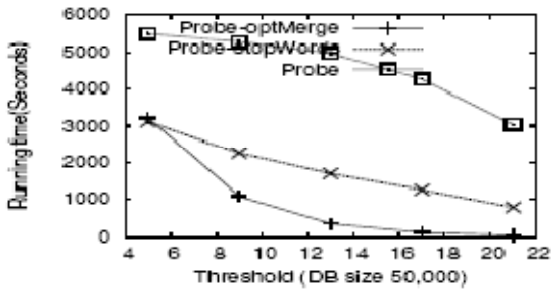
Algorithm 1 MergeOpt(r, T, I)

Let $A = l_1, l_2, \ldots l_t$ be the record lists of index I in decreasing order of length corresponding to the t words $w_1 \ldots w_t$ of r
Compute cumulativeWt(l_i) $= \sum_{j=1}^{i}$ weight(w_j)
$L = l_1, l_2, \ldots, l_k$ such that k is the largest index for which cumulativeWt(l_k) $< T$
Insert frontiers of lists $S = A - L$ in a heap H.
while H not empty **do**
  pop from H current minimum record m along with total weight $m.w$ of all lists in H where m appears
  push in H next records from lists in S that popped.
  for $i = k$ down to 1 **do**
    if ($m.w$+ cumulativeWt(l_i) $< T$) **exit-for**;
    search for m in l_i using a doubling binary search method, and if found,
      increment $m.w$ with weight(word(l_i)).

شکل 1- شبه کد مرحله ادغام بهبود یافته[35]

الگوریتم تغییر یافته به این روش probe-optMerge نامیده شده است.

نتایج ارزیابی‌ها در [35] نشان داده است که probe-optMerge در حد آستانه‌های متفاوت 5 تا 100 بـار از probe-count سریعتر است.



شکل 2 - مقایسه زمان اجرای 3 الگوریتم Probe-count، Probe-stopWords و Probe-optMerge [35]

الگوریتم Probe-count و بهبودهای انجام شده بر روی آن در حافظه اصلی اجرا می‌شوند. نمایه معکـوس در حافظه ساخته می‌شود و عملیات ادغام تماماً در حافظه انجام می‌شود. در صورتی که حجـم مجموعـه داده زیاد باشد و یا متن‌های موجود در مجموعه داده طولانی باشند، اجرای الگـوریتم بـا مشـکل مواجـه می‌شود. در [35]، نسخه خارج از حافظه Probe-optMerge نیز ارائه شده است.

2.2.2 روش‌های فیلترینگ پیشوندی و پسوندی

در بخش قبل راجع به روش‌های مبتنی بر نمایه‌های معکوس بحث شد. الگوریتم Probe-count و جزئیات آن بررسی شد و شرح داده شد که چطور با استفاده از حد آستانه شباهت، کارایی الگوریتم بهبود داده شد و الگوریتم بهبود یافته probe-optMerge به دست آمد.

در این بخش روش‌های دیگری را بررسی می‌کنیم که باز هم با استفاده از حد آستانه شباهت سعی در بهبود روش‌های مبتنی بر نمایه دارند. روش‌هایی که در این بخش بررسی می‌شوند نیز مبتنی بر نمایه‌ها هستند، این روش‌ها می‌توانستند در بخش قبلی و تحت عنوان روش‌های مبتنی بر نمایه‌سازی بررسی شوند. با این حال به دلیل رویکرد خاص این روش‌ها در کاهش حجم نمایه معکوس و روش انجام آن، این روش‌ها در این بخش و به طور جداگانه بررسی می‌شوند. الگوریتم‌های این بخش با استفاده از حد آستانه و دیگر مشخصات مساله سعی می‌کنند تا از ساختن نمایه معکوس به طور کامل اجتناب کنند. در این بخش نیز 2 الگوریتم، All-pairs و PP-join را بررسی می‌کنیم.

2.2.2.1 الگوریتم All-pairs

در بخش قبل دیدیم که چطور الگوریتم Probe-optMerge با استفاده از حد آستانه، بررسی برخی از لیست‌های جدول نمایه را کنار گذاشت و از این طریق به سرعت 5 تا 100 برابر الگوریتم اولیه دست یافت.

الگوریتم All-pairs که در [26] ارائه شده است برخلاف [35] از تمام کلمات متن‌ها در ساختن نمایه معکوس استفاده نمی‌کند بلکه با استفاده از حد آستانه t تنها از بخشی هر متن در ساختن نمایه استفاده می‌شود.

ساختن نمایه با استفاده از تنها بخشی از متن، ضمن تسریع مرحله ساخت نمایه باعث کاهش حجم نمایه و لیست مربوط به هر کلمه نیز می‌شود و این مساله موجب کاهش زمان دسترسی به لیست‌های نمایه و کاهش تعداد متن‌هایی که باید در هر لیست بررسی شوند می‌شود. الگوریتم All-pairs در یک مجموعه متن به دنبال جفت متن‌هایی است که ضرب داخلی بردار وزن آنها از حد آستانه t بزرگتر باشد.

در شکل 2 الگوریتم All-pairs0 را می‌بینیم، این الگوریتم، الگوریتم پایه است و از تمام کلمات هر متن در ساختن نمایه استفاده می‌کند.

```
ALL-PAIRS-0(V, t)
    O ← ∅
    I_1, I_2, ..., I_m ← ∅
    for each x ∈ V do
        O ← O ∪ FIND-MATCHES-0(x, I_1, ..., I_m, t)
        for each i s.t. x[i] > 0 do
            I_i ← I_i ∪ {(x, x[i])}
    return O
```

```
FIND-MATCHES-0(x, I_1, ..., I_m, t)
    A ← empty map from vector id to weight
    M ← ∅
    for each i s.t. x[i] > 0 do
        for each (y, y[i]) ∈ I_i do
            A[y] ← A[y] + x[i] · y[i]
    for each y with non-zero weight in A do
        if A[y] ≥ t then
            M ← M ∪ {(x, y, A[y])}
    return M
```

شکل 3 - الگوریتم All-pairs0 [36]

همانطور که دیده می‌شود، این الگوریتم ضمن ساختن نمایه معکوس، متن‌هایی که شباهت آنها از حد آستانه بیشتر است را تشخیص می‌دهد.

برای بهبود این الگوریتم، الگوریتم All-pairs1 به صورت شکل 4 با تغییر All-pairs0 ارائه شده است:

```
ALL-PAIRS-1(V, t)
    Reorder the dimensions 1...m such that dimensions with
      the most non-zero entries in V appear first.
    Denote the max. of x[i] over all x ∈ V as maxweight_i(V).
    O ← ∅
    I_1, I_2, ..., I_m ← ∅
    for each x ∈ V do
        O ← O ∪ FIND-MATCHES-1(x, I_1, ..., I_m, t)
        b ← 0
        for each i s.t. x[i] > 0 in increasing order of i do
            b ← b + maxweight_i(V) · x[i]
            if b ≥ t then
                I_i ← I_i ∪ {(x, x[i])}
                x[i] ← 0                ;; create x'
    return O
```

```
FIND-MATCHES-1(x, I_1, ..., I_m, t)
    A ← empty map from vector id to weight
    M ← ∅
    for each i s.t. x[i] > 0 do
        for each (y, y[i]) ∈ I_i do
            A[y] ← A[y] + x[i] · y[i]
    for each y with non-zero weight in A do
        ;; Recall that y' is the unindexed portion of y
        s ← A[y] + dot(x, y')
        if s ≥ t then
            M ← M ∪ {(x, y, s)}
    return M
```

شکل 4 - الگوریتم All-pairs1 [36]

All-pairs1 با تغییر حلقه اصلی All-pairs0 با شروع از کلمات با وزن بیشتر به وزن کمتر، کلمات را تا برقرار شدن شرط خاصی نمایه سازی نمی‌کند.

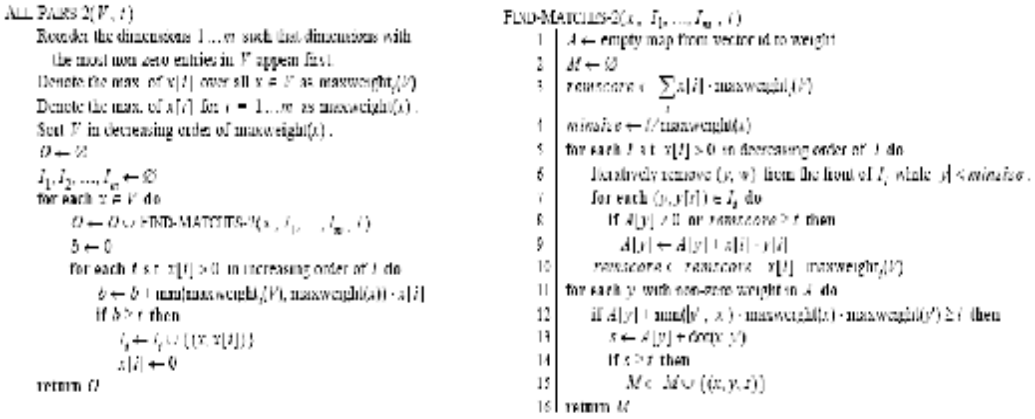
شرط مورد نظر این است که اگر متن مورد پردازش را x در نظر بگیریم، و فرض کنیم بردار متن به ترتیب نزولی وزن مرتب شده باشد، با شروع از اولین کلمه و تا کلمه j ام کلمات نمایه سازی نشوند. j

ایندکسی از بردار متن x است که مجموع وزن‌های هر کلمه ضربدر ماکسیمم وزن آن کلمه در کل مجموعه با شروع از اولین کلمه از حد آستانه t بیشتر نشود. این شرط تضمین می‌کند که در صورتی که متنی مانند y وجود داشته باشد که تمام کلمات از در محل های 1 تا j را دشته باشد و وزن این کلمات در y ماکسیمم وزن ممکن برای آن کلمه باشد، با بقیه کلماتی که نمایه‌سازی شده‌اند بتوان y را شناسایی کرد.

با توجه به اینکه تمام کلمات متن نمایه‌سازی نشده‌اند، با استفاده از نمایه به دست آمده می‌توان عدم برآورده شدن حد آستانه برای دو متن x و y را تشخیص داد. به عبارت دیگر با استفاده از نمایه به دست آمده برای هر متن x مجموعه ای از متن‌هایی به دست می‌آید که ممکن است حد آستانه شباهت برای آنها برقرار باشد. این مجموعه را مجموعه کاندیداها می‌نامیم. متن‌هایی که در مجموعه کاندیداها نیستند را می‌توان با اطمینان کنار گذاشت.

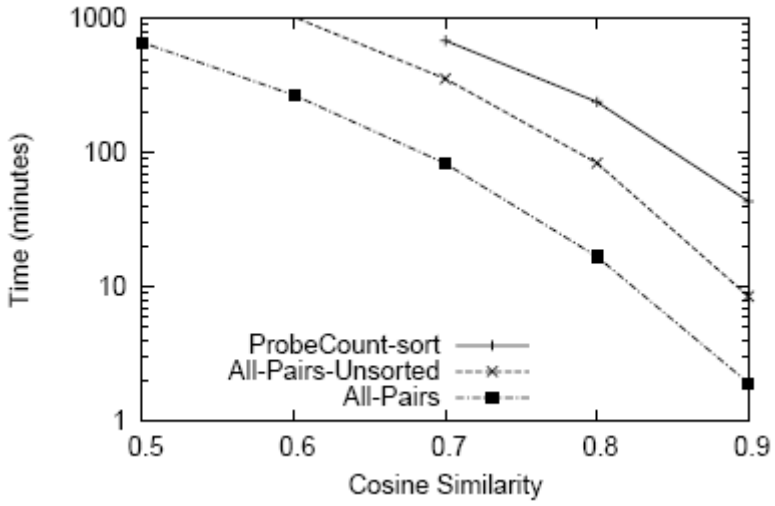
برای درستی الگوریتم بالا مرتب کردن کلمات بر اساس وزن آنها برای فیلتر کردن آنها لازم نیست، بلکه کلمات بر اساس هر رابطه ترتیب دیگری که مرتب شوند درستی الگوریتم بالا برقرار است. در واقع مرتب کردن آنها بر اساس وزن به صورت نزولی باعث می‌شود به دلیل حذف کلماتی که در متن پرکاربردتر هستند، نمایه معکوس ایجاد شده کوچکتر شود.

علاوه بر این اگر maxWeight(x) را به عنوان ماکسیمم وزن کلمات x تعریف کنیم و متن‌های مجموعه متن مورد نظر را بر اساس نزولی maxWeight مرتب کنیم، کاندیداهای هر متن به همین ترتیب انتخاب خواهند شد و باز هم کارایی الگوریتم بهبود می‌یابد چرا که برای هر متن می‌توانیم تعداد کلمات کمتری را نمایه‌سازی کنیم. با اعمال این تغییر بر الگوریتم All-pairs1، الگوریتم All-pairs2 بدست می‌آید که شبه کد آن در شکل 5 آمده است.



شکل 5 - الگوریتم All-pairs2 [36]

شکل 6 مقایسه انجام شده روی یکی از مجموعه داده‌های مورد استفاده در[36] است. در این شکل زمان اجرای الگوریتم‌های Probe-count، All-pairs1 و All-pairs2 مقایسه شده‌اند.



شکل 6 - ارزیابی الگوریتم all-pairs

2.2.2.2 الگوریتم PP-join و PP-join+

همانطور که دیدیم الگوریتم All-pairs با استفاده از نمایه معکوس ساخته شده، برای هر متن مجموعه‌ای از متن‌های کاندیدا می‌سازد و سپس با محاسبه تابع شباهت برای متن‌های این مجموعه، متن‌های با شباهت بالاتر از حد آستانه مورد نظر را شناسایی می‌کند.

در بخش قبل دیدیم All-pairs چطور با استفاده از حد آستانه t و نیز مرتب کردن متن‌های مجموعه، اندازه مجموعه متن‌های کاندیدا را کاهش داده و سرعت را بهبود داد.

در این بخش الگوریتم دیگری را بررسی می‌کنیم که ضمن فرموله کردن اصول استفاده شده در All-pairs با استفاده از روش‌های دیگری نمایه معکوس را فیلتر کرده است و به نتایج بهتری دست یافته است.

اصل فیلترینگ پیشوندی

در [37] قبل از ارائه الگوریتم PP-join و PP-join+ اصلی با عنوان اصل فیلترینگ پیشوندی ارائه شده است. این اصل در واقع پایه و مبنای الگوریتم pp-join است. از همین رو ابتدا این اصل را بیان می‌کنیم.

اصل- فیلترینگ پیشوندی[1] *: فرض کنیم* U *مجموعه جهانی از توکن‌هاست، و* O *یک ترتیب تعریف شده روی* U *است. در این صورت و مجموعه‌ای از رکوردها در اختیار داشته باشیم که توکن‌های آنها بر اساس* O *مرتب شده‌اند. فرض کنیم پیشوند-*p *رکورد* x *،* p *توکن اول* x *باشد. در اینصورت اگر* $O(x,y) > a$ *آنگاه پیشوند-* $(|x| - a + 1)$ *از رکورد* x *و پیشوند-* $(|y| - a + 1)$ *از رکورد* y *باید حداقل در یک توکن اشتراک داشته باشند.*

با استفاده از این اصل الگوریتمی با مراحل زیر ارائه شده است:

1- کافی است برای پیشوندهای متن‌ها نمایه معکوس بسازیم.[2] – فاز نمایه‌سازی

[1] Prefix-Filtering Principle : Consider An Ordering O Of The Token Universe U And A Set Of Records, Each With Tokens Sorted In The Order Of O. Let The P-Prefix Of A Record X Be The First P Tokens Of X. If $O(x,y) \geq a$, Then The $(|x| - a + 1) - prefix$ Of X And $(|y| - a + 1) - prefix$ Of Y Must Share At Least One Token [37].

[2] باید به این نکته اشاره کرد که در All-Pairs در واقع از همین اصل استفاده شده است.

