Introduction to

Information Retrieval

لغت نامه‌ها و بازیابی مقاوم

علی قنبری سرخی

تفاوت نوع (type) / نشانه (token)

- نشانه: نمونه‌ای از دنباله کاراکترها (Word یا Term) در اسناد است که با هم به عنوان یک واحد معنایی برای پردازش
- نوع: دسته تمامی نشانه‌هاست که دارای دنباله کاراکتر یکسان هستند.

- مثال:
  - In June, the dog likes to chase the cat in the barn.
  - 12 word tokens, 9 word types

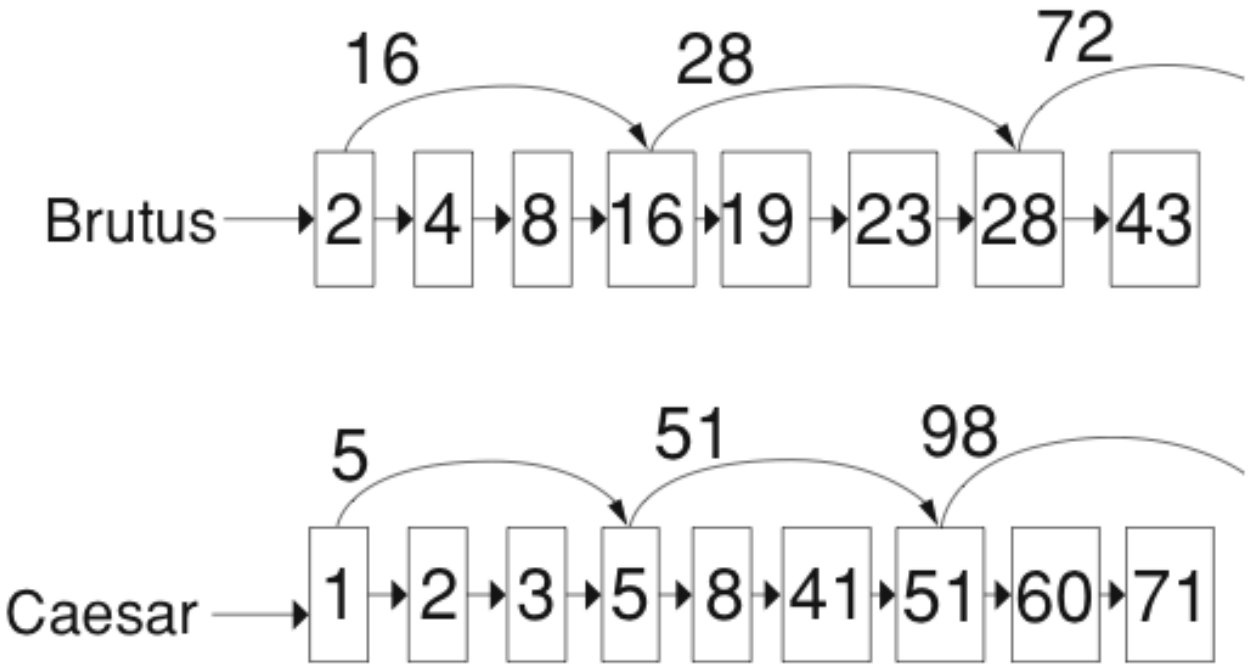
مشکلات نشانه‌گذاری

- جداکننده‌ها (delimiters) چه هستند؟ فضا خالی؟ آپوستروف؟ خط ربط؟
- در این موارد، بعضی از مواقع آنها را محدود می‌کنند. ولی در بعضی از موارد نمی‌توان محدود کرد.
  - در بعضی از زبان‌ها فاصله ندارند! (برای نمونه زبان چینی)
  - کلمات مرکب که فاصله ندارند (برای نمونه زبان‌های آلمانی، هلندی و سوئدی)
  - مثال کلمات مرکب:
    - (*Lebensversicherungsgesellschaftsangestellter*)

مشکلات دسته‌بندی هم‌ارزی

- یک عبارت، یک دسته هم‌ارز از نشانه‌ها است.
- چگونه ما می‌توانیم یک کلاس هم‌ارز را تعریف کنیم؟
- شماره‌ها (3/20/91 vs. 20/3/91)
- نوشتن با حروف بزرگ/غیر حساس کردن به حروف کوچک و بزرگ
- ریشه‌گیری، ریشه‌گیر Porter
- تحلیل ریخت‌شناسی: inflectional vs. Derivational
- مسائل دسته‌بندی هم‌ارزی در زباهای دیگر
  - ریخت‌شناسی بسیار پیچیده‌تر نسبت به زبان انگلیسی
    - فلاندی: یک فعل تنها می‌تواند ۱۲۰۰۰ فرم متفاوت داشته باشد.
    - لهجه، اعراب

اشاره‌گر پرش



شاخص‌های موقعیتی

- لیست‌های پست‌ها در شاخص بدون موقعیت: هر پست فقط یک شماره سند (docID) است.
- لیست‌های پست‌ها در شاخص موقعیتی: هر پست یک شماره سند (docID) و یک لیست از موقعیت‌ها است.
- پرس و جوی نمونه "$to_1\ be_2\ or_3\ not_4\ to_5\ be_6$"

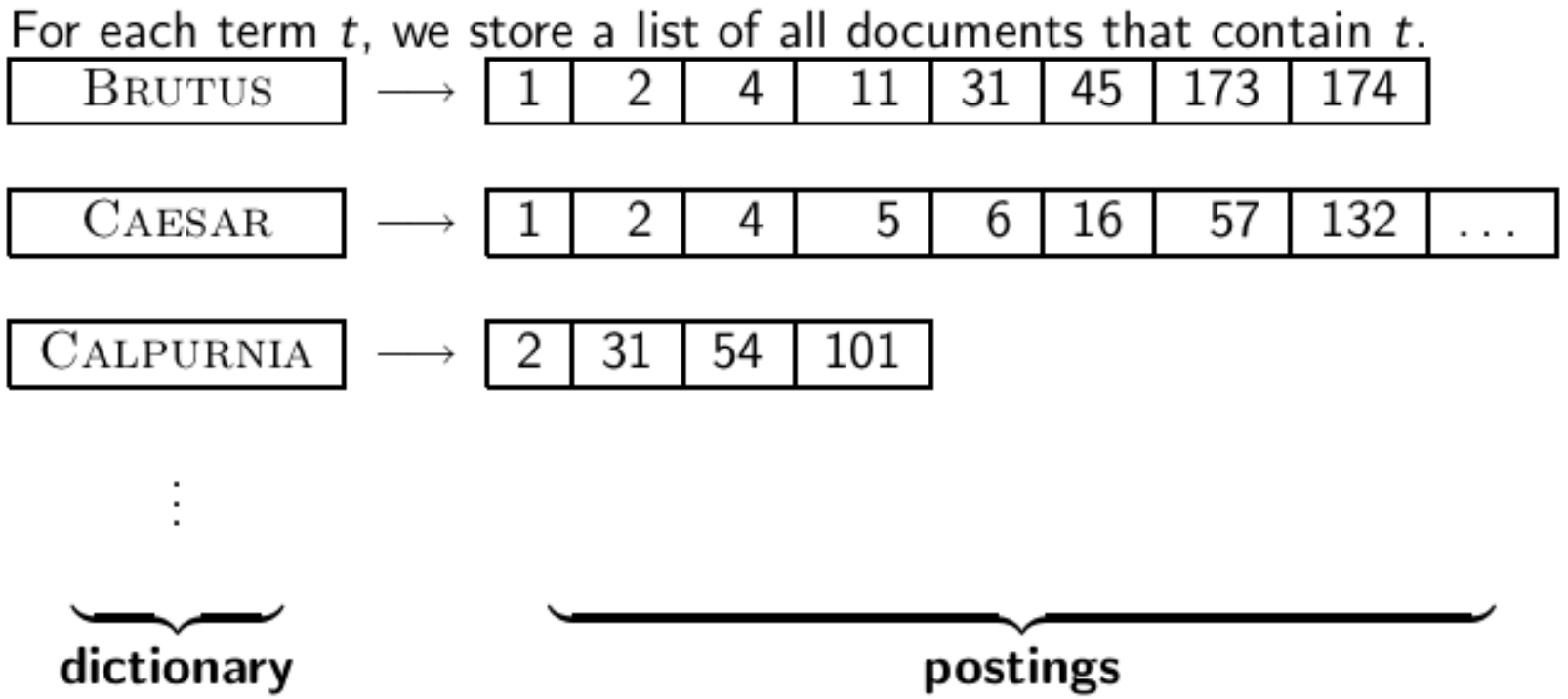
TO, 993427:
‹ 1: ‹7, 18, 33, 72, 86, 231›;
2: ‹1, 17, 74, 222, 255›;
4: ‹8, 16, 190, 429, 433›;
5: ‹363, 367›;
7: ‹13, 23, 191›; . . . ›
BE, 178239:
‹ 1: ‹17, 25›;
4: ‹17, 191, 291, 430, 434›;
5: ‹14, 19, 101›; . . . › Document 4 is a match!

- با شاخص موقعیت، ما می‌توانیم به پرس و جوهای اصطلاح پاسخ دهد.
- با شاخص موقعیت، ما می‌توانیم به پرس و جوهای مجاورت پاسخ دهد.

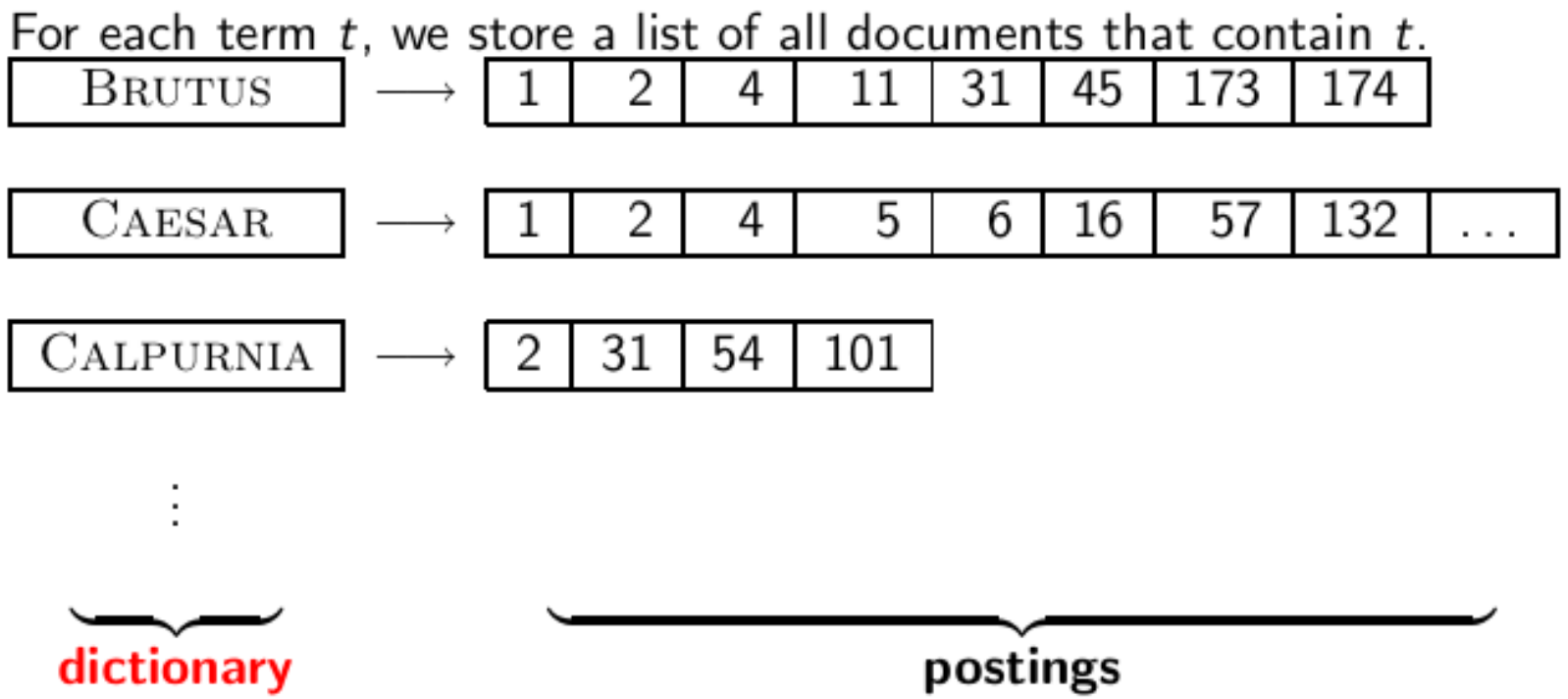
Take-away

- **بازیابی مقاوم:** در صورتی که انطباق دقیق بین عبارت پرس و جو و عبارت استادوجود نداشته باشد چه کاری باید کرد:
  - پرس و جوهای جایگزین
  - تصحیح املایی

شاخص وارونه


For each term t, we store a list of all documents that contain t.
BRUTUS
1 2 4 11 31 45 173 174
CAESAR
1 2 4 5 6 16 57 132 . . .
CALPURNIA
2 31 54 101
dictionary
postings

شاخص وارونه



10

دیکشنری‌ها (لغت نامه ها)

- لغت نامه، ساختار داده برای ذخیره واژگان عبارت است.
  - واژگان عبارت: داده
  - لغت نامه: ساختار داده برای ذخیره واژگان داده است.

لغت نامه به عنوان آرایه‌ای از ورودی‌های با طول ثابت

- برای هر عبارت، ما نیاز به ذخیره یک جفت عبارت داریم:
  - تعداد تکرار اسناد
  - اشاره‌گر به لیست پست‌ها

- فرض کنید که در حال حاضر ما می‌توانیم این اطلاعات را در ورودی با طول ثابت ذخیره کنیم.
- فرض کنید ما این ورودی‌ها را در آرایه ذخیره می‌کنیم.