2- برای هر متن که برای آن به دنبال متن‌های با شباهت بالاتر از حد آستانه هستیم با استناد به این اصل و با استفاده از نمایه معکوس ساخته شده در مرحله قبل مجموعه‌ای از متن‌های کاندیدا را به دست می‌آوریم. – فاز تولید مجموعه کاندیدا

3- شباهت متن با متون موجود در مجموعه کاندیدا را محاسبه کرده و با استفاده از مجموعه نهایی متن‌های تقریباً یکسان با متن مورد نظر را به دست می‌آوریم. – فاز بررسی شباهت

در [37] برای اندازه‌گیری شباهت متون از رابطه jaccard استفاده شده است. به عبارت دیگر دو متن تقریباً یکسان در نظر گرفته می‌شوند به شرطی که $j(x,y) \geq t$ یا $|x| \geq t.|y|$. بنابراین مساله اصلی در این روش این است که برای هر متن پیشوندی از چه اندازه را نمایه‌سازی کنیم تا مطمئن شویم برای هر متن با هر طولی الگوریتمی با مراحل بالا به نتایج درست خواهد رسید؟

در [37] نشان داده شده که بلندترین طول پیشوندی که لازم است برای هر متن مثل x نمایه‌سازی شوند تا درستی الگوریتم بالا تضمین شود $|x| - \lceil t.|x| \rceil + 1$ است.[1] و به این ترتیب برای هر متن می‌توانیم تعداد کلمات متن که نمایه سازی می‌شوند را با فاکتور 1-t کاهش دهیم.

2.2.2.3 اصل فیلترینگ مکانی

الگوریتم بالا، مشابه الگوریتم All-pairs است. [37] با ارائه یک اصل دیگر به نام فیلترینگ مکانی و ترکیب این دو، الگوریتمی به نام pp-join ارائه داده است. در این بخش پس از بررسی اصل فیلترینگ مکانی، الگوریتم pp-join را بررسی می‌کنیم.

اصل فیلترینگ مکانی[2] *: فرض کنیم* U مجموعه *جهانی از توکن‌هاست، و* O *یک ترتیب تعریف شده روی* U *است. در این صورت اگر مجموعه‌ای از رکورد ها در اختیار داشته باشیم که توکن‌های آنها بر اساس* O *مرتب شده اند. فرض کنیم توکن* w = x[i]، w *رکورد مورد نظر را به قسمت راست*

[1] اثبات این نکته در [37] آمده است.

[2] Positional Filtering Principle : Consider An Ordering O Of The Token Universe U And A Set Of Records, Each With Tokens Sorted In The Order Of O. Let Token W=X[I], W Partitions The Record Into The Left Partition $x_l(w) = x[1...(i-1)]$ And The Right Partition $x_r(w) = x[i...|x|]$. If $O(x,y) \geq a$, Then For Every Token $w \in x \cap y$, $O(x_l(w) + y_l(w)) + O(x_r(w) + y_r(w)) \geq a$.

توکن y $\mathbf{I}$ x $\in w$ *خواهیم داشت* $O(x_l(w), y_l(w)) + \min(|x_r(w)|, |y_r(w)|) \geq a$.

شکل 7 شبه کد این الگوریتم را نشان می‌دهد. در واقع با استفاده از اصل فیلترینگ مکانی نه تنها اندازه اشتراک پیشوند دو متن بلکه مکانی که در آن اشتراک دارند، در انتخاب کاندیداها تاثیر دارد. در این روش کلمات بر اساس فرکانس آنها در مجموعه به ترتیب صعودی مرتب می‌شوند. بنابراین این روش علاقه دارد کلمات با فرکانس کمتر در پیشوند متون قرار بگیرند.[1]

```
ppjoin (R, t)
  Input  : R is a multiset
           of records sorted by the increasing order of their
           sizes; each record has been canonicalized by a
           global ordering O; a Jaccard similarity threshold t
  Output : All pairs of records ⟨x, y⟩, such that sim(x, y) ≥ t
1  S ← ∅;
2  I_i ← ∅ (1 ≤ i ≤ |U|);
3  for each x ∈ R do
4      A ← empty map from record id to int;
5      p ← |x| − ⌈t · |x|⌉ + 1;
6      for i = 1 to p do
7          w ← x[i];
8          for each (y, j) ∈ I_w such
           that |y| ≥ t · |x| do     /* size filtering on |y| */
9              α ← ⌈t/(1+t) (|x| + |y|)⌉;
10             ubound ← 1 + min(|x| − i, |y| − j);
11             if A[y] + ubound ≥ α then
12                 A[y] ← A[y] + 1;
13             else
14                 A[y] ← 0;                    /* prune y */;
15         I_w ← I_w ∪ {(x, i)};
           /* index the current prefix */;
16     Verify(x, A, α);
17 return S
```

شکل 7 - الگوریتم pp-join [37]

[1] در All-Pairs کلمات به صورت نزولی مرتب می شوند و کلمات با فرکانس بالا حذف می شوند.

2.2.2.4 فیلترینگ پسوندی

با وجود اعمال فیلترینگ پسوندی رشد اندازه مجموعه کاندیدای متن نسبت به رشد کل مجموعه متون از مرتبه $O(n^2)$ است[37]. بنابراین اعمال روش‌های فیلترینگ بیشتر برای انتخاب مجموعه کاندیدا می‌تواند مفید باشد. برای این کار می‌توانیم اصل فیلترینگ مکانی را برای استفاده در پسوند متن‌ها هم بسط دهیم. مشکل این ایده این است که پسوند متن‌ها نمایه‌سازی نشده‌اند و به همین دلیل میزان همپوشانی آنها قابل محاسبه نیست. برای حل این مساله در [37] راه حل به صورت رابطه (13) ارائه شده است:

$$O(A,B) \geq a \Leftrightarrow H(A,B) \leq |A| + |B| - 2a \qquad (14)$$

در فصل 1 دیدیم که رابطه (10) بین همپوشانی و فاصله همینگ برقرار است . برای یاد آوری این رابطه را تکرار می‌کنیم:

$$J(A,B) \geq t \Leftrightarrow O(A,B) \geq a = \frac{t}{1+t}(|A| + |B|)$$

اگر پسوند x را با x_s نشان دهیم، اگر $J(A,B) \geq a$ و $|y| \leq |x|$، از آنجایی که حداکثر شباهت پیشوند متن های x و y به اندازه پیشوند متن کوتاهتر است، برای فاصله همینگ پسوند رکوردهای x و y خواهیم داشت.

$$H(x_s, y_s) < H_{\max} = 2|x| - 2\left[\frac{t}{1+t}.(|x| + |y|)\right] - (t.|x| = t.|y|) \quad (15)$$

با استفاده از این رابطه و برای بررسی برقراری حد بالای بدست آمده از رابطه بالا می‌توانیم به صورت زیر عمل کنیم: یک توکن w به دلخواه از y_s به دلخواه انتخاب می‌کنیم. این توکن y_s را به دو بخش y_l و y_r تقسیم می‌کند. y_l شامل تمام توکن‌هایی است که بنابر O از w کوچکتر هستند و y_r شامل توکن‌هایی است که از w بزرگتر هستند. به همین ترتیب x_s را هم به دو بخش x_l و x_r تقسیم می‌کنیم (حتی درصورتی که w در x_s رخ ندهد بازهم این تقسیم بندی امکانپذیر است.) از آنجایی که x_l و y_r و همچمین x_r و y_l هیچ توکن مشترکی با یکدیگر ندارند؛ می‌توان گفت :

$$H(x_s, y_s) = H(x_l, y_l) + H(x_r, y_r) \qquad (16)$$

در مورد $H(x_l, y_l)$ می‌توان گفت :

$$H(x_l, y_l) \geq |x_l| - |y_l| \quad (17)$$

و به همین ترتیب برای $H(x_r, y_r)$ حد پایین رابطه (17) وجود دارد :

$$H(x_r, y_r) \geq |x_r| - |y_r| \quad (18)$$

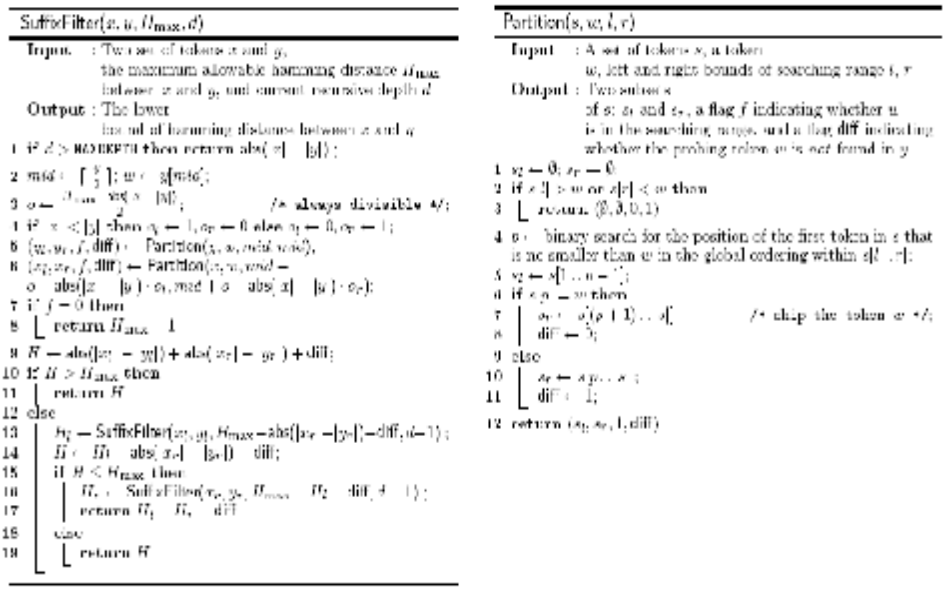
و با استفاده از روابط بالا خواهیم داشت :

$$H(x_r, y_r) \geq H(x_r, y_r) \geq abs(|x_l| - |y_l|) + abs(|x_r| - |y_r|) \quad (19)$$

با استفاده از روابط بالا می‌توانیم کاندیداهایی که حد پائین فاصله همینگ آنها بیشتر از H_{max} هستند را در مجموعه کاندیداها قرار ندهیم.

با اعمال روابط بالا به صورت بازگشتی و در چند مرحله می‌توان تعداد بیشتری از متون کاندیدا را حذف کرد. شبه کد شکل 8 مراحل بالا را به صورت بازگشتی پیاده‌سازی کرده است.

و الگوریتم شکل 9 نیز pp-join+ است که با اضافه کردن فیلترینگ پسوندی به pp-join به دست آمده است. نمودارهای شکل 10 نتایج pp-join ، pp-join+ و All-pairs را روی 3 مجموعه داده با هم مقایسه می‌کنند.



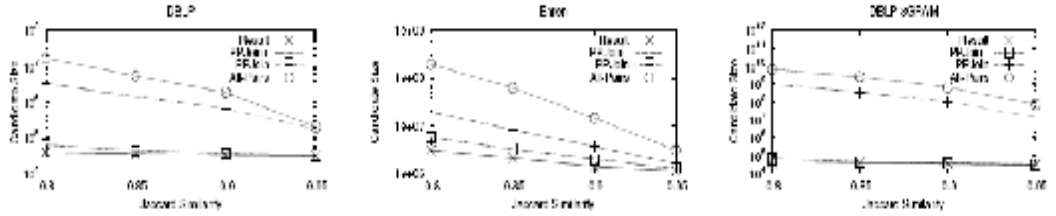
شکل 8 - پیاده سازی روش فیلترینگ پسوندی [37]

ppjoin+(R, t)

Input : R is a multiset of records sorted by the increasing order of their sizes; each record has been canonicalized by a global ordering $\mathcal{O}$; a Jaccard similarity threshold t
Output : All pairs of records $\langle x, y \rangle$, such that $sim(x, y) \geq t$

```
1  S ← ∅;
2  I_i ← ∅ (1 ≤ i ≤ |U|);
3  for each x ∈ R do
4      A ← empty map from record id to int;
5      p ← |x| − ⌈t · |x|⌉ + 1;
6      for i = 1 to p do
7          w ← x[i];
8          for each (y, j) ∈ I_w such
           that |y| ≥ t · |x| do      /* size filtering on |y| */
9              α ← ⌈(t/(1+t))(|x| + |y|)⌉;
10             ubound ← 1 + min(|x| − i, |y| − j);
11             if A[y] + ubound ≥ α then
12                 if A[y] = 0 then
13                     H_max ← |x| + |y| − 2 · ⌈(t/(1+t)) · (|x| + |y|)⌉ − (i + j − 2);
14                     H ← SuffixFilter(x[(i + 1)..|x|], y[(j + 1)..|y|], H_max, 1);
15                     if H ≤ H_max then
16                         A[y] ← A[y] + 1;
17                     else
18                         A[y] ← −∞;      /* avoid considering y again */;
19             else
20                 A[y] ← 0;                    /* prune y */;
21         I_w ← I_w ∪ {(x, i)};
           /* index the current prefix */;
22     Verify(x, A, α);
23 return S
```

شکل 9 - الگوریتم pp-join+ [37]



(a) Jaccard, DBLP, Candidate Size (b) Jaccard, Enron, Candidate Size (c) Jaccard, DBLP-sGRAM, Candidate Size

شکل 10 - بررسی کارایی +ppjoin و ppjoin [37]

در پایان این بخش ذکر این نکته لازم است که الگوریتم‌های Probe، All-pairs و PP-join که در این فصل بررسی شد، برای یافتن متن‌هایی که شباهت آنها از حد آستانه تعیین شده با استفاده از تابع شباهت خاصی ارائه شده اند. Probe از تابع همپوشانی، All-pairs از رابطه کسینوسی و pp-join از رابطه Jaccard برای محاسبه شباهت استفاده کرده‌اند. با این حال با توجه به روابط حاکم بر توابع شباهت که در فصل اول بررسی شد، این روشها بر روی دیگر روابط هم قابل اعمال هستند و در انتهای [35,36,37] به این مساله پرداخته شده است.

2.2.2.5 الگوریتم word-Goups

در ابتدای این فصل اشاره شده که تمام روش‌های دقیق از نمایه استفاده نمی‌کنند. در این بخش شرح مختصری از یکی از این روشها آورده شده است.این الگوریتم با الهام از الگوریت‌های جستجوی مجموعه اقلام مکرر[1] در داده کاوی سعی در یافتن متن‌هایی دارد که مجموع وزن کلمات مشترک مابین آنها از حد آستانه t بزرگتر است. در این روش کلمات اقلام داده‌ای و متن‌ها تراکنش‌ها هستند. حداقل حد پشتیبانی[2] دو و حداکثر وزن مجموعه اقلام t در نظر گرفته می‌شود. با این فرضیات می‌توان از الگوریتم-های جستجوی مجموعه اقلام مکرر از جمله Apriori و FP-Growth برای یافتن مجموعه اقلامی که وزن کل آنها از t بیشتر است استفاده کرد.

[1] Frequent Item-Set Mining Algorithms

[2] Minimum Support

2.3روشهای تقریبی

در این فصل روش‌هایی که تحت عنوان کلی روش‌های تقریبی قرار می‌گیرند، بررسی‌می‌شوند. همانطور که قبلاً اشاره شد روش‌های تقریبی در مقابل روش‌های دقیق قرار می‌گیرند. این روش‌ها برای کاهش زمان جستجو و بالا بردن سرعت خود با استفاده از تکنیک‌هایی که در این بخش بررسی خواهند شد میزان شباهت دو متن را تخمین می‌زنند. و با استفاده از مقدار تقریبی محاسبه شده، در مورد تقریباً یکسان بودن دو متن قضاوت می‌کنند. این روشها نسبت به روش‌های دقیق سریعتر هستند اما به دلیل استفاده از مقدار تقریبی شباهت متون، نرخ خطای مثبت[1] در آنها بالاتر است. وجه اشتراک این روش‌ها استفاده از اثرانگشت[2] متن به جای استفاده از خود آن در شناسایی تقریباً یکسان‌هاست. ایده اصلی الگوریتم‌هایی که از تولید اثرانگشت استفاده می‌کنند این است که با استفاده از روش‌های مناسب کاری کنیم که به متن‌هایی که تقریباً یکسان هستند، اثرانگشت‌های یکسانی اختصاص داده شود.

در این فصل ابتدا الگوریتم I-match را بررسی می‌کنیم، این الگوریتم پس از پیش‌پردازش متن و فیلتر کردن برخی از اجزای متن چنانکه خواهیم دید، از هر متن اثر انگشتی ایجاد می‌کند و در ادامه با استفاده از آن، متن‌های تقریباً یکسان را شناسایی می‌کند.

در ادامه روش‌های Shingling و LSH[3] را که از مهمترین الگوریتم‌های شناسایی متون تقریباً یکسان هستند را بررسی خواهیم کرد. و در انتها یک الگوریتم فازی جهت این منظور را بررسی می‌کنیم.

2.3.1الگویتم I-Match

این روش در [26] و توسط Chowdhury و همکارانش ارائه شده است. این الگوریتم برای یافتن صفحات وب مشابه ارائه شده است.