لغت نامه به عنوان آرایه‌ای از ورودی‌های با طول ثابت

term	document frequency	pointer to postings list
a	656,265	⟶
aachen	65	⟶
. . .	. . .	. . .
zulu	221	⟶

space needed: 20 bytes 4 bytes 4 bytes

- چطور ما یک عبارت پرس و جو q_i را در این آرایه در مرحله پرس و جو جستجو می‌کنیم؟
- به این معنا که: ما چه ساختار داده‌ای برای قرار دادن ورودی (سطر) در آرایه که q_i ذخیره شده، استفاده می‌کنیم

13

ساختار داده‌ها برای جستجو عبارت

- دو کلاس اصلی ساختار داده‌ها: درخت‌ها و درهم سازی‌ها
- بعضی از سیستم‌های بازیابی اطلاعات از درهم سازی و بعضی‌ها از درخت‌ها استفاده می‌کنند
- معیارهای استفاده از درهم سازی‌ها در مقابل درخت‌ها
  - چه تعداد کلید خواهیم داشت؟
  - آیا این تعداد ثابت باقی می‌ماند یا به شدت تغییر می‌کند- و در مورد تغییرات، تنها کلیدهای جدید درج شده را داریم، یا برخی از کلیدها در لغت‌نامه باید حذف شوند؟
  - فراوانی نسبی دسترسی به کلیدهای مختلف چیست؟

درهم سازی‌ها

- هر عبارت مجموعه واژگان (کلید) به یک عدد صحیح روی فضایی که به اندازه کافی بزرگ است درهم سازی می‌شود.
- سعی می‌شود از تصادم اجتناب شود.
- در زمان پرس و جو مراحل زیر انجام می شود
  - درهم سازی عبارت پرس و جو
  - حل تصادم ها
  - قرار دادن ورودی در آرایه با طول ثابت

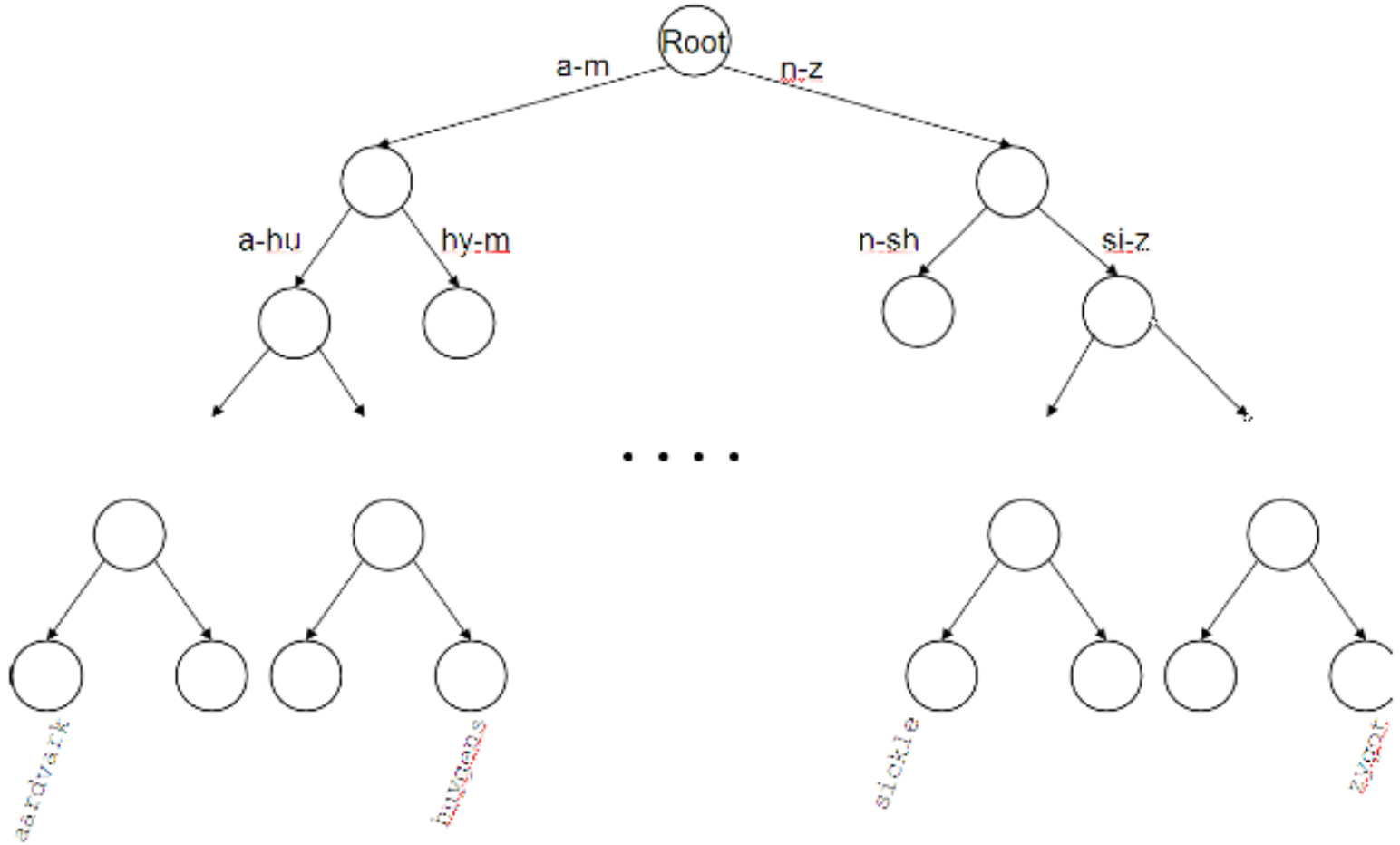
درهم سازی‌ها

- نقاط قوت: جستجو در درهم‌سازی سریع‌تر از جستجو در درخت می‌باشد
  - زمان جستجو ثابت است
- نقاط ضعف:
  - هیچ راه آسانی برای پیدا کردن تغیرات کوچک وجود ندارد (برای نمونه عبارات لهجه دار و بی‌لهجه .vs *resume* *résumé*)
  - جستجو پیشوندی انجام نمی‌شود (برای نمونه تمام عباراتی که با automat شروع می شود را نمی‌توان جستجو کرد)
  - در صورتی که عبارت واژگان در حال رشد کردن باشد نیاز به دوباره درهم‌سازی برای هر چیز به صورت دوره‌ای می‌باشد. (مثلا وب)

درخت‌ها

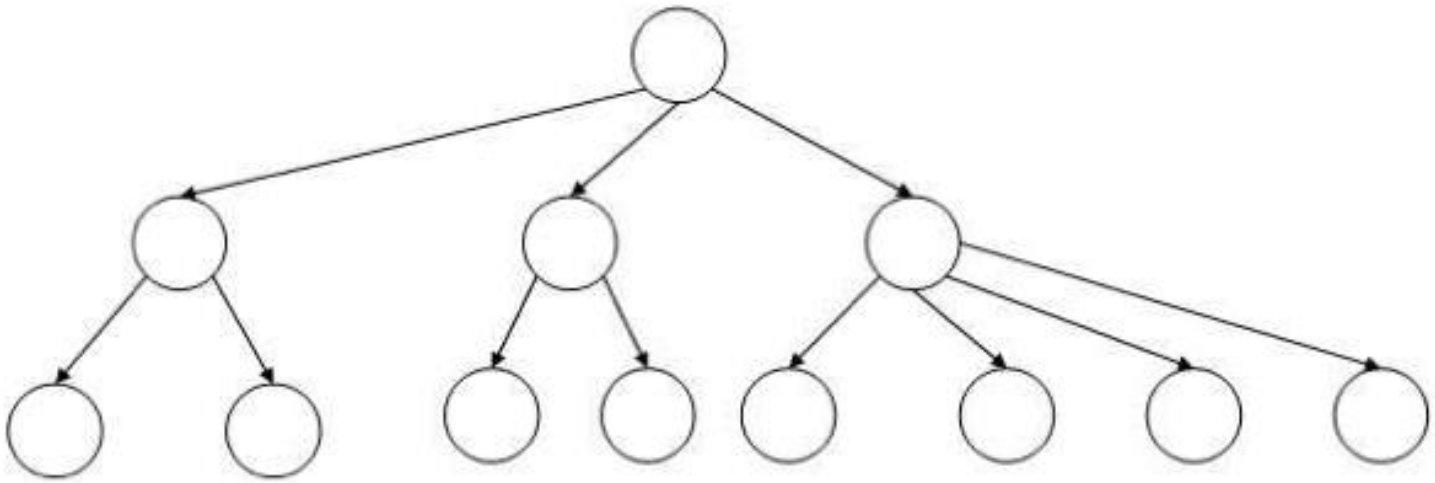
- درخت ها مشکلات پیشوندی را حل کردند (پیدا کردن همه ی عبارات که با automat شروع می شوند)
- ساده‌ترین درخت: درخت باینری
- جستجو کمی آهسته‌تر از درهم‌سازی ها هست: هزینه جستجو O(logM)، در اینجا M اندازه واژگان می‌باشد.
- هزینه O(logM) فقط برای درخت های متوازن می‌باشد
- توازن مجدد درخت‌های باینری بسیار پرهزینه می‌باشد.
- B-trees، مسئله توازن مجدد را کاهش می‌دهند.
- تعریف B-tree:یک درخت جستجو است که در آن هر گره داخلی دارای تعدادی فرزند در بازه [a,b] است. که در آن a و b اعداد صحیح مثبت هستند. برای نمونه درخت باینری .[2, 4]

درخت باینری



18

B-tree



19

پرس و جوهای جایگزین

- *mon: پیدا کردن همه‌ی اسنادی که شامل، عبارتی هستن که با mon شروع می‌شوند.
- به راحتی با لغت‌نامه B-tree انجام می‌شود: همه‌ی عباراتی به مانند t، که در بازه $mon \leq t < moo$ هستند بازیابی می‌شوند.
- mon*: پیدا کردن همه‌ی اسنادی که شامل، عبارتی هستن که به mon ختم می‌شوند.
  - حفظ یک درخت اضافی برای عبارات عقب‌گرد
  - سپس همه‌ی عبارات t که در بازه $nom \leq t < non$ هستن بازیابی می‌شوند
- نتیجه: یک مجموعه از عبارات که با پرس و جوی جایگزین انطباق دارند
- سپس اسنادی که شامل هر یک از این عبارات هستند بازیابی می‌شوند.

کنترل * در وسط عبارات

- نمونه: m*nchen
- ما می‌توانیم *m و *nchen را در B-tree جستجو نماییم و مجموعه‌های دو عبارت را اشتراک بگیریم
- بسیار پر هزینه می‌باشد.
- جایگزین: شاخص جایگردانی
- ایده پایه: چرخش هر پرس و جو جایگزین، بنابراین * در انتها اتفاق می افتد.
- هریم از این چرخش‌ها در لغت‌ناهمه (B-tree) ذخیره می‌شود.

شاخص جایگردانی

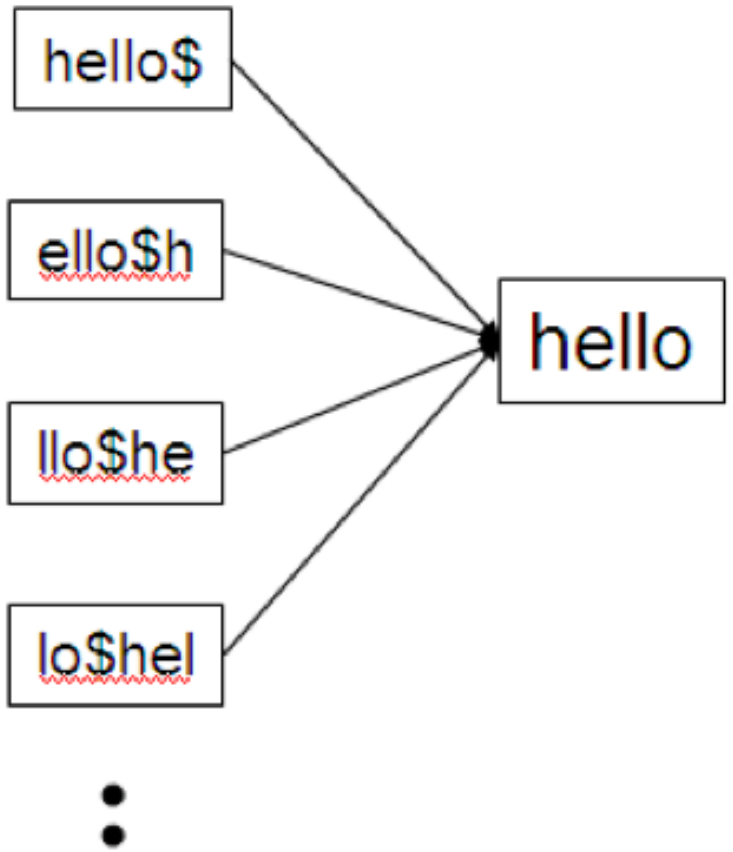
- برای عبارت HELLO: عبارات

hello$, ello$h, llohe, lohel, and o$hell

به B-tree اضافه می‌شوند.

- در اینجا $ یک نماد ویژه است

Permuterm → term mapping



شاخص جایگردانی

- برای عبارت HELLO: ما عبارات hello$, ello$h, llohe, lohel, and o$hell را ذخیره کرده‌ایم.
- پرس و جوها:
  - For X, look up X$
  - For X*, look up X*$
  - For *X, look up X$*
  - For *X*, look up X*
  - For X*Y, look up Y$X*
  - Example: For hel*o, look up o$hel*
- بهتر است بجای استفاده از شاخص جایگردانی از درخت جایگردانی استفاده شود. ولی شاخص جایگزینی رایج‌تر است.
- شاخص جایگردانی به ما امکان می‌دهد دوران‌هایی با پیشوند مشخص را انتخاب کنیم.

جستجوکردن در شاخص جایگردانی

- چرخش پرس و جو جایگزین به راست
- استفاده از جستجو B-tree به مانند قبل
- مشکل: لغت نامه جایگزینی بسیار گسترده شده است و شامل تمامی دوران‌های هر عبارات است.
- بیش از ۴ برابر اندازه لغت نامه در مقایسه B-tree (عدد تجربی)

شاخص‌های k–گرمی

- این شاخص بسیار کارامدتر از شاخص جایگردانی است.
- لغت نامه شامل تمام کارکترهای k-گرم‌هایی است که در هر عبارت مجموعه واژگان رخ می‌دهد (یک k-گرمی دنباله‌ای از k کاراکتر است).
- ۲-گرمی به اصطلاح bigrams گفته می‌شود.
- مثال:
  - ”April is the cruelest month”
    - bigrams: $a ap pr ri il l$ $i is s$ $t th he e$ $c cr ru ue el le es st t$ $m mo on nt h$
  - $یک نماد مرزی ویژه به مانند قبل می‌باشد.
  - یک شاخص وارونه از ۲-گرمی‌ها برای عباراتی که شامل ۲-گرم هستند را حفظ می‌کند.
  - .

لیست پست‌ها در یک شاخص وارونه ۳–گرمی

فرآیند عبارت‌های جایگزین در شاخص bigram

- حالا پرس وجوی*mon می‌تواند به عنوان $m AND mo AND on اجرا شود.
- تمام عباراتی که با پیشوند mon می‌باشند را در اختیبار ما قرار می‌دهد.
  - اما دارای "مثبت کاذب" به مانند Moon می‌باشد.
- ما باید این عبارات را در برابر پرس و جو پس فیلتر (postfilter) کنیم
- سپس عبارات باقی مانده (زنده مانده) در شاخص وارونه شده عبارت-سند جستجو می شود.
- شاخص k-گرمی در مقابل شاخص جایگزینی
  - شاخص k-گرمی بسیار از لحاظ فضا کارامدتر است.
  - شاخص جایگزینی به پسافیلتر نیازی ندارد.