در این روش پس از انجام پیش‌پردازش‌هایی مثل حذف کلمات با تعداد کاراکترهای کمتر از متوسط تعداد کاراکترهای هر کلمه در مجموعه متون، حذف کلمات با کاراکترهای با بیش از 25 کاراکتر و ...،

[1] Positive False
[2] Fingerprint
[3] Locality Sensitive Hashing

فرکانس تکرار کلمات باقی‌مانده در متن محاسبه می‌شود، با استفاده از فرکانس محاسبه شده برخی از کلمات فیلتر می‌شوند و کلمات باقی‌مانده به عنوان نماینده متن مرتب می‌شوند. مرتب کردن کلمات باقی‌مانده می‌تواند اثر جابجایی پاراگراف‌ها و جملات را حذف کند. پس از مرتب کردن کلمات باقی‌مانده، متون با استفاده از الگوریتم درهم‌سازی[1] SHA1 به یک عدد 160 بیتی درهم‌سازی می‌شوند و از این عدد برای مقایسه با دیگر متون برای یافتن متون تقریباً یکسان استفاده می‌شود.

در این روش دو متن در صورتی با هم مشابه یا تقریباً یکسان خواهند بود که مقدار درهم‌سازی بدست آمده برای آنها با هم یکسان باشند. به عبارت دیگر دو متن در صورتی تقریباً یکسان هستند که پس از عملیات پیش‌پردازش و فیلتر کردن کلمات بر اساس فرکانس تکرارشان مجموعه کلمات کاملاً یکسانی در آنها باقی مانده باشد.

2.3.2 روش shingling

Andrei Z.Broder در [22]، به همراه Steven Glassman، Mark Manasse و Geoffrey Zweig روشی موسوم به Shingling را مطرح کردند. Broder پس از آن و در [23]، ضمن شرح بیشتر روش استفاده شده در [22]، آن را با استفاده از روابط ریاضی شرح داده است.

این روش دو ویژگی اصلی دارد. اول اینکه در روش Shingling، مساله اندازه‌گیری شباهت بین دو متن به مساله اشتراک دو مجموعه تبدیل شده است. و دوم اینکه برای تخمین اشتراک 2 مجموعه از نمونه-برداری تصادفی[2] از هر متن استفاده شده است. این نمونه‌برداری برای هر متن می‌تواند به صورت مستقل از بقیه متون صورت بگیرد؛ برخلاف روشی مثل I-Match که در نمونه‌برداری از متن‌ها و انتخاب نماینده هر متن به دیگر متون و کل مجموعه وابسته است. به علاوه در این مقاله نشان داده شده است که چه طور با داشتن نمونه‌ای با اندازه یکسان و ثابت برای تمام متن‌ها می‌توان میزان شباهت آنها را اندازه‌گیری کرد.

در [30] نیز Broder روش خود را بسط داده و ایده Super Shingling را که در [22] به اختصار به آن اشاره شده بود را به طور دقیق تر مطرح کرده است.

---

[1] Hashing Algorithms

[2] Random Sampling

Broder در [23]، ضمن اشاره به این نکته که هیچ یک از معیارهایی که برای اندازه‌گیری فاصله در رشته‌ها مطرح هستند، مثل فاصله همینگ و فاصله ویرایش ، نمی‌توانند، تقریباً یکسان[1] بودن را به خوبی نشان دهند؛ مفهوم همانندی[2] یا شباهت را به صورت زیر تعریف می‌کند:

همانندی دو متن A و B، r(A,B) ، عددی مابین 0 و 1 است که هر چه مقدار آن به 1 نزدیکتر باشد، احتمال اینکه دو متن A و B یکسان باشند بیشتر است.[3]

به طور مشابه مفهوم دربرداشتن یا شمول در این مقاله به صورت زیر تعریف شده است:

شمول متن A در B[4]، c(A,B) ، عددی مابین 0 و 1 است که هر چه مقدار آن به 1 نزدیکتر باشد نشاندهنده این است که B تقریباً شامل A است.[5]

هر متن در این روش به عنوان توالی از توکن‌ها در نظر گرفته می‌شود. هر توکن می‌تواند یک کاراکتر، کلمه، جمله یا خط از متن باشد. در انتخاب توکن یا همان واحدهای متن محدودیتی وجود ندارد به جز اینکه مجموعه توکن‌های متن باید قابل شمارش باشند.

قبل از هر گونه عملیاتی، فرض بر این است که هر متن با استفاده از یک برنامه پارسر[6] یا تجزیه کننده به یک فرم متعارف[7] تبدیل شده است. منظور از فرم متعارف فرمی است که در آن اطلاعات اضافی که ممکن است مورد توجه ما نباشد، مثل تگ[8] های HTML یا نقطه گذاری، حذف شده‌اند و پس از این هر متن به معنای توالی توکن‌های آن خواهد بود.

---

[1] Roughly The Same

[2] Resemblance

[3] The Resemblance R(A,B) Of Two Documents, A And B, Is A Number Between 0 And 1, Such That When The Resemblance Is Close To 1 It Is Likely That The Documents Are Roughly The Same [23].

[4] A Is Roughly Contained Within B

[5] The Containment C(A,B) Of A In B Is A Number Between 0 And 1 That, When Close To 1, Indicates That A Is Roughly Contained Within B [23].

[6] Parser

[7] Canonical Form

[8] Tag

هر زیر توالی پیوسته از متن را یک Shingle نامیده می‌شود[1]. هر متن پس از تبدیل به فرم متعارف مورد نظر تبدیل به مجموعه[2] ای از shingle ها می‌شود.

به عنوان مثال متن زیر را که با حذف جزئیات و اطلاعات غیر ضروری به فرم متعارف تبدیل شده است در نظر می‌گیریم:

(a,rose,is,a,rose,is,a,rose)

تمام shingle های با اندازه 4 در این متن عبارند از :

{ (a,rose,is,a), (rose,is,a,rose), (is,a,rose,is), (a,rose,is,a), (rose,is,a,rose)}

که باحذف موارد تکراری تبدیل به مجموعه زیر می‌شود:

{ (a,rose,is,a), (rose,is,a,rose), (is,a,rose,is),}

اگر D، یک متن باشد مجموعه تمام shingle های با اندازه w آن را با S(D,w) نمایش می‌دهیم. پس از اینکه اندازه مناسب برای shingle ها،w، را انتخاب کردیم. همانندی بین 2 متن A و B را به صورت رابطه (19) اندازه‌گیری می‌کنیم.

$$r_w(A,B)=\frac{|S(A,w)\ \mathbf{I}\ S(B,w)|}{|S(A,w)\mathbf{U}\ S(B,w)|} \qquad (20)$$

همانطور که می‌بینید این رابطه همان رابطه (2) یا همان ضریب Jaccard است. با این تعریف خواهیم داشت $0\le r_w \le 1$ ، $r_w(A,A)=1$، یعنی تحت این تعریف شباهت یک متن به خودش برابر با 1 است یا به عبارتی هر متن کاملا با خودش مشابه است.

و به طور مشابه شمول را نیز به صورت رابطه (20) تعریف می‌کنیم.

[1] A Contiguous Subsequence Contained In A Document Is Called A Shingle [22].

[2] با توجه به امکان تکرار شدن Shingle ها در یک متن در واقع هر متن تبدیل به یک Bag یا Multiset از Shingle ها خواهد شد. در [23] به این مساله اشاره شده است و 2 راهکار برای ادامه کار در نظر گرفته شده است، یکی اینکه تکرار Shingle ها را نادیده بگیریم و دیگر اینکه همراه با هر Shingle تعداد تکرار های آن در متن را هم در نظر بگیریم. که نظر به اینکه در ادامه و در پیاده سازی های انجام شده تکرار ها در نظر گرفته نشده اند، ما نیز مساله را با استفاده از مجموعه ها بیان می کنیم.

$$c_w(A,B)=\frac{|S(A,w)\ \mathbf{I}\ S(B,w)|}{|S(A,w)|} \qquad (21)$$

در خصوص رابطه‌های (19) و (20) باید به چند نکته اشاره کنیم. اول اینکه این دو رابطه هر دو نسبت به w حساس هستند، یعنی با shingle های با اندازه‌های مختلف میزان شباهت به دست آمده متفاوت خواهد بود، به عنوان مثال اگر 2 متن زیر را در نظر می‌گیریم:

A = (a,rose,is,a,rose,is,a,rose) و

B = (a,rose,is,a,flower,which,is,a,rose)

با w=1، A و B 60% ، با w=2 50% و با w=3 42.85% همانندی دارند.

دومین نکته اینکه، اگر همانندی دو متن مثل A و B با استفاده از رابطه (19) و w=1 برابر با 100% باشد، به این معنا نیست که این دو متن کاملا یکسان هستند؛ بلکه به این معناست که متن‌های A و B، ترکیب[1] یکدیگر هستند. حتی برای w≥2، هم اگر همانندی 100% باشد به طور قطع نمی‌توان مطمئن بود که A و B کاملاً یکسان هستند بلکه در این حالت هم متن‌های A و B، می‌توانند ترکیبی از یکدیگر باشند؛ البته در این حالت تعداد ترکیب‌های محتمل و احتمال اینکه دو متن A و B یکسان نباشند، کمتر است.

در این کار انواع روش‌های فیلتر کردن کلمات آزمایش شده‌اند و در نهایت حذف کلماتی که شاخص idf[2] آنها در مجموعه از 0.1 کمتر است به عنوان بهترین روش برای فیلتر کردن کلمات انتخاب شده‌اند.

برای پیدا کردن کلاسترهای تقریباً یکسان از متون، پس از محاسبه مقدار تابع درهم‌ساز برای هر متن، مقادیر <docId , hashValue> به دست آمده در یک درخت درج می‌شوند. درج <docId , hashValue> با hashValue های مساوی در برگهای درخت نشاندهنده تقریباً یکسان بودن متن‌ها است. به این ترتیب هر برگ درخت یک کلاستر از متون تقریباً یکسان خواهد بود.

[1] Permutation

[2] Inverse Document Frequency

2.3.2.1 تخمین همانندی با استفاده از نمونه‌برداری از متن

فرض کنیم Ωمجموعه تمام shingle های با اندازه w باشد. بدون اینکه به کلیت مطالب خدشه‌ای وارد کنیم می‌توانیم فرض کنیم مجموعه Ω ، یک مجموعه مرتب یا ترتیب دار[1] است. اگر پارامتر s را که یک عدد صحیح مثبت است مفروض بدانیم؛ برای $W \subseteq \Omega$ مجموعه $MIN_s(W)$را به صورت رابطه (21) تعریف می‌کنیم :

$$MIN_s(W) = \begin{cases} the\,set\,of\ smallest\,s\,elements\,in\,W, & if\ |W| \geq s \\ W \quad , & otherwise \end{cases} \quad (22)$$

که در این تعریف "smalleset" اشاره به ترتیبی دارد که روی Ωتعریف شده است.

اگر $I \subseteq N$، مجموعه $MOD_m(I)$را به صورت رابطه (22) تعریف می‌کنیم.

$$MOD_m(I) = the\,set\,of\ elements\,of\,W\,that\ are. \bmod m \quad (23)$$

قضیه : اگر داشته باشیم $g:\Omega \to N$ و $p:\Omega \to \Omega$ ک ترکیب از Ω باشد که به صورت رندم و یکنواخت انتخاب شده است.[2] اگر

---

[1] Ordered Set

[2] برای وضوح بیشتر تعریف این قضیه را به زبان اصلی تکرار می کنیم:

Theorem : Let $g:\Omega \to N$ Be An Arbitrary Injection, Let $p:\Omega \to \Omega$ E A Permutation Of Ω Chosen Uniformly At Random And Let $M(A) = MIN_s(p(S(A,w)))$ And $L(A) = MOD_m(g(p(S(A,w))))$. Define M(B) And L(B) Analogously. The Value $\frac{|MIN_s(M(A)\mathbf{U}M(B))\,\mathbf{I}\,M(A)\,\mathbf{I}\,M(B)|}{|MIN_s(M(A)\mathbf{U}M(B))|}$ Is An Unbiased Estimate Of The Resemblance Of A And B.

The Value $\frac{|L(A)\,\mathbf{I}\,L(B)|}{|L(A)\mathbf{U}\,L(B)|}$ Is An Unbiased Estimate Of The Resemblance Of A And B.

The Value $\frac{|L(A)\,\mathbf{I}\,L(B)|}{|L(A)|}$ Is An Unbiased Estimate Of The Containment Of A In B.

(24)
$$M(A) = MIN_s(p(S(A,w)))$$

و

$$L(A) = MOD_m(g(p(S(A,w)))) \qquad (25)$$

و M(B) و L(B) را هم به طریق مشابه تعریف کنیم در این صورت مقدار

$$\frac{|MIN_s(M(A)\mathbf{U}M(B))\ \mathbf{I}\ M(A)\ \mathbf{I}\ M(B)|}{|MIN_s(M(A)\mathbf{U}M(B))|} \qquad (26)$$

یک تخمین بدون بایاس[1] از همانندی A و B است. و مقدار

$$\frac{|L(A)\ \mathbf{I}\ L(B)|}{|L(A)\mathbf{U}L(B)|} \qquad (27)$$

هم یک تخمین بدون بایاس از همانندی A و B است. به علاوه مقدار رابطه (27) هم یک تخمین بدون بایاس از شمول A در B است:

$$\frac{|L(A)\ \mathbf{I}\ L(B)|}{|L(A)|} \qquad (28)$$

اثبات این روابط در [23] به تفصیل آمده است. ما در اینجا از آوردن اثبات این روابط چشم‌پوشی می‌کنیم و کاربرد آنها را در تشخیص متون تقریباً یکسان بررسی می‌کنیم.

با استفاده از این قضیه و روابط به دست آمده نتیجه می‌گیریم که اگر برای هر دو متن،D و D'، یکی از مقادیر M(D) و D'M((یا L(D) و)D'L(را که در روابط 4-5 و 4-5 معرفی شده‌اند را بدست بیاوریم می‌توانیم با استفاده از رابطه (25) یا رابطه (26) می‌توانیم میزان همانندی D و D' را محاسبه کنیم.

مزیت M(D) و L(D) این است که در محاسبه همانندی می‌توانیم از M(D) و یا L(D) که هر دو نمونه‌ای از متن هستند استفاده کنیم. یا به عبارت دیگر به جای استفاده از تمام متن برای محاسبه همانندی

[1] Unbiased Estimate

می‌توانیم برای هر متن تنها یک شِما یا طرح کلی[1] ذخیره و از آن برای محاسبه همانندی استفاده کنیم. استفاده از M(D) و L(D)، فضا و زمان لازم در محاسبه همانندی را کاهش می‌دهد.

M(D)، در واقع پس از اعمال یک ترکیب روی S(D,w) ، از s عنصر ابتدایی آن به عنوان طرح کلی متن استفاده می‌کند. اگر از M(D) به عنوان طرح کلی متن استفاده کنیم، نمونه یا طرح کلی به دست آمده برای متون هم اندازه خواهند بود و این یک مزیت است، اما توسط M(D)، شمول را نمی‌توانیم تخمین بزنیم.

L(D)، پس از نگاشت shingle های بدست آمده به اعداد طبیعی، اعدادی را که بر عددی مشخص مثل m بخش پذیر هستند به عنوان طرح کلی متن استفاده می‌کند. اندازه L(D)، بدست آمده برای هر متن با طول متن متناسب است و با استفاده از آن می‌توان همانندی و شمول را تخمین زد.

اندازه مجموعه shingle های بدست آمده برای هر متن نسبتاً بزرگ است. به عنوان مثال اگر w=7،یعنی هر shingle متشکل از 7 توکن باشد، و کلمات را به عنوان توکن های متن در نظر بگیریم، هر shingle به طور متوسط بین 40 تا 50 بایت خواهد بود. برای کاهش فضا به هر shingle یک شاخص،id، که طول آن l بیت است را اختصاص می‌دهیم. و سپس روی یک ترکیب π مجموعه $\{0, 1, 2, \ldots, 2^l\}$ اعمال می‌کنیم.

انتخاب l باید به دقت صورت بگیرد، اگر l به حد کافی بزرگ نباشد، ممکن است دو shingle متفاوت به شاخص یکسانی نگاشته شود و به اصطلاح برخورد[2] به وجود بیاید. انتخاب l بزرگ مشکل برخورد را از بین می‌برد اما فضای لازم برای ذخیره را افزایش می‌دهد.