تمرین

- Google یک حمایت محدود برای پرس و جوهای جایگزینی دارد.
  - برای مثال، همچین پرس و جویی به مانند [gen* universit*] عملکرد خیلی خوبی در Google ندارند.
  - هدف: شما می‌خواهید عباراتی به مانند the University of Geneva را جستجو نمائید اما نمی‌دانیم کدام لهجه برای استفاده از کلمات فرانسوی برای عبارت university and Geneva استفاده می‌شود.
- بر اساس جستجو Google در سال 2010-04-29: "توجه داشته باشید که عملگر * برای کل کلمات نه قسمتی از کلمات عمل می‌کند "
- اما این مورد به صورت کاملا درست نمی باشد. تلاش کنید عبارات [pythag*] and [m*nchen]
- تمرین: چرا Google به صورت کامل از پرس و جوهای جایگزینی حمایت نمی‌کند؟

فرآیند پرس و جوهای جایگزینی در شاخص عبارت–سند

- مسئله ۱:ما باید به صورت بالقوه یک تعداد زیادی از پرس و جوهای بولی را اجرا نمائیم.
- Most straightforward semantics: Conjunction of disjunctions
- For [gen* universit*]: geneva university OR geneva université OR genève university OR genève université OR general universities OR . . .
  - بسیار پر هزینه
  - مسئله ۲: کاربران نسبت به نوع نفرت دارند.
  - اگر پرس و جوی به صورت اختصار وجود داشته باشد برای مثال اجازه داشته باشید از [pyth* theo*] برای [pythagoras’ theorem] استفاده نمائید بسیار از کاربران از آنها استفاده خواهند کرد.
  - این عمل هزینه پاسخ‌دهی به پرس و جوها را به صورت قابل توجه افزایش می‌دهد.
  - تا حدودی توسط پیشنهادات گوگل رفع می‌شود.

تصحیح املایی

- دو اصل استفاده می‌شود:
  - اسنادی که شاخص‌گذاری شده تصحیح شوند.
  - پرس و جوهای کاربران تصحیح شود.
- دو روش متفاوت برای تصحیح املایی وجود دارد:
  - تصحیح املایی عبارت مجزا
    - هر عبارت را به صورت غلط املایی چک کنید
  - تلاش می کنیم تا یک عبارت واحد پرس و جو را تصحیح کنیم- حتی زمانیکه پرس و جو چند عبارتی داشته باشیم.
    برای مثال: an asteroid that fell form the sky
  - تصحیح املایی حساس به متن
    - به عبارات پیرامون نگاه کن
    - در این صورت می‌توان خطای بالا form/from را تصحیح کرد.

تصحیح اسناد

- در این کلاس، ما علاقه‌ای به تعامل تصحیح خطا اسناد (برای مثال MS Word) را نداریم.
- در بازیابی اطلاعات، ما تصحیح اسناد اصلی برای اسناد OCR شده را استفاده می‌کنیم.
  - منظور از OCR = optical character recognition

- فلسفه اصلی در IR :
  - تغییر ندادن اسناد

تصحیح پرس و جو

- ابتدا: تصحیح املای عبارت مجزا
- فرض ۱: یک لیست از "عبارات صحیح" که در آن املایی درست آمده است.
- فرض ۲: یک راهی برای محاسبه فاصله بین عبارت اشتباه (با غلط املایی) و عبارت صحیح وجود دارد.
- ساده‌ترین الگوریتم تصحیح املایی، عبارت "صحیح" که کوچکترین فاصله به عبارت اشتباه دارد را بر می‌گرداند.

- Example: informaton → information
- ما می‌توانیم برای لیستی از عبارات صحیح، از واژگانی از تمام عباراتی که در مجموعه ما رخ می‌دهند استفاده نمائیم
- چرا این مشکل‌ساز است؟

استفاده کردن واژگان عبارت

- یک لغت نامه استاندارد (Webster’s, OED etc.)
- یک لغت نامه خاص منظور صنعتی (مثلا برای سیستم های بازیابی اطلاعات خاص منظوره شده اند)
- واژگان عبارت از مجموعه که به صورت صحیح وزندار شده اند.

فاصله بین کلمات اشتباه و واژه "صحیح"

- ما چندین گزینه را برای محاسبه فاصله مطالعه می‌کنیم.
  - فاصله ویرایشی و فاصله Levenshtein
  - فاصه ویرایش وزندار
  - k-gram overlap

فاصله ویرایشی

- فاصه ویرایشی بین دو رشته S1 و S2
  - حداقل تعداد عملیات ویرایشی مورد نیاز برای تبدیل s1 به s2 است.
- فاصله Levenshtein
  - عملیات ویرایش قابل قبول، درج (insert)، حذف (delete) و جایگزینی (replace) است.
  - Levenshtein distance *dog-do*: 1
  - Levenshtein distance *cat-cart*: 1
  - Levenshtein distance *cat-cut*: 1
  - Levenshtein distance *cat-act*: 2
- Damerau-Levenshtein شامل انتقال (transposition) به عنوان چهارمین عملیات ویرایشی قابل قبول می‌باشد.
  - Damerau-Levenshtein distance cat-act: 1

محاسبه فاصله Levenshtein

		f	a	s	t
	0	1	2	3	4
c	1	1	2	3	4
a	2	2	1	2	3
t	3	3	2	2	2
s	4	4	3	2	3

الگوریتم فاصله Levenshtein

```
LevenshteinDistance(s1, s2)
 1  for i ← 0 to |s1|
 2  do m[i, 0] = i
 3  for j ← 0 to |s2|
 4  do m[0, j] = j
 5  for i ← 1 to |s1|
 6  do for j ← 1 to |s2|
 7       do if s1[i] = s2[j]
 8            then m[i, j] = min{m[i-1, j]+1, m[i, j-1]+1, m[i-1, j-1]}
 9            else  m[i, j] = min{m[i-1, j]+1, m[i, j-1]+1, m[i-1, j-1]+1}
10  return m[|s1|, |s2|]
```

Operations: insert (cost 1), delete (cost 1), replace (cost 1), copy (cost 0)

الگوریتم فاصله Levenshtein

```
LevenshteinDistance(s1, s2)
 1  for i ← 0 to |s1|
 2  do m[i, 0] = i
 3  for j ← 0 to |s2|
 4  do m[0, j] = j
 5  for i ← 1 to |s1|
 6  do for j ← 1 to |s2|
 7       do if s1[i] = s2[j]
 8            then m[i, j] = min{m[i-1, j]+1, m[i, j-1]+1, m[i-1, j-1]}
 9            else  m[i, j] = min{m[i-1, j]+1, m[i, j-1]+1, m[i-1, j-1]+1}
10  return m[|s1|, |s2|]
```

Operations: insert (cost 1), delete (cost 1), replace (cost 1), copy (cost 0)

الگوریتم فاصله Levenshtein

```
LevenshteinDistance(s1, s2)
 1  for i ← 0 to |s1|
 2  do m[i, 0] = i
 3  for j ← 0 to |s2|
 4  do m[0, j] = j
 5  for i ← 1 to |s1|
 6  do for j ← 1 to |s2|
 7       do if s1[i] = s2[j]
 8            then m[i, j] = min{m[i-1, j]+1, m[i, j-1]+1, m[i-1, j-1]}
 9            else  m[i, j] = min{m[i-1, j]+1, m[i, j-1]+1, m[i-1, j-1]+1}
10  return m[|s1|, |s2|]
```

Operations: insert (cost 1), delete (cost 1), replace (cost 1), copy (cost 0)

الگوریتم فاصله Levenshtein

```
LevenshteinDistance(s1, s2)
 1  for i ← 0 to |s1|
 2  do m[i, 0] = i
 3  for j ← 0 to |s2|
 4  do m[0, j] = j
 5  for i ← 1 to |s1|
 6  do for j ← 1 to |s2|
 7       do if s1[i] = s2[j]
 8            then m[i, j] = min{m[i-1, j]+1, m[i, j-1]+1, m[i-1, j-1]}
 9            else  m[i, j] = min{m[i-1, j]+1, m[i, j-1]+1, m[i-1, j-1]+1}
10  return m[|s1|, |s2|]
```

Operations: insert (cost 1), delete (cost 1), replace (cost 1), copy (cost 0)

الگوریتم فاصله Levenshtein

```
LevenshteinDistance(s1, s2)
 1  for i ← 0 to |s1|
 2  do m[i, 0] = i
 3  for j ← 0 to |s2|
 4  do m[0, j] = j
 5  for i ← 1 to |s1|
 6  do for j ← 1 to |s2|
 7       do if s1[i] = s2[j]
 8            then m[i, j] = min{m[i-1, j]+1, m[i, j-1]+1, m[i-1, j-1]}
 9            else  m[i, j] = min{m[i-1, j]+1, m[i, j-1]+1, m[i-1, j-1]+1}
10  return m[|s1|, |s2|]
```

Operations: insert (cost 1), delete (cost 1), replace (cost 1), copy (cost 0)

مثال فاصله Levenshtein

		f		a		s		t	
	0	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
c	**1** **1**	*1* *2*	*2* *1*	**2** 3 **2** **2**		**3** 4 **3** **3**		**4** 5 **4** **4**	
a	**2** **2**	**2** **2** 3 **2**		*1* *3* *3* *1*		*3* *4* *2* *2*		4 5 **3** **3**	
t	**3** **3**	**3** **3** 4 **3**		3 **2** 4 **2**		*2* *3* *3* *2*		*2* *4* *3* *2*	
s	**4** **4**	**4** **4** 5 **4**		4 **3** 5 **3**		**2** 3 4 **2**		*3* *3* *3* *3*	

43

هر سلول ماتریس Levenshtein

هزینه دریافت از همسایه بالایی (delete)	هزینه دریافتی از همسایه بالا سمت چپ (copy or replace)
کمینه سه عنصر این ماتریس؛ ارزانترین راه برای رسیدن به اینجا	هزینه دریافت از همسایه چپ (insert)

مثال فاصله Levenshtein

		f		a		s		t	
	0	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
c	**1** **1**	*1* *2* *2* *1*		**2** 3 **2** **2**		**3** 4 **3** **3**		**4** 5 **4** **4**	
a	**2** **2**	**2** **2** 3 **2**		*1* *3* *3* *1*		*3* *4* *2* *2*		4 5 **3** **3**	
t	**3** **3**	**3** **3** 4 **3**		3 **2** 4 **2**		*2* *3* *3* *2*		*2* *4* *3* *2*	
s	**4** **4**	**4** **4** 5 **4**		4 **3** 5 **3**		**2** 3 4 **2**		*3* *3* *3* *3*	

45

فاصله ویرایشی وزندار

- وزن یک عملیات بستگی به کارکترهای درگیر دارد.
- برای مدیریت کردن خطاهای کیبورد. برای نمونه: احتمال اینکه m به صورت اشتباه n تایپ شود بیشتر از q می‌باشد.
- بنابراین، جایگزینی (replacing)، m به وسیله n یک فاصله ویرایشی کوچکتر نسبت به m به وسیله q دارد.
- در اینجا ما به یک ماتریش وزندار به عنوان ورودی نیاز داریم.
- برنامه نویسی پویا برای کنترل وزن‌ها اصلاح می‌شود.

استفاده از فاصله ویرایشی تصحیح املایی

- با توجه به پرس‌وجو:
  - ابتدا تمام توالی‌های کارکترها را در یک فاصله ویرایشی از پیش تعیین شده (احتمالا وزندار) ثبت کنید
  - اشتراک این مجموعه را با لیست کلمات "صحیح" خود بدست آورید.
  - سپس عبارتی را از این اشتراک‌گیری به کاربر پیشنهاد دهید.

تمرین

- ماتریش فاصله Levenshtein را برای OSLO – SNOW محاسبه کنید؟
- چرا عملیات ویرایشی Levenshtein عبارت cat را به catcat تبدیل می‌کند؟

		s	n	o	w
	0	1 1	2 2	3 3	4 4
o	1 / 1				
s	2 / 2				
l	3 / 3				
o	4 / 4				

49

		s		n		o		w	
	0	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
o	**1** / **1**	1 / 2	2 / ?						
s	**2** / **2**								
l	**3** / **3**								
o	**4** / **4**								

50

		s	n	o	w
	0	1 1	2 2	3 3	4 4
o	1 / 1	1 2 / 2 1			
s	2 / 2				
l	3 / 3				
o	4 / 4				

51

		s	n	o	w
	0	1 1	2 2	3 3	4 4
o	1 / 1	1 2 / 2 1	2 3 / 2 ?		
s	2 / 2				
l	3 / 3				
o	4 / 4				

52

		s		n		o		w	
	0	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
o	**1** **1**	**1** 2	2 **1**	**2** **2**	3 **2**				
s	**2** **2**								
l	**3** **3**								
o	**4** **4**								

53

		s	n	o	w
	0	**1** **1**	**2** **2**	**3** **3**	**4** **4**
o	**1** / **1**	**1** 2 / 2 **1**	**2** 3 / **2** **2**	2 4 / 3 ?	
s	**2** / **2**				
l	**3** / **3**				
o	**4** / **4**				

54

		s	n	o	w
	0	1 1	2 2	3 3	4 4
o	1 / 1	1 2 / 2 1	2 3 / 2 2	2 4 / 3 2	
s	2 / 2				
l	3 / 3				
o	4 / 4				

55

		s		n		o		w	
	0	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
o	**1** / **1**	**1** / 2	2 / **1**	**2** / **2**	3 / **2**	**2** / 3	4 / **2**	4 / 3	5 / ?
s	**2** / **2**								
l	**3** / **3**								
o	**4** / **4**								

56

		s		n		o		w	
	0	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
o	**1** **1**	**1** 2	2 **1**	**2** **2**	3 **2**	**2** 3	4 **2**	4 **3**	5 **3**
s	**2** **2**								
l	**3** **3**								
o	**4** **4**								

57

		s		n		o		w	
	0	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
o	**1**	**1**	2	**2**	3	**2**	4	4	5
	1	2	**1**	**2**	**2**	3	**2**	**3**	**3**
s	**2**	1	2						
	2	3	?						
l	**3**								
	3								
o	**4**								
	4								

58

		s		n		o		w	
	0	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
o	**1** / **1**	**1** / 2	2 / **1**	**2** / 3	3 / **2**	**2** / 4	4 / **2**	4 / **3**	5 / **3**
s	**2** / **2**	**1** / 3	2 / **1**						
l	**3** / **3**								
o	**4** / **4**								

59

		s	n	o	w
	0	**1** **1**	**2** **2**	**3** **3**	**4** **4**
o	**1** / **1**	**1** 2 / 2 **1**	**2** 3 / **2** **2**	**2** 4 / 3 **2**	4 5 / **3** **3**
s	**2** / **2**	**1** 2 / 3 **1**	2 3 / 2 ?		
l	**3** / **3**				
o	**4** / **4**				

60

		s		n		o		w	
	0	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
o	**1** / **1**	**1** / 2	2 / **1**	**2** / **2**	3 / **2**	**2** / 3	4 / **2**	4 / **3**	5 / **3**
s	**2** / **2**	**1** / 3	2 / **1**	**2** / **2**	3 / **2**				
l	**3** / **3**								
o	**4** / **4**								