فرض کنیم $f : p \to \{0,1,2,\ldots,2^l\}$ ، تابعی باشد که نگاشت مورد بحث در بالا را انجام دهد. در این صورت خواهیم داشت.

$$g_{w,f}(A,B) = \frac{|f(S(A,w)) \mathbf{I}\ f(S(B,w))|}{|f(S(A,w)) \mathbf{U}\ f(S(B,w))|} \qquad (29)$$

در [23] نشان داده شده است که

[1] Sketch

[2] Collision

$$\left|g_{w,f}(A,B)-g_{w}(A,B)\right|<\frac{\left|S(A,w)\mathbf{U}\,S(B,w)\right|}{2^{l-11}} \qquad (30)$$

در [22,23,20] برای f از روش Rabin's Fingerprinting استفاده شده است و دلیل این مساله هم سرعت و کارایی این روش است.

Broder و همکارانش در [22] این روش را روی مجموعه‌ای از صفحات وب که در یک بار مرور وب توسط موتور جستجوی Alta Vista[1] بدست آمده بود و شامل بیش از سی میلیون صفحه بود، اعمال کرده و در نهایت 3.6 میلیون کلاستر شامل 12.3 میلیون صفحه بدست آوردند. به عبارت دیگر در این مجموعه بیش از 30 درصد صفحات در رابطه تقریباً یکسانی با صفحات دیگر قرار گرفتند.

2.3.2.2 کلاسترینگ نحوی مجموعه متون

در پیاده سازی این روش در [22]، برای کلاستر کردن مجموع وب، از الگوریتم چهار مرحله ای زیر استفاده شده است:

1- مجموعه صفحات وب توسط موتور جستجوی Alta Vista جمع آوری شد.
2- طرح کلی[2] هر صفحه محاسبه می‌شود.
3- طرح کلی هر جفت از صفحه‌ها را با هم مقایسه کرده و اگر شباهت مابین آنها از حد آستانه‌ای مشخص بیشتر بود، به عنوان دو متن یا صفحه تقریباً یکسان شناسایی می‌شوند.
4- از ترکیب و شناسایی صفحاتی که به عنوان تقریباً یکسان شناسایی شده‌اند، کلاسترهایی از صفحات تقریباً یکسان تشکیل می‌شود.

به دلیل حجیم بودن مجموعه داده مورد نظر و عدم امکان انجام تمام محاسبات لازم در حافظه، دسترسی به I/O در این الگوریتم نقش مهمی دارد و در پیاده‌سازی باید تا حد امکان این مساله درنظر گرفته شود.

[1] Http://Www.Altavista.Com

[2] Sketch

در [22,23] از الگوریتم چهار مرحله‌ای بالا استفاده شده است و برای غلبه بر حجیم بودن داده‌ها از روش تقسیم و غلبه[1] استفاده شده است و زمان کلی ایجاد کلاسترها به O(n Log(n)) کاهش یافته است.

2.3.2.3 بهبود سرعت روش Shingling

علی رغم اینکه در الگوریتم Shingling نمونه برداری از متن تاحد زیادی در سرعت و کارایی الگوریتم تاثیر مثبت دارد؛ با این حال این الگوریتم همچنان نسبتاً کند است. در [22] برای مقابله با این مساله از چند روش متداول استفاده شده است. یکی از این روش‌ها حذف Shingle هایی است که در پرکاربرد هستند و در تعداد زیادی از متون تکرار شده اند. این کار علاوه بر بالا بردن سرعت الگوریتم در افزاریس دقت الگوریتم نیز موثر بوده است، چرا که در این مقاله از مجموعه صفحات وب به عنوان مجموعه داده استفاده شده است و وجود متن‌ها و محتوای مشترک بین صفحات یک سایت از جمله مشکلاتی است که الگوریتم‌های شناسایی متون تقریباً یکسان با آن مواجه هستند. حذف Shingle های پرکاربرد (در این مقاله Shingle هایی که بیش از 1000 بار در کل مجموعه آمده اند، کنار گذاشته شده اند.) می‌تواند تا حدی چنین بخش‌هایی از متن را حذف کند.

2.3.2.4 Super Shingles یا ویژگی‌های سند

در [23]، ایده Super Shingle ها به صورت اجمالی مطرح شده است و سپس در [30] ضمن بیان تازه‌ای از روش Shingling، Super Shingle ها به طور جزئی‌تری بررسی شده اند.

پس از استخراج shingle ها از متن و بدست آوردن درهم‌سازی و نمونه برداری از آنها ، توالی shingle های باقی‌مانده از هر متن نماینده آن متن خواهد بود.

با انجام دوباره shingling روی توالی shingle های متن، یا به عبارتی shingle کردن دوباره shingle ها به ابرطرح[1] متن یا طرح از طرح[2] متن دست می‌یابیم. به shingle هایی که به این ترتیب به آنها دست می‌یابیم به اصطلاح super shingle گفته می‌شود.

[1] Divide And Conquer

اگر دو متن یک super shingle مشترک داشته باشند به این معناست که طرح های آنها دارای یک توالی از shingle های مشترک هستند. به این ترتیب با وجود یک یا بیشتر super shingle مشترک بین دو متن، این احتمال که این دو متن تقریباً یکسان باشند بسیار بالاست.

بنابراین در مقایسه متن‌ها با استفاده از super shingleها کافی است تنها وجود یک یا بیشتر supershingle مشترک را بررسی کنیم و این می‌تواند به کارایی الگوریتم کمک زیادی کند.

مشکل اصلی، استفاده از super shingle ها در یافتن متن‌های تقریباً یکسان را می‌توان در این دانست که super shingle ها به اندازه shingle ها دقیق و انعطاف پذیر نیستند. به علاوه متن‌های کوتاه تعداد super shingle های کمتری دارند و استفاده از super shingle ها درصد خطا را در آنها بالا می‌برد. به علاوه از super shingle ها نمی‌توان در محاسبه شمول استفاده کرد.

Super shingling نسبت به shingling بسیار کاراتر است و دلیل آن هم استفاده از Super Shingle ها به جای Shingle ها است. هر دو روشهای shingling و super shingling نسبت به اندازه shingle ها حساس هستند. طوری که با یک تنظیم خاص ممکن است super shingling از shingling محدودتر باشد. در این بخش روش super shingling را به بررسی خواهیم کرد.

2.3.2.5 Min-wise Independent Finger prints

در [30]، Broder پس از ارائه بیانی تازه از shingling، Super Shingling را که در [22] تنها به عنوان یک ایده مطرح شده بود را به روش دقیق تر ارائه کرده است. که در این بخش سعی می‌کنیم به اختصار این روش‌ها را بررسی کنیم.در واقع در [30]، Broder نشان داده است که چه طور با استفاده از sketch هایی کمتر از 50 بایت، متن‌های تقریباً یکسان را می‌توان تشخیص داد.

همانطور که در بخش‌های قبلی دیدم، تولید اثرانگشت یک جزء اصلی در روش shingling است، در بیان تازه shingling، ایجاد pseudo-random permutation روی یک مجموعه بزرگ نیز به عنوان جزء جدیدی از این روش معرفی شده است. در واقع در این بیان تازه با انجام عملیات بیشتر روی متن، طرح کلی کوتاه‌تری از آن ایجاد می‌شود.

[1] Meta-Sketch
[2] Sketch Of Sketch

اگر از همان تعریف بالا استفاده کنیم، در [30]، نشان داده شده است که

$$\Pr(\min\{p(S(A,w))\}=\min\{p(S(B,w))\})=\frac{|S(A,w)\ \mathbf{I}\ S(B,w)|}{|S(A,w)\mathbf{U}\ S(B,w)|}=g_w(A,B)$$

با توجه به اینکه w در طرفین تساوی بالا ثابت است، برای ساده شدن، می‌توانیم w را به طور ضمنی از روابط حذف کنیم، به این ترتیب رابطه بالا به شکل رابطه (30) در می‌آید:

$$\Pr(\min\{p(S_A)\}=\min\{p(S_B)\})=\frac{|S_A\ \mathbf{I}\ S_B|}{|S_A\ \mathbf{U}\ S_B|} \quad (31)$$

اثبات این رابطه در [30] آمده است و ما در اینجا از آوردن آن صرف نظر می‌کنیم. بر اساس این رابطه احتمال اینکه کوچکترین عضو دو مجموعه S_A و S_B ، پس از اعمال ترکیب π با یکدیگر برابر باشند برابر است با میزان همانندی این دو مجموعه.

بنابراین می‌توانیم از رابطه (30) استفاده کنیم و با در نظر گرفتن t عدد ترکیب $p_1,p_2,...,p_t$، برای هر متن با یک طرح کلی به صورت رابطه (31) تشکیل دهیم:

$$\overline{S}_A=(\min\{p_1(S_A)\},\min\{p_2(S_A)\},...,\min\{p_t(S_A)\}) \quad (32)$$

که در آن S_A مجموعه shingle های متن A یا همان $S(A,w)$ است. t به عنوان مثال می‌تواند 100 باشد.

اگر از رابطه (31) و بردار معرفی شده در آن به عنوان طرح کلی هر متن استفاده کنیم، برای محاسبه شباهت می‌توانیم، از تعداد عناصر مشترک بین بردار دو متن به عنوان معیاری برای شباهت آن دو استفاده کنیم.

در مورد شباهت این روش با روش قبلی shingling باید گفت در روش قبلی از s عنصر کوچکتر یک ترکیب به عنوان بردار متن استفاده می‌شد، رابطه (21) ، اما در اینجا از کوچکترین عنصر t ترکیب متفاوت استفاده می‌شود.

ترکیب‌هایی که برای تشکیل بردار متن استفاده می‌شوند، چنانکه قبلاً نیز گفته شد باید به صورت رندم و یکنواخت از بین ترکیب‌های ممکن روی مجموعه shingle ها انتخاب شوند. این ویژگی را می‌توانیم به این صورت بیان کنیم:

اگر $[n]$را به صورت رابطه (32) تعریف کنیم،

$$[n] \stackrel{def}{=} \{0,1,2,\ldots,n-1\} \qquad (33)$$

و $X \subseteq [n]$ و x هر عضو دلخواهی از X باشد؛ در این صورت باید رابطه (33) برای ترکیب مورد نظر، p، برقرار باشد:

$$pr(\min\{p(X)\} = p(x)) = \frac{1}{|X|} \qquad (34)$$

به عبارت دیگر باید ترکیب مورد طوری انتخاب شود که شانس هر عضو برای اینکه کوچکترین عضو آن باشد، با تمام دیگر اعضا برابر باشد.

در [30] ضمن اشاره به نکته که عملاً انتخاب ترکیب‌هایی که دارای این خاصیت باشند روی مجموعه-های بزرگ امکان ندارد، مفهوم min-wise independent permutation معرفی شده است و از آنها برای محاسبه بردار متن استفاده شده است.

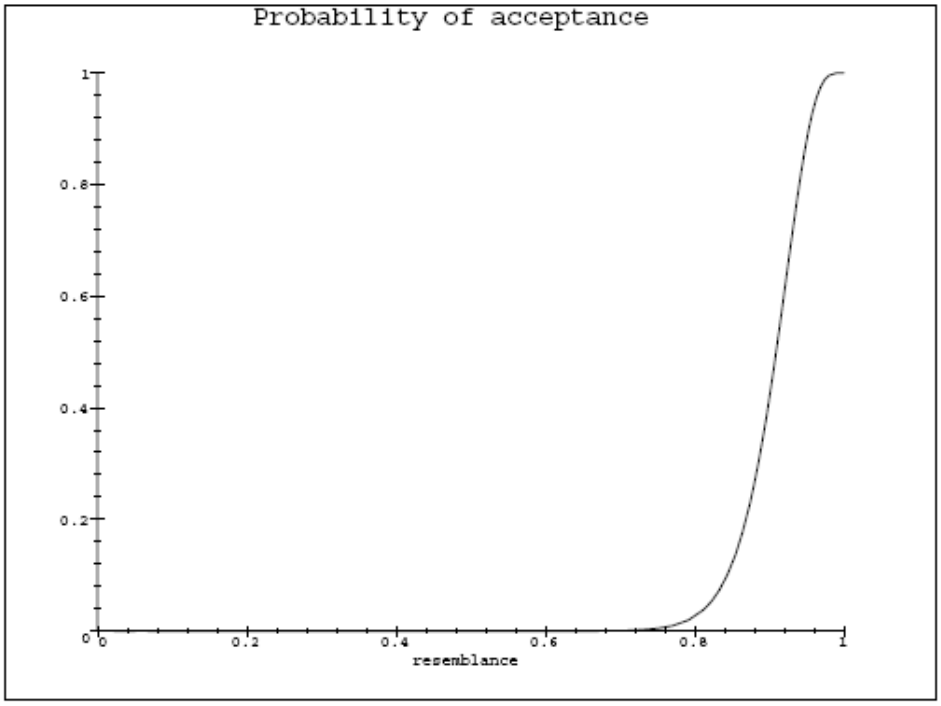
پس از بدست آوردن بردارهای متن می‌توانیم از الگوریتم چهار مرحله‌ای ارائه شده برای ایجاد کلاسترهای متن‌های تقریباً یکسان استفاده کنیم.

اگر همانندی دو متن A و B، با استفاده از رابطه (21)، r اندازه گیری شده باشد، در صورتی که r به یک نزدیک باشد، آنگاه بیشتر عناصر $\overline{S}_A$ و $\overline{S}_B$ با هم یکسان خواهند بود.

اگر بردارهای $\overline{S}_A$ و $\overline{S}_B$ را به k بخش که هر یک دارای s عنصر هستند تقسیم کنیم. در این صورت احتمال اینکه هر اینکه تمام عناصر یک بخش برای هر دو متن یکسان باشند r^s خواهد بود و احتمال اینکه تعداد r یا بیشتر از این بخش ها بین دو بردار کاملا یکسان باشند با رابطه (34) قابل محاسبه است:

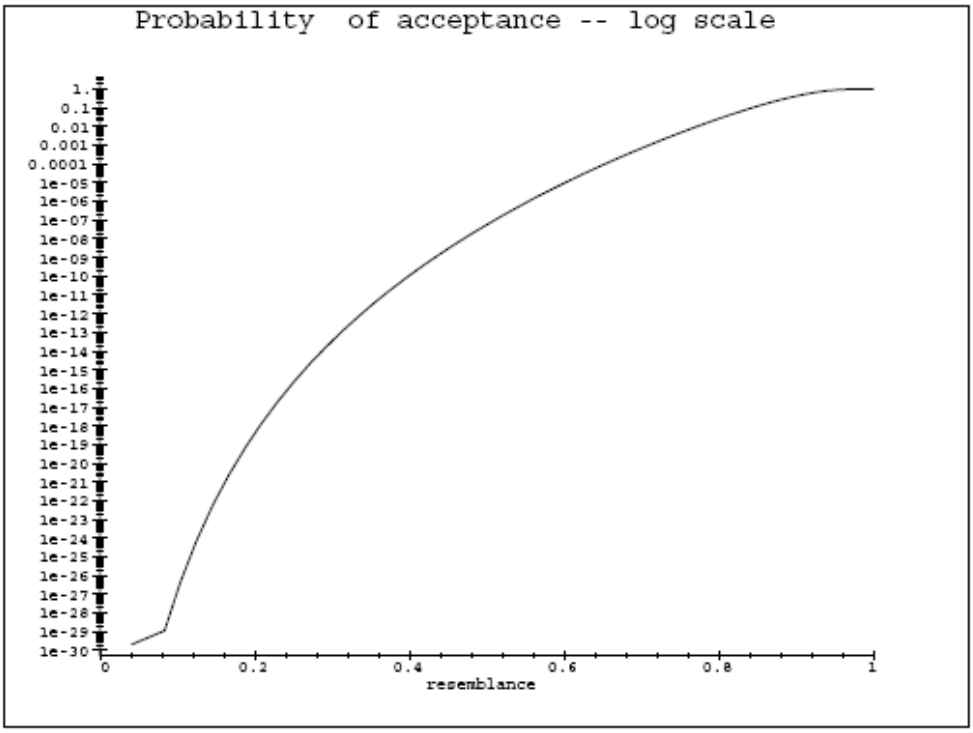
$$pr_{k,s,r} = \sum_{r \le i \le k} \binom{k}{i} r^{s,i} \left(1 - r^s\right)^{k-i} \qquad (35)$$

در [30] نشان داده شده است که اگر [k , s, r] درست انتخاب شوند، چندجمله ای $pr_{k,s,r}$ مثل یک فیلتر بالاگذر مناسب[1] عمل خواهد کرد. به عنوان مثال شکل های 11 و 12 که از [30] آورده شده اند، $pr_{6,14,2}$ را در مقیاس معمولی و لگاریتمی نشان می‌دهند.