61

		s		n		o		w	
	0	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
o	**1** **1**	**1** 2	2 **1**	**2** **2**	3 **2**	**2** 3	4 **2**	4 **3**	5 **3**
s	**2** **2**	**1** 3	2 **1**	**2** **2**	3 **2**	3 3	3 ?		
l	**3** **3**								
o	**4** **4**								

62

		s		n		o		w	
	0	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
o	**1** **1**	**1** 2	2 **1**	**2** **2**	3 **2**	**2** 3	4 **2**	4 **3**	5 **3**
s	**2** **2**	**1** 3	2 **1**	**2** **2**	3 **2**	**3** **3**	**3** **3**		
l	**3** **3**								
o	**4** **4**								

63

		s		n		o		w	
	0	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
o	**1** **1**	**1** 2	2 **1**	**2** **2**	3 **2**	**2** 3	4 **2**	4 **3**	5 **3**
s	**2** **2**	**1** 3	2 **1**	**2** **2**	3 **2**	**3** **3**	**3** **3**	3 4	4 ?
l	**3** **3**								
o	**4** **4**								

64

		s		n		o		w	
	0	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
o	**1** / **1**	**1** / 2	2 / **1**	**2** / **2**	3 / **2**	**2** / 3	4 / **2**	4 / **3**	5 / **3**
s	**2** / **2**	**1** / 3	2 / **1**	**2** / **2**	3 / **2**	**3** / **3**	3 / **3**	**3** / 4	4 / **3**
l	**3** / **3**								
o	**4** / **4**								

65

		s		n		o		w	
		0	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
o		**1** / **1**	**1** / 2	2 / **1**	**2** / **2**	**2** / 3	3 / **2**	**2** / 4	**3** / **3**	4 / 5
s		**2** / **2**	**1** / 2	3 / **1**	**2** / **2**	**3** / **3**	**3** / **3**	**3** / 4	4 / **3**	**3** / 4
l		**3** / **3**	3 / 2	4 / ?						
o		**4** / **4**								

66

		s	n	o	w
	0	**1** **1**	**2** **2**	**3** **3**	**4** **4**
o	**1** / **1**	**1** 2 / 2 **1**	**2** 3 / **2** **2**	**2** 4 / 3 **2**	4 5 / **3** **3**
s	**2** / **2**	**1** 2 / 3 **1**	**2** 3 / **2** **2**	**3** **3** / **3** **3**	**3** 4 / 4 **3**
l	**3** / **3**	3 **2** / 4 **2**			
o	**4** / **4**				

67

		s		n		o		w	
	0	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
o	**1** **1**	**1** 2	2 **1**	**2** **2**	3 **2**	**2** 3	4 **2**	4 **3**	5 **3**
s	**2** **2**	**1** 3	2 **1**	**2** **2**	3 **2**	**3** **3**	**3** **3**	**3** 4	4 **3**
l	**3** **3**	3 4	**2** **2**	2 3	3 ?				
o	**4** **4**								

68

		s		n		o		w	
	0	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
o	**1** **1**	**1** 2	2 **1**	**2** **2**	3 **2**	**2** 3	4 **2**	4 **3**	5 **3**
s	**2** **2**	**1** 3	2 **1**	**2** **2**	3 **2**	**3** **3**	**3** **3**	**3** 4	4 **3**
l	**3** **3**	3 4	**2** **2**	**2** 3	3 **2**				
o	**4** **4**								

69

		s		n		o		w	
	0	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
o	**1** **1**	**1** 2	2 **1**	**2** **2**	3 **2**	**2** 3	4 **2**	4 **3**	5 **3**
s	**2** **2**	**1** 3	2 **1**	**2** **2**	3 **2**	**3** **3**	**3** **3**	**3** 4	4 **3**
l	**3** **3**	3 4	**2** **2**	**2** 3	3 **2**	3 3	4 ?		
o	**4** **4**								

70

		s		n		o		w	
	0	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
o	**1** **1**	**1** 2	2 **1**	**2** **2**	3 **2**	**2** 3	4 **2**	4 **3**	5 **3**
s	**2** **2**	**1** 3	2 **1**	**2** **2**	3 **2**	**3** **3**	**3** **3**	**3** 4	4 **3**
l	**3** **3**	3 4	**2** **2**	**2** 3	3 **2**	**3** **3**	4 **3**		
o	**4** **4**								

71

		s		n		o		w	
	0	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
o	**1** **1**	**1** 2	2 **1**	**2** **2**	3 **2**	**2** 3	4 **2**	4 **3**	5 **3**
s	**2** **2**	**1** 3	2 **1**	**2** **2**	3 **2**	**3** **3**	**3** **3**	**3** 4	4 **3**
l	**3** **3**	3 4	**2** **2**	**2** 3	3 **2**	**3** **3**	4 **3**	4 4	4 ?
o	**4** **4**								

72

		s		n		o		w	
	0	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
o	**1** **1**	**1** 2	2 **1**	**2** **2**	3 **2**	**2** 3	4 **2**	4 **3**	5 **3**
s	**2** **2**	**1** 3	2 **1**	**2** **2**	3 **2**	**3** **3**	**3** **3**	**3** 4	4 **3**
l	**3** **3**	3 4	**2** **2**	**2** 3	3 **2**	**3** **3**	4 **3**	**4** **4**	**4** **4**
o	**4** **4**								

73

		s		n		o		w	
	0	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
o	**1** **1**	**1** 2	2 **1**	**2** **2**	3 **2**	**2** 3	4 **2**	4 **3**	5 **3**
s	**2** **2**	**1** 3	2 **1**	**2** **2**	3 **2**	**3** **3**	**3** **3**	**3** 4	4 **3**
l	**3** **3**	3 4	**2** **2**	**2** 3	3 **2**	**3** **3**	4 **3**	**4** **4**	**4** **4**
o	**4** **4**	4 5	3 ?						

74

		s	n	o	w
	0	**1** **1**	**2** **2**	**3** **3**	**4** **4**
o	**1** **1**	**1** 2 / 2 **1**	**2** 3 / **2** **2**	**2** 4 / 3 **2**	4 5 / **3** **3**
s	**2** **2**	**1** 2 / 3 **1**	**2** 3 / **2** **2**	**3** **3** / **3** **3**	**3** 4 / 4 **3**
l	**3** **3**	3 **2** / 4 **2**	**2** 3 / 3 **2**	**3** 4 / **3** **3**	**4** **4** / **4** **4**
o	**4** **4**	4 **3** / 5 **3**			

75

		s		n		o		w	
	0	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
o	**1** **1**	**1** 2	2 **1**	**2** **2**	3 **2**	**2** 3	4 **2**	4 **3**	5 **3**
s	**2** **2**	**1** 3	2 **1**	**2** **2**	3 **2**	**3** **3**	**3** **3**	**3** 4	4 **3**
l	**3** **3**	3 4	**2** **2**	**2** 3	3 **2**	**3** **3**	4 **3**	**4** **4**	**4** **4**
o	**4** **4**	4 5	**3** **3**	3 4	3 ?				