شکل 11 - $pr_{6,14,2}$ **در مقیاس معمولی** –[30]

[1] Sharp



شکل 12 - $pr_{6,14,2}$ در مقیاس لگاریتمی – [30]

با استفاده از این نکته می‌توانیم بردارهای بدست آمده را بازهم فشرده‌تر کنیم.

برای این کار، اگر D متنی باشد که به دنبال ایجاد طرح کلی آن هستیم، ابتدا $\overline{S}_D$ را با استفاده از رابطه (21) و با k.s ترکیب مستقل از هم ایجاد می‌کنیم. پس از آن $\overline{S}_D$را به k بخش با طول s تقسیم می‌کنیم و بعر هر بخش را دوباره درهم‌سازی می‌کنیم. برای اینکه بین بخش‌های هر متن وابستگی وجود نداشته باشد هر بخش را با تابع متفاوتی درهم‌سازی می‌کنیم.(مثلا اگر از روش Rabin's Fingerprint استفاده می‌کنیم می‌توانیم هر بخش را بر چند جمله‌ای متفاوتی تقسیم کنیم.)

با استفاده از این روش برای هر متن کافی است تنها Kتا از مقادیر درهم‌سازی شده را نگهداری کنیم. در [30] هر یک از مقادیر درهم‌سازی شده بخش‌های بردار متن، یک ویژگی متن[1] نامیده شده‌اند.

[1] Document Feature

با استفاده از این روش احتمال اینکه اثرانگشت[1] هر یک از بخش‌های مربوطه در دو متن با هم برابر باشند، عبارت است از

$$r^s + r_f$$

که در آن r_f ، احتمال برخورد[2] است، یعنی احتمال اینکه به دو بخش متفاوت اثرانگشت یکسانی اختصاص داده شود. و این یعنی اگر از اثرانگشت های به حد کافی بزرگ استفاده کنیم، این احتمال کاهش یافته و دقت حفظ می‌شود.

در[30] ، از اثرانگشت‌های 64 بیتی استفاده شده است و به این نکته اشاره شده است که 64 بیت برای کاربرد مورد نظر آنها، کلاستر کردن وب، به حد کافی بزرگ است.

توجه کنید که ایده تقسیم $\overline{S}_D$ به چند بخش و دوباره اثرانگشت کردن هر بخش مشابه با ایده super shingling است.

2.3.2.6 روش Winnowing

این روش هم یکی از روش‌هایی است که از توابع درهم ساز[3] برای ایجاد نمایشی از متن یا اشیایی که به دنبال تقریباً یکسان‌ها بین آنها می‌گردد استفاده می‌کنند.

روش winnowing[4] که در [32] ارائه شده است در واقع روشی است برای انتخاب مقادیر درهم‌سازی به دست آمده از n-gram های متن به طوری که بتوان تضمین کرد در صورت وجود رشته‌ای مشترک بین دو متن که طول آن از حد آستانه خاصی بلندتر است، حتماً چنین تطابقی بین دو متن قابل شناسایی باشد. در واقع این روش برای رفع معایب موجود در روش در انتخاب مقادیر درهم‌سازی به دست آمده از n-gram های متن توسط روش mod m = 0 ارائه شده است.

[1] در این روش‌ها معمولاً کلمات Fingerprint و Hash Value به صورت معادل به کار گرفته می‌شوند.

[2] Collision

[3] Hash Functions

[4] Winnowing در اصطلاح به معنای غربال کردن می باشد.

همانطور که نویسندگان این مقاله به آن اشاره کرده اند،در روش mod m = 0 ، ممکن است از بخش‌های طولانی از متن هیچ n-gram ای در مجموعه نماینده متن حضور نداشته باشد چرا که مقادیر درهم‌سازی اختصاص داده شده به n-gram های آن، هیچ یک بر m بخش پذیر نیستند. و هر زیر رشته مشترک بین دو متن هرقدر هم که طول آن زیاد باشد تنها در صورتی قابل شناسایی است که مقادیر درهم‌سازی اختصاص داده شده به n-gram های آن بر m بخش پذیر باشند. در [32] گزارش شده است که در مشاهدات آنها، در برخی از صفحات وب، در بخش‌هایی طولانی‌تر از متوسط طول صفحات وب مورد بررسی، هیچ n-gram ای برای مجموعه نماینده متن انتخاب نشده است.

به همین دلیل و برای حل این مساله در [32] دسته‌ای از الگوریتم‌ها برای انتخاب مقادیر درهم‌سازی اختصاص داده شده به n-gram ها برای مجموعه نماینده متن به نام الگوریتم‌های محلی[1] ارائه شده است که تضمین می‌کند که از هر تطابق به حد کافی بزرگ بین دو متن قابل شناسایی باشد.

پس از تعریف خصوصیات این دسته از الگوریتم‌ها نیز الگوریتم winnowing ارائه شده است که نوعی الگوریتم محلی است و سپس با روش 0 mod m مقایسه شده است.

2.3.2.7 الگوریتم‌های محلی

هدف از معرفی و استفاده از الگوریتم‌های محلی این است که :

1- اگر زیر رشته ای مشترک بین دو متن با حداقل طول t وجود دارد این تطابق تشخیص داده شود.

2- نسبت به هر تطابق کوتاه تر از حد آستانه نویز حساس نباشد.

ورد اول برای رفع مشکلی است که در روش 0 =mod m وجود دارد و در بخش قبلی از آن صحبت شد. مورد دوم هم برای این است که روش ایجاد شده در برخورد با کلمات پرکاربردی مثل "the" که در اغلب متن‌ها وجود دارد گمراه نشود. برای برخورد با مشکل اول n-gram های متن را به عنوان اجزای آن در نظر می‌گیریم. و برای برخورد با مشکل اول اگر فرض کنیم $h_1,...,h_l$، hash های متنی باشد که می‌خواهیم هر انطباق با طول بیشتر از t در آن را با هر متن دیگری تشخیص دهیم. در این صورت اگر w = t-n+1 را به عنوان اندازه پنجره لغزنده روی $h_1,...,h_l$ قرار دهیم از هر یک از پنجره‌های متن باید که

[1] Local Algorithms

hash را انتخاب کرده و مجموعه نماینده متن قرار دهیم. در این صورت برای متنی با hash های $h_1,...,h_l$ برای هر $1 \le i \le l-w+1$ ، h_i شروع یک پنجره خواهد بود.

تعریف - الگوریتم محلی: *اگر* S *یک تابع انتخاب روی یک* w *تایی باشد و با دریافت یک* w *تایی عددی بین* 0 *و* w-1 *را برگرداند، یک الگوریتم انتخاب مجموعه نماینده متن محلی است اگر برای هر پنجره روی مقادیر* hash، $h_i,...,h_{i+w-1}$، *عنصری را که در موقعیت* $i+S(h_i,...,h_{i+w-1})$ *است را برگزیند.*[1]

در واقع روش‌های محلی روش‌هایی هستند که از پنجره به طول w در متن حتماً یک مقدار hash در نماینده متن وجود دارد.

در [32] اثبات شده است که هر روش محلی با تعریف بالا می‌تواند تضمین کند که در هر زیر رشته مشترک با حداقل طول t را بین 2 متن شناسایی کند. که در آن t=w-n+1، w اندازه پنجره متن و n طول n-gram ها است.

اگر نسبت تعداد hash هایی که یک روش انتخاب برای نمایش متن برمی‌گزینند به کل تعداد مقادیر hash موجود را چگالی آن روش انتخاب بنامیم در [32] اثبات شده است که برای چگالی تمام روش‌های محلی حد پایینی وجود دارد که برابر است با

$$d \ge \frac{1/5}{w+1} \qquad (36)$$

که در آن w اندازه پنجره‌های متن است.

2.3.2.8 الگوریتم winnowing

دیدیم که برای تضمین تشخیص هر زیررشته به طول t باید از هر پنجره روی متن با اندازه w=t-n+1، یک hash انتخاب شود. الگوریتم winnowing که یک الگوریتم محلی است برای این کار ارائه شده

[1] Let S Be A Selection Function Taking A W-Tuple Of Hashes And Returning An Integer Between Zero And W-1, Inclusive. A Fingerprinting Algorithm Is Local With Selection Function S, If , For Every Window $h_i, \dots, h_{i+w-1}$, The Hash At Position $h_i, \dots, h_{i+w-1}$ Is Selected As A Fingerprint.

است. این الگوریتم سعی دارد ضمن اینکه تعداد hash های نماینده متن را کم نگه دارد برای هر پنجره یک hash انتخاب شده داشته باشد.

الگوریتم winnowing: در هر پنجره کوچکترین مقدار hash را انتخاب کنید. اگر در یک پنجره بیش از یک hash با کوچکترین مقدار وجود دارد سمت راست ترین مقدار را انتخاب کنید. و پس از آن تمام hash های انتخاب شده را در مجموعه نماینده متن مورد نظر قرار دهید.

برای روشن تر شدن این مساله یک مثال از [32] ذکر می‌کنیم:

اگر متن مورد نظر ما متن زیر باشد:

A do run run run ,a do run run

پس از انجام پیش‌پردازش‌هایی مثل حذف فضاهای خالی، کوچک کردن حروف بزرگ و حذف نقطه-گذاری‌ها این متن تبدیل می‌شود به

adorunrunrunadorunrun

5-gram های این متن :

adoru, dorun, orunr , runru, unrun , nrunr, runru,
unrun, nruna, runad, unado, nador , adoru, dorun,
orunr, runru, unrun

hash های بدست آمده برای 5-gram های بالا:

77, 74, 42, 17, 98, 50, 17,
98, 8 , 88, 67, 39, 77, 7
4 , 42, 17, 98

پنجره‌های متن و hash های انتخاب شده در هر پنجره:

(17, 98, 50, 17) (42, 17, 98, 50) (74, 42, 17, 98) (77, 74, 42, 17)
(98, 8 , 88, 67) (17, 98, 8 , 88) (50, 17, 98, 8) (98, 50, 17, 98)
(39, 77, 74, 42) (67, 39,77, 74) (88, 67, 39, 77) (8, 88, 67, 39)
(74, 42, 17, 98) (77, 74, 42, 17)

Hash انتخاب شده در هر پنجره با رنگ دیگری مشخص شده است. همانطور که دیده می‌شود، انتخاب کوچکترین hash در هر پنجره باعث می‌شود در صورتی که در پنجره بعدی hash کوچکتری وارد نشده باشد، همان hash قبلی در این پنجره هم انتخاب شود. اگر هر hash را تنها در اولین پنجره‌ای که انتخاب شده است در نظر بگیریم مجموعه hash های انتخاب شده برای متن بالا عبارت خواهد بود از (hashهایی که با رنگ دیگر مشخص شده‌اند و پررنگ شده‌اند.)

17, 17, 8, 39, 17

در [32] اثبات شده است که حدبالای چگالی این روش $\frac{2}{w+1}$ است.

2.3.3 LSH

موزز در [31]، روشی دیگر برای شناسایی متون تقریباً یکسان ارائه کرده است. چاریکار در این مقاله به این نکته اشاره می‌کند که بسیاری از الگوریتم‌های مطرح در خصوص پردازش روی داده‌های جریانی[1]، داده‌های با حجم زیاد و یا با ابعاد بالا، برای غلبه بر پیچیدگی‌های زمانی و فضایی نیازمند نگهداری ردپایی از داده‌هایی که تابحال دیده‌اند هستند. این الگوریتم‌ها برای ایجاد این ردپا، از ویژگی‌های مهم و تاثیرگذار داده‌هایی که تابحال دیده‌اند نماینده یا طرحی کلی ایجاد و نگهداری می‌کنند تا در صورت نیاز اندازه گیری‌ها یا مقایسه‌هایی که لازم است روی داده‌های اصلی انجام شوند به صورت کارا از روی این طرح‌های کلی یا نماینده‌ها قابل اندازه‌گیری یا تخمین باشند. در واقع برخی از این الگوریتم‌ها منجر به روش‌هایی برای ایجاد طرح کلی از انواع مختلف داده‌ها شده اند. این طرح‌های کلی از نقطه نظر اینکه چه نوع مقایسه‌ای بین داده‌های اصلی مورد نظر هستند و چه ویژگی‌هایی در این مقایسه مهم هستند می‌توانند تفاوت داشته باشند.

Charikar در [31] پس از اشاره به این نکته، روی روش‌هایی که در ایجاد طرح کلی به منظور مقایسه تساوی داده‌های اصلی کار می‌کنند، تمرکز کرده است و پس از ارائه تعریفی از توابع درهم‌سازی حساس به موقعیت[2] و بررسی خواص روابط شباهتی که توسط این توابع قابل تخمین هستند، ارتباط جالبی بین الگوریتم‌های تقریب و توابع درهم سازی حساس به موقعیت برقرار کرده و با استفاده از آنها برای دو

[1] Stream Data

[2] Locality Sensitive Hashing Functions

رابطه شباهت، تعریفی از توابع درهم‌سازی حساس به موقعیت معرفی کرده است تا از این طریق بتوان برای اشیا قابل مقایسه توسط آنها طرح‌هایی کلی بدست آورد. یکی از این روابط شباهت رابطه cosine است که در مقایسه متون و بازیابی اطلاعات کاربرد زیادی دارد و برخی از روش‌های شناسایی متون تقریباً یکسان نیز از این معیار در مقایسه متون استفاده می‌کنند. در بخش‌های بعدی این فصل این روش را به تفصیل بررسی می‌کنیم.

2.3.3.1 توابع درهم‌ساز حساس به موقعیت

مفهوم توابع درهم‌ساز حساس به موقعیت اولین بار توسط Indyk و Motwani معرفی شد. آنها هر خانواده،F، از توابع درهم‌ساز را که در خاصیت صدق کند، خانواده توابع درهم ساز حساس به موقعیت نامیدند:

اگر $sim(x,y) \geq r_1$ آنگاه داشته باشیم:

$$\Pr_{h \in F}[h(x) = h(y)] \geq p_1 \qquad (37)$$

و اگر $sim(x,y) \leq r_2$ آنگاه داشته باشیم:

$$\Pr_{h \in F}[h(x) = h(y)] \leq p_2 \qquad (38)$$

که در این تعریف، sim(x,y) میزان شباهت دو شی، به عنوان مثال دو متن، تحت یک معیار شباهت خاص است. موتوانی و همکارانش از این تعریف در مساله نزدیکترین همسایگی تقریبی[1] استفاده کرده‌اند. در [31]، این تعریف با کمی تغییر به صورت زیر آورده شده است:

تعریف- توابع درهم‌ساز حساس به موقعیت: یک روش درهم‌سازی حساس به موقعیت، یک توزیع از یک خانواده،F، از توابع درهم‌ساز روی یک مجموعه از اشیا است به طوری که برای هر دو شی مثل x , y در مجموعه مذکور داشته باشیم :

$$\Pr_{h \in F}[h(y)] = sim(x,y) \qquad (39)$$

[1] Approximate Nearest Neighbor

که در این رابطه، sim(x,y) یک رابطه شباهت است که روی مجموعه ای از اشیا تعریف شده است.[1]

با داشتن یک خانواده از توابع درهم‌ساز حساس به موقعیت برای یک تابع شباهت می‌توان نمایش فشرده یا طرحی کلی برای اشیا به دست آورد به طوری که شباهت مابین اشیا را بتوان از این نمایش‌های فشرده تخمین زد.

2.3.3.2 توابع درهم‌ساز حساس به موقعیت برای بردارها

مجموعه‌ای از بردارهای d بعدی را در فضای R^d فرض کنید. و فرض کنیم $\vec{r}$ یک بردار رندم از توزیع گوسی d بعدی است، یعنی هر یک از عناصر آن از توزیع گوسی یک بعدی انتخاب شده‌اند، در این صورت برای بردارهای مفروض در فضای R^d می‌توانیم تابع درهم‌سازی به شکل تعریف کنیم :

$$h(\vec{u}) = \begin{cases} 1 & if\ \vec{u}.\vec{r} \geq 0 \\ 0 & if\ \vec{u}.\vec{r} < 0 \end{cases} \qquad (40)$$

بنابراین برای بردارهای $\vec{v}$ و $\vec{u}$ خواهیم داشت

$$pr[h_{\vec{r}}(\vec{u}) = h_{\vec{r}}(\vec{v})] = 1 - \frac{q(\vec{u},\vec{v})}{p} \qquad (41)$$

در این رابطه $\frac{q(\vec{u},\vec{v})}{p}$ نماینده زاویه بین بردارهای $\vec{v}$ و $\vec{u}$ است. $1-\frac{q(\vec{u},\vec{v})}{p}$ با کسینوس این زاویه مرتبط است. به همین دلیل این رابطه به معیار شباهت کسینوسی که در بازیابی اطلاعات به صورت گسترده ای استفاده می‌شود بسیار مرتبط است.

بنابراین اگر متون را با استفاده از بردارها نمایش دهیم و معیار شباهت آنها را هم معیار کسینوسی در نظر بگیریم در این صورت با استفاده از این تابع درهم ساز می‌توانیم متونی را که تحت این معیار تقریباً یکسان هستند را جستجو کنیم.

[1] A Locality Sensitive Hashing Scheme Is A Distribution On A Family F Of Hash Functions Operating On A Collection Of Objects, Such That For Two Objects X,Y $pr_{h\in F}[h(x)=h(y)]=sim(x,y)$

Here Sim(X,Y) Is Some Similarity Function Defined On The Collection Of Objects.

یکی از مزایای این روش این است که بردارهای متن با فضای زیاد را به برداری با ابعاد کوچکتر تبدیل می‌کند. به عنوان مثال در [33]، که از این روش برای یافتن جفت‌های تقریباً یکسان در مجموعه ای 8 بیلیونی از صفحات وب استفاده شده است، نشان داده شده است که استفاده از یک اثر انگشت تنها 64 بیتی برای هر صفحه کافی است.

2.3.3.3 وجود توابع درهم‌ساز حساس به موقعیت برای روابط شباهت

در بخش‌های قبلی به این نکته اشاره شد که اگر برای رابطه شباهتی بتوانیم خانواده‌ای از توابع درهم‌ساز حساس به موقعیت را معرفی کنیم، با استفاده از آن می‌توانیم طرحی کلی یا نمایشی فشرده از اشیا بدست بیاوریم و میزان شباهت اشیا را با استفاده از آن تخمین بزنیم. اما آیا برای هر تابع شباهتی می‌توانیم چنین توابع درهم‌سازی که در واقع شباهت را حفظ می‌کنند را معرفی کنیم؟ در این بخش در مورد خواصی که لازم است یک رابطه شباهت داشته باشد تا برای آن چنین توابعی وجود داشته باشد را بررسی می‌کنیم.

Charikar در [31]، پس از بیان و اثبات چند شرط لازم برای اینکه رابطه شباهتی دارای چنین توابع درهم سازی باشد، نشان داده است که چند رابطه شباهت متداول مثل sim_{Dice} (رابطه شباهت Dice) و sim_{ovl} (رابطه شباهت همپوشانی یا overlap) نمی‌توانند دارای چنین توابع درهم‌سازی باشند. ما هم در اینجا پس از بیان این شرایط وجود یا عدم وجود توابع درهم‌ساز حافظ شباهت را برای این روابط بررسی می‌کنیم.

قضیه- برای هر تابع شباهت sim(x,y) که دارای توابع درهم ساز حساس به موقعیت است که در تعریف 1 صدق می‌کنند، تابع فاصله 1 – sim(x,y) در نامساوی مثلثی صدق می‌کند.[1]

یعنی برای 3 شی x ,y, z داریم :

(42)

$$[1-sim(x,y)]+[1-sim(x,y)]\le[1-sim(x,y)]$$

[1] For Any Similarity Function Sim(X,Y) That Admits A Locality Sensitive Hash Function Family, The Distance Function 1-Sim(X,Y) Satisfies Triangle Inequality.

اثبات این قضیه در [31] آمده است و ما در اینجا از آوردن آن صرف نظر می‌کنیم.

با استفاده از همین قضیه می‌توانیم نشان دهیم که برای sim_{Dice} و sim_{ovl} با رابطه (42) و (43) نمی توانیم توابع درهم‌ساز حساس به موقعیت داشته باشیم،چرا که $1-sim_{Dice}$ و $1-sim_{Ovl}$ در نامساوی مثلثی صدق نمی‌کند.

$$sim_{Dice}(A,B)=\frac{|A \mathbf{I} B|}{\frac{1}{2}(|A|+|B|)} \qquad (43)$$

$$sim_{Ovl}(A,B)=\frac{|A \mathbf{I} B|}{\min(|A|,|B|)} \qquad (44)$$

اگر فرض کنیم $A=\{a\}$ و $B=\{b\}$ و C={a,b}، در این صورت

$$,Sim_{Dice}(A,B)=0,Sim_{Dice}(C,B)=\frac{2}{3},Sim_{Dice}(A,C)=\frac{2}{3}$$
$$1-Sim_{Dice}(A,C)+1-Sim_{Dice}(C,B)<1-Sim_{Dice}(A,B)$$

به طور مشابه برای Sim_{ovl} و با همان فرضیات بالا :

$$,Sim_{ovl}(A,B)=0,\ Sim_{ovl}(C,B)=1,Sim_{ovl}(A,C)=1$$
$$1-Sim_{Ovl}(A,C)+1-Sim_{Ovl}(C,B)<1-Sim_{Ovl}(A,B)$$

بنابراین $1-Sim_{Dice}$ و $1-Sim_{ovl}$در نامساوی مثلثی صدق نمی‌کنند.

قضیه - اگر یک خانواده از توابع درهم ساز حساس به موقعیت مطابق با تابع شباهت sim(x,y) داشته باشیم و آن را با F نمایش بدهیم، می‌توانیم یک خانواده دیگر از توابع درهم‌ساز حساس به موقعیت دیگر، که آن را با 'F نمایش می‌دهیم، را می‌توانیم بدست آوریم که اشیا را به مجموعه {0,1} می‌نگارد و مطابق با تابع شباهت $\frac{1+sim(x,y)}{2}$ است.[1]

[1] Given A Locality Sensitive Hash Function Family F Corresponding To A Similarity Function Sim(X,Y), We Can Obtain A Locality Sensitive Hash Function Family F' That Maps Objects To {0,1} And Corresponds To The Similarity Function $\frac{1+sim(x,y)}{2}$.

همچون قضیه قبلی اثبات این قضیه هم در [31] آمده است. از این قضیه می‌توانیم استفاده کنیم و شرط قوی‌تری برای وجود یا عدم وجود توابع درهم‌ساز حساس به موقعیت مطابق با تابع شباهت مورد نظر به دست بیاوریم که در قضیه 3 آمده است.

قضیه 3- برای هر تابع شباهت sim(x,y) که برای آن بتوانیم خانواده ای از توابع درهم‌ساز حساس به موقعیت معرفی کنیم، تابع فاصله 1- sim(x,y) به صورت ایزومتریک در مکعب همینگ محاط می‌شود.[1]

برای اثبات این قضیه نیز به [31] باید مراجعه کرد.

از این قضیه می‌توان نتیجه‌گیری کرد که از هر تبدیل ایزومتریک از 1-sim(x,y) که قابل محاط شدن در مکعب همینگ باشد می‌توان یک خانواده از توابع درهم‌ساز حساس به موقعیت برای $\frac{a+sim(x,y)}{a+1}$ بدست آورد که در آن $a>0$.

2.3.4 اثر انگشت فازی

در [11] Stein با معرفی کلاس‌های پیشوندی[2] ، به عنوان تمام توکن‌هایی که با پیشوند یکسانی شروع می‌شوند، توزیع احتمال هر یک از کلاس‌ها را در کل مجموعه و در هر متن محاسبه می‌کند. سپس از انحراف موجود مابین توزیع به دست آمده برای هر متن و کل مجموعه یک اثر انگشت فازی[3] محاسبه می‌کند. فرض این روش در واقع این است که متون مشابه مقدار اثر انگشت فازی یکسانی خواهند داشت و پس از به دست آوردن آن برای تمام متون می‌توان متون مشابه به یک متن را در زمان خوبی به دست آورد.

Stein در این مقاله مشابه با تمام روش‌های مبتنی بر شباهت[4] تابع hash ای را معرفی کرده است که در رابطه (44) صدق می‌کند:

[1] For Any Similarity Function Sim(X,Y) That Admits A Locality Sensitive Function Family, The Distance Function 1-Sim(X,Y) Is Isometrically Embedded In The Hamming Cube.

[2] Prefix Classes

[3] Fuzzy-Fingerprint

[4] Similarity-Based

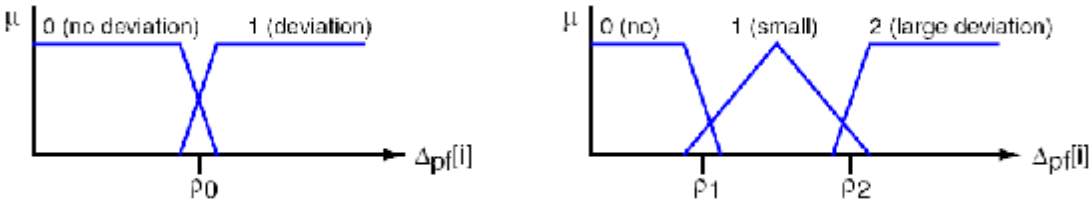
(45)

$$h_j(d) = h_j(d') \Rightarrow j(d,d') \geq 1-e, 0 < e << 1$$

که در این رابطه d , d' دو متن متفاوت هستند. تابع j، تابع شباهت است و ε آستانه شباهت است. h_j یک تابع hash فازی است. این رابطه جزء روش‌های درهم‌سازی حساس به مکان است و همچنان که می‌بینید؛ j می‌تواند هر نوع تابع شباهت باشد. Stein در [11] برای تابع شباهت کسینوسی این تابع hash ارائه شده است. به این صورت که تابع hash برای متن d از رابطه (45) محاسبه می‌شود:

$$h_j^{(r)}(d) = \sum_{i=0}^{k-1} d_i^{(r)} . r^i , with d_i^{(r)} \in \{0,...,r-1\} \qquad (46)$$

در این رابطه r تعداد ترم‌های فازی تعریف شده بر حسب تابع شباهت cosine است و σi هم درجه عضویت متن در ترم i ام است. Stein در این مقاله 2 مجموعه ترم به صورت زیر تعریف کرده است :



شکل 13- مجموعه ترم های مورد استفاده در سیستم فازی [11]

2.4 روشهای معنایی

در این بخش مقالاتی را که برای محاسبه شباهت متون از روشهای معنایی استفاده کرده‌اند را بررسی می‌کنیم. همانطور که قبلاً هم اشاره شد، این روشها اغلب در کاربرد شناسایی تقلب ارائه شده‌اند.

استفاده از لغتنامه، اصطلاح‌نامه‌ها و سلسله مراتب کلمات یکی از روش‌هایی است که در این زمینه مورد استفاده قرار گرفته است. [39] ضمن مقایسه آماری ساختار جملات متن‌ها، با استفاده از یک اصطلاحنامه، کلمات مترادف جایگزین شده در متن را شناسایی می‌کند. [39] برای استخراج ساختار جملات از یک پارسر استفاده می‌کند. [40] با استفاده از یک ساختار سلسله مراتبی از کلمات، که رابطه جز و کل در کلمات را نشان می‌دهد و نگاشت هر کلمه به کلی ترین حالت ممکن اقدام به شناسایی متون تقریباً یکسان می‌کند. [41] نیز با به کارگیری روش LSI، تقلب شناسایی کرده است.

3 چالش‌های شناسایی متون تقریباً یکسان

با توجه به روش‌های مطرح شده در فصل قبل، از جمله چالش‌های پیش روی شناسایی متون تقریباً یکسان، می توان به موارد زیر اشاره کرد:

1- ارائه نمایشی مناسب از متن که حاوی تمام اطلاعات مورد نظر از متن برای تشخیص متون تقریباً یکسان باشد، یکی از چالش‌های اصلی این زمینه است. همانطور که در روش‌های مطرح شده در فصل دوم مشاهده می‌شود تمام این الگوریتم‌های موجود برای نمایش متن از جمله مدل فضای برداری و n-gram ها استفاده می‌کنند. در حالی که این روش‌ها به عنوان نماینده متن،بسیاری از اطلاعات موجود در متن را نادیده می‌گیرند. به عنوان مثال مدل فضای برداری قابلیت حفظ اطلاعات مربوط به ترتیب کنار هم قرار گرفتن کلمات را ندارد.

2- به همین ترتیب و پیرو چالش بالا معرفی روشی مناسب برای اندازه گیری شباهت مابین دو مت ناز دیگر چالش‌های این مساله است.

3- اندازه مجموعه متون و همچنین ابعاد بالای داده‌های متنی یکی از چالش‌های پیش روی شناسایی متون تقریباً یکسان است.

همانطور که در فصل دوم نیز اشاره شد، شناسایی متون تقریباً یکسان حالتی خاص از مساله نزدیکترین همسایگی تقریبی است. برای حل این مساله روش‌های مختلفی از جمله درخت‌های k بعدی[1] و درخت‌های متریک[2] ارائه شده است. اما به دلیل ابعاد بالای متن این روش‌ها بر متن قابل اعمال نیستند.

4- زمان پاسخگویی سیستم، خصوصاً در کاربردهایی مثل شناسایی صفحات یکسان در وب از دیگر چالش‌های این سیستم‌هاست. همین مساله است که منجر به ارائه روش‌های موسوم به روش‌های تقریبی شده است، که در فصل قبل به تفصیل مورد بررسی قرار گرفت.

5- عدم وجود مجموعه داده استاندارد در این زمینه جهت مقایسه و ارزیابی روش‌های مختلف یکی دیگر از مسائل پیش رو در این حیطه است. مجموعه داده های متنی استاندارد موجود به دلایل

[1] K-D Trees
[2] Metric Trees

مختلفی از جمله، درصد پایین متن‌های با درجه شباهت بالا در آنها و نیز عدم وجود مقایسه دوبه دو مابین متن‌های آنها جهت بررسی تقریباً یکسانی برای ان کاربرد مفید نیستند. از سوی دیگر ایجاد مجموعه داده‌ای بزرگ که دارای ویژگی‌های لازم در این زمینه باشد مستلزم وقت و هزینه فراوانی است؛ چرا که دست کم برای تهیه چنین مجموعه‌ای تمام متون مجموعه باید دوبه دو با هم مقایسه شوند که انجام چنین کاری توسط انسان با توجه به حافظه لازم برای این کار غیر ممکن به نظر می‌رسد.

در کنار این مساله باید به شناسایی متون تقریباً یکسان در مجموعه داده‌های چندزبانه اشاره کرد که نسبت به مجموعه داده تک زبانی برای ایجاد آن نیاز به تلاش و کار مضاعفی خواهیم داشت.

البته در اینجا لازم است به[5] تلاش‌هایی در این راستا صورت گرفته است. به علاوه در [38] برای ساختن مجموعه داده نسبتاً کوچک ساخته شده از برخی از ابزارهای اتوماتیک استفاده شده است که در ساختن مجموعه داده استاندارد می‌تواند مورد بررسی قرار بگیرد.

6- تمام روش‌های مورد بررسی بر روی متون با اندازه کوتاه اعمال شده‌اند. به عنوان مثال هیچ یک از این روش‌ها بر روی متون طولانی انجام نشده‌اند و کارایی این روش‌ها در خصوص این روش‌ها مشخص نیست. در صورتی که یکی از کاربردهای شناسایی متون تقریباً یکسان می‌تواند بررسی رعایت قانون کپی رایت باشد.

7- در نهایت باید به این نکته اشاره کرد که در بسیاری از کاربردها نیاز به سیستمی داریم که بتواند تصمیم خود را توجیه کند و دلایل قانع کننده در این رابطه بیاورد. اگر همان سیستم بررسی رعایت قانون کپی رایت که بند قبل اشاره شد را در نظر بگیریم، در این سیستم نیاز است سیستم قادر باشد دلایل قانع کننده‌ای در خصوص تصمیم خود ارائه دهد، چرا که در صورت شناسایی متون تقریباً یکسان با متن ورودی، نتیجه عدم رعایت این قانون از جانب دست کم نویسندگان یکی از متن‌هاست.

3.1 چالش‌های زبان فارسی

چالش‌هایی که در بخش قبل بررسی شد به طور کلی در خصوص تمام سیستم‌های شناسایی متون تقریباً یکسان برقرار هستند. از اینرو در تولید سیستم‌های شناسایی متون تقریباً یکسان فارسی نیز باید این چالش‌ها مورد توجه قرار بگیرند.

به جز روش‌های معنایی هیچ یک از روش‌هایی که در فصل دوم مورد بحث قرار گرفته‌اند، وابسته به ویژگی خاص از زبان خاصی نیستند، به همین دلیل تمام این روش‌ها بر زبان فارسی قابل اعمال هستند. تنها پیش‌نیازهایی که در خصوص به کارگیری این روش‌ها برای زبان فارسی وجود دارد، تنظیم برخی از پارامترهای این روش‌هاست که وابسته به زبان هستند. در فصل پنجم به تفصیل به این روش‌ها و پارامترهای آنها خواهیم پرداخت.

در خصوص روش‌های معنایی عدم وجود اصلاحنامه و لغت نامه‌های جامعی که در این زمینه قابل استفاده باشند، مهمترین چالش پیش روی زبان فارسی به شمار می رود.

4 روش‌های ارزیابی متون تقریباً یکسان

مفهوم شباهت، متون تقریباً یکسان، یا تقریباً کپی با اینکه در نظر اول مفهوم ساده و واضحی به نظر می رسد؛ اما درعمل تعریف واحدی ندارد. همانطور که قبلاً نیز اشاره شد متون مشابه به دلایل مختلفی مثل mirroring در سرورهای وب، ویرایش و به‌روز رسانی متون، تقلب و موارد دیگری ایجاد می‌شوند. و در هر مورد معنای متفاوتی دارد.

این مساله روش ارزیابیِ روش‌های ارائه شده را نیز تحت تاثیر قرار می‌دهد. معمولاً شناسایی متون یکسان در مورد مجموعه‌هایی با سایز بسیار زیاد انجام می‌شود. در این گونه مجموعه‌ها امکان بررسی داده‌ها و برچسب‌گزاری آنها وجود ندارد. به همین دلیل در اکثر مقالات و کارهایی که انجام شده است نویسندگان تعریفی برای این متون ارائه کرده‌اند و سپس با همین تعریف الگوریتم یا روش خود را ارزیابی کرده‌اند. این به اصطلاح یک دور منطقی است و نشان می‌دهد که عملاً نمی‌توانیم به نتایج این الگوریتم‌ها در واقعیت تکیه کنیم. برای روشن شدن این مساله مثالی را که در [25]، Zobel و Bernstein از آن استفاده کرده‌اند را ذکر می‌کنیم. ایشان ضمن اشاره به این دور، این مساله را مثل این دانسته اند که محققی الگوریتمی برای پیدا کردن متن هایی که خاصیت خاصی دارند ، مثلا grue هستند، بنویسد و هر متن grue باشد، اگر توسط الگوریتم او به عنوان grue شناسایی شود. [1] این مساله در مورد بسیاری از روش‌های موجود برقرار است.

به مساله ارزیابی روشهای شناسایی متون یکسان خواهیم پرداخت اما در واقع علت این مساله را می‌توان در تنوع انواع متون مشابه جستجو کرد.به عنوان مثال در موتورهای جستجو، درجه بالایی از شباهت بین 2 صفحه مورد نظر است، مثلاً اینکه 2 صفحه ظاهر یکسانی داشته باشند. یا اینکه محتوای اصلی آنها یکسان باشد و تنها در تبلیغاتی که به صورت پویا به صفحه اضافه می‌شود یا تاریخ و ساعت بازدید از صفحه که در برخی از صفحات درج می‌شود و یا آدرس پایه لینک‌ها و عکس‌ها[2] با هم تفاوت داشته باشند. اما در تشخیص تقلب دو متن می‌توانند با درجه کمتری مشابه باشند، مثلاً پارگراف‌های آنها جابه-جا شده باشند، برخی از کلمات با مترادف‌هایشان جایگزین شده باشند و ... . در [25] نویسندگان ضمن

[1] "A Researcher Develops An Algorithm For Locating Documents That Are Grue (Where Grue Is A New Property Of Documents That The Researcher Has Decided To Investigate) And Documents Are Dened As Being Grue If They Are Located By The Algorithm." [25]

[2] این مورد وقتی پیش می آید که سایت مورد بررسی روی یک سرور دیگر اصطلاحا Mirror شده باشد.

اشاره به این مساله 3 نوع شباهت مابین صفحات از دید موتورهای جستجو تعریف کرده‌اند و بر اساس این تعاریف سیستم خود را ارزیابی کرده‌اند. در زیر لیستی از روش های ارزیابی مورد استفاده و مقالاتی که از این روشها استفاده کرده‌اند می‌آید.

4.1 صحت و فراخوانی

در اکثر کارهایی که در زمینه بازیابی اطلاعات[۱] هستند استفاده از شاخص‌های ارزیابی صحت[۲] و فراخوانی[۳] متداول است. در تشخیص متون مشابه نیز در بسیاری از مقالات، مساله شناسایی متون تقریباً یکسان به صورت پیدا کردن متون مشابه یا تقریباً یکسان با متن مورد پرسش[۴] در نظر گرفته شده است. و به همین دلیل از صحت و فراخوانی برای ارزیابی سیستم استفاده شده است.

در صورتی که D مجموعه داده برچسب دار باشد و q متنی باشد که به دنبال تقریباً یکسان‌های آن در D هستیم. اگر D_q مجموعه تقریباً یکسان‌های q در D باشد و $\hat{D}_q$ مجموعه تقریباً یکسان‌هایی باشد که توسط الگوریتم مورد نظر شناسایی شده‌اند در این صورت صحت و فراخوانی از روابط (46) و (47) محاسبه می‌شوند :

$$prec = \frac{\left|\hat{D}_q \ \mathbf{I} \ D_q\right|}{\hat{D}_q} \qquad (47)$$

$$rec = \frac{\left|\hat{D}_q \ \mathbf{I} \ D_q\right|}{D_q} \qquad (48)$$

سیستمی با صحت بالا سیستمی است که درصد خطای مثبت آن کم است و سیستمی با فراخوانی بالا سیستمی است که درصد خطای منفی[۵] آن کم است.

---

[۱] Information Retrieval

[۲] Precision

[۳] Recall

[۴] Query Document

[۵] False Negative

اما همانطور که در بخش معیارهای شباهت نیز بحث شد استفاده از صحت و فراخوانی برای ارزیابی اینگونه سیستم‌ها مفید نیست چرا که برای ارزیابی صحت نیاز به داده‌های برچسب دار داریم در حالی که در این سیستم‌ها چنین داده‌هایی در دسترس نیست و عملاً هر خروجی الگوریتم به عنوان جواب درست در نظر گرفته می‌شود. به همین دلیل محاسبه صحت در این سیستم‌ها معنایی ندارد.

در [25] ، Zobel و Bernstein هم به این مساله اشاره کرده‌اند. در [2]، نیز Zobel و Hoad ضمن اشاره به این نکته برای ارزیابی سیستم خود 2 معیار تفکیک[1] و بزرگترین درصد شباهت غلط[2] را معرفی و استفاده کرده‌اند که شرح آن در بخش بعد خواهد آمد.

در[11]، Stein نشان داده است که چطور با استفاده از روش Fuzzy-Fingerprinting معرفی شده در آن مقاله به مقادیر مناسب فراخوانی دست پیدا کرده است؛ اما به دلیل ماهیت روش Fuzzy-Fingerprinting مورد استفاده باید روی مجموعه متون برگردانده شده یک الگوریتم دیگر اجرا شود تا بتوان به مقادیر صحت خوبی رسید. در [14] هم برای ارزیابی سیستم از معیارهای صحت و فراخوانی استفاده شده است.

در [38] مجموعه داده‌ای کوچک جمع آوری و سپس به صورت دستی برچسب گزاری شده است.[3] و سپس برای ارزیابی سیستم از شاخص های صحت و فراخوانی استفاده شده است.

4.2 تفکیک و بزرگترین درصد شباهت غلط

این دو کمیت در [2] و توسط Zobel و Hoad معرفی و استفاده شده اند. HFM بالاترین درصد شباهتی است که در عملیات بازیابی به یک نتیجه اشتباه اختصاص داده می‌شود. واضح است که HFM کمتر نشان دهنده سیستم بهتر است. تفکیک نیز تفاوت بین کمترین درصدی است که به یک جواب درست برگشتی اختصاص داده شده است. تفکیک در واقع نشان دهنده این است که سیستم چقدر مابین جواب های درست و نادرست تفاوت قائل می‌شود. این معیار وقتی قابل استفاده است که تمام جواب های درست برگردانده شده باشند؛ یعنی صحت و فراخوانی هر دو 1 باشند[2]. برای ارزیابی یک سیستم می-

[1] Separation

[2] Highest False Match یا بزرگترین درصد شباهت غلط که به اختصار Hfm گفته می‌شود.

[3] این مجموعه داده جمعاً دارای 1621 صفحه از 27 سایت خبری است [38].

بایست HFM و تفکیک را با هم در نظر گرفت. با وجود اینکه گفته شد HFM کمتر بهتر است، اما HFM بالاتر هم می تواند خوب و مورد پذیرش باشد اگر تفکیک هم بالا باشد. به طور مشابه تفکیک کوچک هم می‌تواند مورد قبول باشد؛ اگر HFM نیز کوچک باشد.

4.2.1ارزیابی توسط انسان

در [21]، Zobel و Bernstin برای ارزیابی سیستم خود، نتایج سیستم خود را با قضاوت افراد مقایسه کرده‌اند.

4.3تست مرتبط بودن

در [22]، پس از به دست آوردن کلاسترهایی از متون تقریباً یکسان روی مجموعه داده TREC add-hoc، که برای بازیابی اطلاعات طراحی شده است؛ نتایج بدست آمده را با استفاده از این مجموعه داده ارزیابی کرده‌اند. فرض این روش این است که اگر یکی از متون یک کلاستر از متن های تقریباً یکسان با یک متن مورد پرسش[1] مرتبط باشد باید متن‌های دیگر در آن کلاستر نیز با متن مورد پرسش مرتبط باشند.

4.4درصد متون تقریباً یکسان تشخیص داده شده

در [22]، علاوه بر استفاده از مجموعه داده TREC add-hoc و مقایسه نتایج بدست آمده با آن، از یک مجموعه داده دیگر که به صورت دستی تهیه شده، استفاده شده است. مجموعه مورد بحث طوری انتخاب شده است که متون آن حتماً تعداد زیادی متن مشابه یا تقریباً یکسان در مجموعه داشته باشند. در مقایسه الگوریتم ها و روشهای مختلف، در مورد این مجموعه، الگوریتمی موفق تر و دقیق تر ارزیابی شده است که تعداد بیشتری متون تقریباً یکسان شناسایی کرده باشد.

[1] Query

5 سیستم شناسایی متون تقریباً یکسان

در فصل دوم، روش‌های شناسایی متون تقریباً یکسان بررسی شد. تمام روشهای بررسی شده در فصل دوم بر روی متون زبان انگلیسی یا تمام وب که مجموعه‌ای است متشکل از متون به زبان‌های مختلف انجام شده‌اند. همانطور که در فصل دوم نیز اشاره شد تمام این روش‌ها پس از تنظیم پارامترهای لازم قابل اعمال بر زبان‌های دیگر از جمله زبان فارسی هستند.

در این فصل پس از بررسی پیش‌نیازهای اعمال هر یک از روش‌های مطرح شده در فصل دوم برای زبان فارسی، کارایی و ویژگی‌های این روش‌ها را بررسی خواهیم کرد.

همانطور که در فصل دوم اشاره شد از نظر نحوه تعریف مساله روش‌های شناسایی متون تقریباً یکسان را می توان به دو دسته روش‌های یک‌به چند و چندبه چند تقسیم کرد. روش‌های یک‌به چند با دریافت یک متن در مجموعه متون موجود، تمام متن‌های تقریباً یکسان با متن دریافت شده را جمع‌آوری می کنند.

روش‌های چندبه چند تمام کلاسترهای تقریباً یکسان موجود در یک مجموعه متن را شناسایی می‌کنند.

در این فصل شکل یک‌به چند روش‌های مطرح را بررسی می‌کنیم.

5.1 روش‌های مبتنی بر نمایه معکوس و فیلترینگ پیشوندی و پسوندی

در بخش 2-1 از فصل دوم روش‌های دقیق در دو دسته کلی روش‌های مبتنی بر نمایه معکوس و نیز روش‌های فیلترینگ بررسی شد. همانطور که اشاره شد هر دو این روش‌ها از نمایه معکوس استفاده می‌کنند با این تفاوت که روش‌های فیلترینگ با استفاده از اصولی که مطرح شد از ساختن نمایه معکوس کامل اجتناب می‌کنند و برای هر متن تنها از پیشوند آن متن در ساختن نمایه معکوس استفاده می‌کنند.

پس از ساخته شدن نمایه معکوس روش‌های مبتنی بر نمایه معکوس با استفاده از نمایه می توانند شباهت متون را به طور کامل محاسبه کنند، اما روش‌های فیلترینگ با استفاده از نمایه معکوس ساخته

شده مجموعه‌ای از متن‌ها که ممکن است با متن دریافتی تقریباً یکسان باشند را بـه دسـت مـی‌آورنـد و پس از آن تقریباً یکسان بودن اعضا این مجموعه را با متن مورد نظر بررسی می کنند. بـه بنـابراین طـور خلاصه این روش‌ها شامل مراحل زیر هستند:

1. در یکبار اسکن کردن مجموعه متون، کلمات متن‌ها استخراج شده و نمایه معکوس ساخته می-شود. همانطور که اشاره شد در این فاز ممکن است نمایه ساخته شده کامل و یا پیشوندی باشد.
2. با دریافت متن d، تمام یا بخش پیشوندی متن d در نمایه معکوس جستجو می‌شود و تمام متن هایی که با متن d از حد آستانه مورد نظر شباهت بیشتری دارند در مجموعه کاندیداها قرار می-گیرند.
3. اگر نمایه کامل از متون ساخته شده باشد مجموعه کاندیداها در واقع مجموعـه جـواب و شـامل متونی است که با متن d تقریباً یکسان هستند. در غیر اینصورت میزان شباهت مابین مـتن d و اعضا مجموعه کاندیدا باید محاسبه شود و متونی که شباهت آنها با d از حد آسـتانه مـورد نظـر کمتر است از این مجموعه کنار گذاشته شوند. پس از کنار گذاشتن این متون مجموعه باقیمانده مجموعه متون تقریباً یکسان با متن d خواهد بود.

این روش بر روی متون زبان انگلیسی اعمال شده است اما همانطور که دیده می‌شود این روش وابسته به ویژگی خاصی از زبان نیست و تنها پارامتری که در این روش باید تنظـیم شـود آسـتانه شـباهت تقریبـاً یکسان بودن دو متن است.

نکته دیگری که در خصوص این دو روش باید اشاره کرد این است که، از آنجایی در روش‌های فیلترینگ، تمام متن نمایه‌سازی نمی‌شود؛ برای محاسبه شباهت دقیق در مجموعه کاندیداها و متن d نیاز بـه خـود متن‌ها داریم؛ به همین دلیل حتی با وجود ساختن نمایه معکوس این روش به مجموعه متن نیاز دارد. اما در روش‌های مبتنی بر نمایه معکوس پس از ساختن نمایه معکوس دیگر نیازی به مجموعه مـتن نیسـت چرا که نمایه معکوس حاوی تمام اطلاعات مفید مجموعه است. اما به دلیـل اسـتفاده از نمایـه معکـوس کوچکتر، همانطور که در فصل دوم اشاره شد روش‌های فیلترینگ سریع تر از روش‌های مبتنی بـر نمایـه معکوس هستند.

5.2الگوریتم I-Match

به طور خلاصه این روش شامل مراحل زیر است:

1. ابتدا متون مجموعه وارد یک مرحله پیش‌پردازش می‌شوند. این مرحله شامل حذف کلمات کوتاهتر از طول متوسط کلمات مجموعه و کلمات طولانی تر از 25 کاراکتر و ... است.
2. فیلتر کردن کلمات براساس میزان شاخص idf آنها.
3. درهم‌سازی متون با استفاده از الگوریتم در هم‌سازی SHA1.
4. پس از درهم‌سازی متون، با دریافت متن d، متن مورد نظر تمام مراحل بالا را طی می‌کند. با مقایسه مقدار اختصاص داده شده به متن d در مرحله 4، با مقادیر اختصاص داده شده به متون مجموعه، متون تقریباً یکسان با متن d در مجموعه شناسایی می شوند. متون تقریباً یکسان با d متونی هستند که مقدار درهم‌سازی اختصاص داده شده به آنها با d یکسان باشد.

در رابطه با این روش اشاره به چند نکته ضروری است. اول اینکه از آنجایی که شاخص idf اختصاص داده شده به هر کلمه به کل مجموعه وابسته است، فیلتر کردن کلمات یک متن و به دست آوردن مقدار تابع درهم‌سازی برای آن متن، به تنهایی ممکن نیست و محاسبه این مقدار به کل مجموعه وابسته است.

و نکته دیگر اینکه با اینکه شکل کلی این روش وابسته به زبان نیست، نحوه فیلتر کردن کلمات به نظر می‌رسد که به زبان و حتی مجموعه داده وابسته باشد. در [26] پس از بررسی روش‌های مختلف از جمله فیلتر کردن کلمات با idf کمتر از یک حد آستانه، بیش از یک حد آستانه، بین دو حد آستانه بالا و پایین و یا خارج از یک بازه، فیلتر کردن کلمات با idf کمتر از یک حد آستانه، 0.1، از دیگر روش‌ها موثرتر شاخته شده و از این روش استفاده شده است. لازم به ذکر است که idf با فرکانس تکرار کلمه در مجموعه متن نسبت عکس دارد و حذف کلمات با شاخص idf کوچک مترادف با حذف کلمات پرکاربرد در متن است. بنابراین برای هر زبان دیگر، حتی هر مجموعه داده دیگر، لازم است شکل روش فیلترینگ و حد(یا حدود) آستانه آن تعیین شوند.

5.3روش Shingling

در فصل دوم، دربخش shingling، الگوریتم‌های shingling و super shingling با جزئیات مطرح شدند.

در این بخش، شرح مختصری از هریک، نقاط ضعف و قوت و پیش‌نیازهای اعمال هر یک بر زبان فارسی را بررسی می‌کنیم.

5.3.1 الگوریتم shingling

به طور خلاصه این الگوریتم شامل مراحل زیر است :

1. ابتدا متون مجموعه وارد مرحله پیش‌پردازش می‌شوند. دراین مرحله نقطه گذاری‌ها و جزئیاتی که مورد توجه نیستند حذف می‌شوند.
2. پس از آن n-gram ها یا shingle های متن استخراج می شوند. پس از این مرحله هر متن در مجموعه متون تبدیل به مجموعه‌ای از shingle ها خواهد شد. به منظور سهولت مراحل بعد و کاهش حجم مجموعه ها، shingle های به دست آمده توسط یک تابع درهم‌سازی مناسب به یک عدد صحیح نگاشته می‌شوند، به جای استفاده از خود shingle ها از این اعداد استفاده می-کنیم. به این اعداد هم shingle های متن گفته می‌شود.
3. پس از به دست آوردن مجموعه shingle های هر متن با استفاده از روش‌های پیشنهاد شده در فصل دو این مجموعه ها نمونه برداری می‌شوند. و از این پس مجموعه نمونه برداری شده نماینده متن مورد نظر خواهد بود.

پس از به دست آوردن مجموعه های نماینده متون در مرحله 3 و با دریافت متن d این متن نیز مراحل بالا را طی می‌کند تا مجموعه نماینده آن به دست آید. و پس از آن با استفاده از رابطه (25) و (26) شباهت این متون نسبت به تمام متون مجموعه بررسی می‌شود و در صورتی که شباهت اندازه گیری شده از حد آستانه خاصی بیشتر بود به عنوان متون تقریباً یکسان شناسایی می‌شوند.

shingling به اندازه shingle ها حساس است و این مقدار در دقت این روش تاثیر مستقیم دارد. اندازه shingle ها مقداری است وابسته به زبان و باید برابر با متوسط طول عبارات آن زبان باشد، برای زبان انگلیسی این مقدار بین 4 تا 10 کلمه تخمین زده شده است.

نحوه نمونه برداری از متن هم در نتایج و دقت shingling تاثیرگذار است. در فصل دوم با استفاده از روابط (21) و (22) دو روش برای نمونه برداری از متن مطرح شد. رابطه (22) shingle های متن را بر اساس بخش‌پذیر بودن بر عدد ثابتی مثل m نمونه برداری می‌کند. در صورت استفاده از این روش m یکی از پارامتر هایی است که باید تنظیم شود. البته همانطور که در بخش winnowing در فصل دوم بحث شد، در این روش ممکن است در بخش‌هایی طولانی از متن هیچ shingle ای از متن در نمونه متن انتخاب نشود.

روش دیگر برای نمونه برداری از متن روشی است که در رابطه (21) مطرح شده است. در این روش هم انتخاب ترتیب تعریف شده روی shingle هاست.

علاوه بر این روش‌های بالا در بخش 2-3-2 روش دیگری برای نمونه برداری از متن ارائه شد. در این روش به جای استفاده از s عنصر کوچکتر تحت یک ترکیب از shingle ها روی، از کوچکترین عنصر مجموعه تحت s ترکیب مختلف استفاده می شود. این روش می تواند نرخ خطای روش قبل را کاهش دهد. ترکیب‌های استفاده شده در این روش باید از یک دیگر مستقل باشند.

به علاوه در فصل دوم اشاره شد که برای کاهش حجم محاسبات و پایین آوردن نرخ خطای مثبت ، shingle های پرکاربرد متن از مجموعه های نماینده متن حذف شده اند.

5.4الگوریتم LSH

به طور خلاصه الگوریتم LSH را می‌توانیم در مراحل زیر بر روی متن پیاده‌سازی کنیم:

1. هر متن پس از انجام مراحل پیش پردازش به صورت یک بردار، مدل فضای برداری، نمایش داده می‌شود.
2. اگر ابعاد بردارهای متن d باشد. یک بردار d بعدی $\vec{r}$ که توزیع گوسی d بعدی انتخاب شده است را در نظر گرفته و ضرب داخلی آن را در بردارهای متن محاسبه می‌کنیم. با استفاده از m بردار $\vec{r}$ و رابطه (39) برای هر متن یک اثرانگشت[1] محاسبه می‌شود.

[1] Fingerprint

3. پس از محاسبه اثرانگشت تمام متن‌ها، با دریافت یک متن d و محاسبه اثرانگشت متن d در مراحل بالا، تمام متونی که فاصله همینگ اثرانگشت آنها از متن d، کمتر از k باشد با متن d تقریباً یکسان خواهند بود.

به این ترتیب پارامترهای این روش m و k هستند که باید تنظیم شوند.

الگوریتم LSH هر متن را به یک اثرانگشت کوچک می‌نگارد. در [33] نشان داده شده است که یک اثرانگشت 64 بیتی، m=64، برای هر متن برای پیدا کردن تمام متون تقریباً یکسان در مجموعه تمام صفحات وب کافی است. این درحالی است که الگوریتم‌های shingling و super shingling به فضای بیشتری احتیاج دارند.

الگوریتم LSH مانند روش‌های shingling و super shingling از روش‌های تقریبی است و دارای هر دو نرخ خطای مثبت و منفی است.

5.5سیستم شناسایی متون تقریباً یکسان

همانطور که در بخش قبل بررسی شد هیچ یک از روش‌های مطرح وابسته به زبان خاصی نیست؛ بلکه با تنظیم پارامترهای وابسته به زبان در هریک می‌توان بدون تغییر هر یک را برای زبان های متفاوتی به کار برد.

در طراحی سیستم شناسایی متون تقریباً یکسان همانند هر سیستم دیگری بیش از هر چیز، کاربر آن روش و هدف از طراحی سیستم اهمیت دارد. در مورد سیستم شناسایی متون تقریباً یکسان اشاره به نکات زیر می‌تواند مفید باشد.

1. اگر هدف طراحی سیستمی است که تمام تقریباً یکسان‌های یک متن را شناسایی کند، استفاده از یکی از روش‌های دقیق توصیه می‌شود.
2. در استفاده از روش‌های دقیق باید توجه داشت که این روش‌ها کندتر از روش‌های تقریبی هستند و در صورت استفاده از روش‌های فیلترینگ نیاز به نگهداری کل مجموعه متن هم می‌باشد.

3. در صورت نیاز به سیستمی که در مجموعه بزرگی از متن بتواند تقریباً یکسان‌های یک متن را تشخیص دهد، استفاده از روش‌های تقریبی مثل Shingling، super shingling و LSH توصیه می‌شود.
4. در استفاده از روش‌های تقریبی باید به این نکته توجه داشت که این روش‌ها دارای هر دو نوع خطای مثبت و منفی هستند.

در تمام روش‌های مطرح این قابلیت وجود دارد که مجموعه متون با ورد متن جدید به روز رسانی شود.

مراجع

[1] A.Broder, N.Eiron, M.Fontoura, M.Herscovici, R.Lempel, J.Mcpherson and E.Shekita, "Indexing Shared Content in Information Retrieval Systems", Proc. 10th Int. Conf Extending Database Technology, 2006, pp. 313-330.

[2] T.C.Hoad and J.Zobel, "Methods for Identifying Versioned and Plagiarised Documents", the American Society for Information Science and Technology, 54(3), 2003, pp. 203-215.

[3] M.Henzinger, "Finding Near-Duplicate Web Pages: a Large-Scale Evaluation of Algorithms", Proc. 29th Int. ACM SIGIR Conf. Research and development in information retrieval, 2006, pp. 284-291.

[4] S.YE, J.WEN and W.MA , "A Systematic Study of Parameter Correlations in Large Scale Duplicate Document Detection" , Proc. 10th Pacific-Asia Conf. Knowledge Discovery and Data Mining, 2006, pp. 275-284.

[5] J.G.Conrad and C.P.Schriber, "Constructing a text corpus for inexact duplicate detection" , Proc. 27th Int. ACM SIGIR Conf. Research and development in information retrieval, 2004, pp. 582-583.

[6] M.Potthast and B.Stein, "New Issues in Near-duplicate Detection", Data Analysis,Machine Learning and Applications: Proceedings of the 31st Annual Conference, 2007 (Studies in Classification, Data Analysis, and Knowledge Organization) ,2008 , pp. 601-609.

[7] S.Chakrabarti, Mining the Web: Discovering Knowledge from Hypertext Data, Morgan Kaufmann Publishers, San Francisco, CA, 2003.

[8] D.Fetterly, M.Manasse, M.Najork, "On the Evolution of Clusters of Near-Duplicate Web Pages", Proc. 1th Conf. Latin American Web Congress, 2003, pp. 37-45.

[9] A.R.Pereira, N.Ziviani, "Syntactic Similarity of Web Documents", Proc. of the 1th Conf. Latin American Web Congress, 2003, pp. 194-200.

[10] F.Culein, A.Macleod and T.Lancaster, "Source Code Plagiarism in UK HEComputing Schools , Issues, Attitudes and Tools", Technical Report , South Bank University, 2001.

[11] B.Stein, "Fuzzy-Fingerprints for Text-based Information Retrieval", Proc. 5th Int. Conf. Knowledge Management, 2005, pp. 572-579.

[12] J.Poster, RFC Collection, 2004 , http://www.rfc-editor.org.

[13] G.Aanar , The Linux Documentation Project , 2004, http://www.tldp.org.

[14] Y.Bernstein, M.Shokouhi and J.Zobel, "Compact features for detection of near duplicates in distributed retrieval", Proc. 13th Inte. Conf. String Processing and Information Retrieval Symp. ,2006 ,pp. 110-121.

[15] W.Pugh and M.Henzinger, "Detecting duplicate and near-duplicate files", (United States Patent 6,658,423), 2003.

[16] A.Kolcz, A.Chowdhury and J.Alspector ,"Improved robustness of signature-based near-replica detection via lexicon randomization", Proc. ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, 2004 , pp. 605–610.

[17] U.Manber, "Finding similar files in a large file system", Proc. USENIX Winter 1994 Technical Conference, 1994, pp. 1–10.

[18] C.Lyon, J.Malcolm and B.Dickerson, "Detecting short passages of similar text in large document collections", Proc. 2001 Conf. Empirical Methods in Natural Language Processing, 2001, pp. 118-125.

[19] J.Conrad, X.Guo and C.Schriber, "Online duplicate document detection:Signature reliability in a dynamic retrieval environment" , Proc. 12th

ACM international Conf. Information and knowledge management, 2003, pp. 443–452.

[20] S.Brin, J.Davis, and H.Garc´ia-Molina, "Copy detection mechanisms for digital documents", Proc.ACM Int. Conf. Management of Data, 1995, pp. 398–409.

[21] Y.Bernstein and J.Zobel**,** "Redundant documents and search effectiveness", Proc. 14th ACM Int. Conf. Information and knowledge management,2005, pp. 736–743.

[22] A.Broder,S.Glassman, M.Manasse and G.Zweig, "Syntactic clustering of the web",Computer Networks and ISDN Systems, 29(8-13), 1997, pp. 1157-1166.

[23] A.Broder, "On the resemblance and containment of documents", Proc. Compression and Complexity of Sequences, 1997 , pp. 21–29.

[24] M.Shivakumar and M.Garcia-Molina, "SCAM: a copy detection mechanism for digital documents"**,** Proc. 2nd Annual Conference on the Theory and Practice of Digital Libraries, 1995.

[25] J.Zobel, and Y.Bernstein, "The case of the duplicate documents: Measurement, search, and science", Proc. Asia-Pacific Web Conference, 2006, pp. 26–39.

[26] A.Chowdhury, O.Frieder, D.Grossman and M.McCabe,″ Collection statistics for fast duplicate document detection", ACM Transactions on Information Systems, 20(2), 2002 , pp. 171-191.

[27] G.Forman, K.Eshghi, S.Chicochetti, "Finding Similar Files in Large Document Repositories", ACM Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining, 2005, pp. 21-25.

[28] H.Yang, J.Callan, "Near-Duplicate Detection by Instance-level Constrained Clustering", Proc. 29th Int. ACM SIGIR Conf. Research and development in information retrieval, 2006, pp. 421-428.

[29] Y.Bernstein, and J.Zobel, "A scalable system for identifying co-derivative documents", Proc. String Processing and Information Retrieval Symp., 2004, pp. 55-67.

[30] A.Broder, "Identifying and filtering near-duplicate documents", Proc. 11th Symp. Combinatorial Pattern Matching, 2000, pp. 1–10.

[31] M.Charikar, "Similarity estimation techniques from rounding algorithms",Proc. 34th ACM Symp. Theory of computing, 2002, pp. 380-388.

[32] S.Schleimer, D.Wilkerson, A.Aiken, "Winnowing: Local Algorithms for document Fingerprinting", Proc. 2003 ACM SIGMOD Int. Conf. Management of Data, 2003, pp. 76-85.

[33] G.S.Manku, J.Arvind, D.D.Sarma, "Detecting Near-Duplicates for Web Crawling", Proc. 16th Int. Conf. World Wide Web, 2007, pp. 141-150.

[34] A.Kołcz, A.Chowdhury, "Lexicon randomization for near-duplicate detection with I-Match", Supercomputing, 45(3), 2008, pp.255-276.

[35] S.Sarawagi, and A.Kirpal, "Efficient set joins on similarity predicates", Proc. 2004 ACM SIGMOD Int. Conf. Management of data, 2004, pp. 743-754.

[36] R.J.Bayardo, Y.Ma, and R.Srikant, "Scaling up all pairs similarity search", Proc. 16th Int. Conf. World Wide Web, 2007, pp. 131-140.

[37] C.Xiao, W.Wang, X.Lin and J.Yu, "Efficient Similarity Joins for Near Duplicate Detection", Proc. 17th Int. Conf. World Wide Web, 2008, pp. 131-140.

[38] J.Gibsom, B.Wallner, and S.Lubar, "Identification of Duplicate News Stories in Web Pages", Proc. 4th Web as a Corpus Workshop, 2008.

[39] S. Tachaphetpiboon, N. Facundes and T. Amornraksa , "Plagiarism Indication by Syntactic-Semantic Analysis" , Proc. 13th Asia-Pacific Conf. on Communications (APCC2007) , 2007, pp.237-240.

[40] M. Mozgovoy, V. Tusov, V. Klyuev, “The Use of Machine Semantic Analysis in Plagiarism Detection”, Proceedings of the 9th International Conference on Humans and Computers, Japan, 2006, p. 72-77.

[41] D.C.Tran and T.C.Tran , "Copy Detection Using Latent Semantic Similarity", IEEE Conf. on Research, Innovation and Vision for the future(RIVF2008) , 2008 , Vietnam.