76

		s		n		o		w	
	0	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
o	**1** **1**	**1** 2	2 **1**	**2** **2**	3 **2**	**2** 3	4 **2**	4 **3**	5 **3**
s	**2** **2**	**1** 3	2 **1**	**2** **2**	3 **2**	**3** **3**	**3** **3**	**3** 4	4 **3**
l	**3** **3**	3 4	**2** **2**	**2** 3	3 **2**	**3** **3**	4 **3**	**4** **4**	**4** **4**
o	**4** **4**	4 5	**3** **3**	**3** 4	**3** **3**				

77

		s		n		o		w	
	0	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
o	**1** **1**	**1** 2	2 **1**	**2** **2**	3 **2**	**2** 3	4 **2**	4 **3**	5 **3**
s	**2** **2**	**1** 3	2 **1**	**2** **2**	3 **2**	**3** **3**	**3** **3**	**3** 4	4 **3**
l	**3** **3**	3 4	**2** **2**	**2** 3	3 **2**	**3** **3**	4 **3**	**4** **4**	**4** **4**
o	**4** **4**	4 5	**3** **3**	**3** 4	**3** **3**	2 4	4 ?		

78

		s		n		o		w	
	0	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
o	**1** **1**	**1** 2	2 **1**	**2** **2**	3 **2**	**2** 3	4 **2**	4 **3**	5 **3**
s	**2** **2**	**1** 3	2 **1**	**2** **2**	3 **2**	**3** **3**	**3** **3**	**3** 4	4 **3**
l	**3** **3**	3 4	**2** **2**	**2** 3	3 **2**	**3** **3**	4 **3**	**4** **4**	**4** **4**
o	**4** **4**	4 5	**3** **3**	**3** 4	**3** **3**	**2** 4	4 **2**		

79

		s	n	o	w
	0	**1** **1**	**2** **2**	**3** **3**	**4** **4**
o	**1** **1**	**1** 2 2 **1**	**2** 3 2 **2**	**2** 4 3 **2**	4 5 **3** **3**
s	**2** **2**	**1** 2 3 **1**	**2** 3 2 **2**	**3** **3** 3 **3**	**3** 4 4 **3**
l	**3** **3**	3 **2** 4 **2**	**2** 3 3 **2**	**3** 4 3 **3**	**4** **4** **4** **4**
o	**4** **4**	4 **3** 5 **3**	**3** **3** 4 **3**	**2** 4 4 **2**	4 5 3 ?

80

		s		n		o		w	
	0	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
o	**1** **1**	**1** 2	2 **1**	**2** **2**	3 **2**	**2** 3	4 **2**	4 **3**	5 **3**
s	**2** **2**	**1** 3	2 **1**	**2** **2**	3 **2**	**3** **3**	**3** **3**	**3** 4	4 **3**
l	**3** **3**	3 4	**2** **2**	**2** 3	3 **2**	**3** **3**	4 **3**	**4** **4**	**4** **4**
o	**4** **4**	4 5	**3** **3**	**3** 4	**3** **3**	**2** 4	4 **2**	4 **3**	5 **3**

81

		s		n		o		w	
	0	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
o	**1** **1**	**1** 2	2 **1**	**2** **2**	3 **2**	**2** 3	4 **2**	4 **3**	5 **3**
s	**2** **2**	**1** 3	2 **1**	**2** **2**	3 **2**	**3** **3**	**3** **3**	**3** 4	4 **3**
l	**3** **3**	3 4	**2** **2**	**2** 3	3 **2**	**3** **3**	4 **3**	**4** **4**	**4** **4**
o	**4** **4**	4 5	**3** **3**	**3** 4	**3** **3**	**2** 4	4 **2**	4 **3**	5 **3**

82

		s	n	o	w
	0	**1** **1**	**2** **2**	**3** **3**	**4** **4**
o	**1** / **1**	**1** 2 / 2 **1**	**2** 3 / **2** **2**	**2** 4 / 3 **2**	4 5 / **3** **3**
s	**2** / **2**	**1** 2 / 3 **1**	**2** 3 / **2** **2**	**3** **3** / **3** **3**	**3** 4 / 4 **3**
l	**3** / **3**	3 **2** / 4 **2**	**2** 3 / 3 **2**	**3** 4 / **3** **3**	**4** **4** / **4** **4**
o	**4** / **4**	4 **3** / 5 **3**	**3** **3** / 4 **3**	**2** 4 / 4 **2**	4 5 / **3** **3**

How do

I read out the editing operations that transform OSLO into SNOW?

83

		s	n	o	w
	0	**1** **1**	**2** **2**	**3** **3**	**4** **4**
o	**1** / **1**	**1** 2 / 2 **1**	**2** 3 / **2** **2**	**2** 4 / 3 **2**	4 5 / **3** **3**
s	**2** / **2**	**1** 2 / 3 **1**	**2** 3 / **2** **2**	**3** **3** / **3** **3**	**3** 4 / 4 **3**
l	**3** / **3**	3 **2** / 4 **2**	**2** 3 / 3 **2**	**3** 4 / **3** **3**	**4** **4** / **4** **4**
o	**4** / **4**	4 **3** / 5 **3**	**3** **3** / 4 **3**	**2** 4 / 4 **2**	4 5 / **3** **3**

cost	operation	input	output
1	insert	*	w

		s	n	o	w
	0	**1** **1**	**2** **2**	**3** **3**	**4** **4**
o	**1** / **1**	**1** 2 / 2 **1**	**2** 3 / **2** **2**	**2** 4 / 3 **2**	4 5 / **3** **3**
s	**2** / **2**	**1** 2 / 3 **1**	**2** 3 / **2** **2**	**3** **3** / **3** **3**	**3** 4 / 4 **3**
l	**3** / **3**	3 **2** / 4 **2**	**2** 3 / 3 **2**	**3** 4 / **3** **3**	**4** **4** / **4** **4**
o	**4** / **4**	4 **3** / 5 **3**	**3** **3** / 4 **3**	**2** 4 / 4 **2**	4 5 / **3** **3**

cost	operation	input	output
0	(copy)	o	o
1	insert	*	w

		s	n	o	w
	0	**1** **1**	**2** **2**	**3** **3**	**4** **4**
o	**1** / **1**	**1** 2 / 2 **1**	**2** 3 / **2** **2**	**2** 4 / 3 **2**	4 5 / **3** **3**
s	**2** / **2**	**1** 2 / 3 **1**	**2** 3 / **2** **2**	**3** **3** / **3** **3**	**3** 4 / 4 **3**
l	**3** / **3**	3 **2** / 4 **2**	2 3 / 3 2	**3** 4 / **3** **3**	**4** **4** / **4** **4**
o	**4** / **4**	4 **3** / 5 **3**	**3** **3** / 4 **3**	2 4 / 4 2	4 5 / **3** **3**

cost	operation	input	output
1	replace	l	n
0	(copy)	o	o
1	insert	*	w

		s	n	o	w
	0	**1** **1**	**2** **2**	**3** **3**	**4** **4**
o	**1** **1**	**1** 2 / 2 **1**	**2** 3 / **2** **2**	**2** 4 / 3 **2**	4 5 / **3** **3**
s	**2** **2**	**1** 2 / 3 **1**	**2** 3 / **2** **2**	**3** **3** / **3** **3**	**3** 4 / 4 **3**
l	**3** **3**	3 **2** / 4 **2**	**2** 3 / 3 **2**	**3** 4 / **3** **3**	**4** **4** / **4** **4**
o	**4** **4**	4 **3** / 5 **3**	**3** **3** / 4 **3**	**2** 4 / 4 **2**	4 5 / **3** **3**

cost	operation	input	output
0	(copy)	s	s
1	replace	l	n
0	(copy)	o	o
1	insert	*	w

		s	n	o	w
	0	**1** **1**	**2** **2**	**3** **3**	**4** **4**
o	**1** / **1**	**1** 2 / 2 **1**	**2** 3 / **2** **2**	**2** 4 / 3 **2**	4 5 / **3** **3**
s	**2** / **2**	**1** 2 / 3 **1**	**2** 3 / **2** **2**	**3** **3** / **3** **3**	**3** 4 / 4 **3**
l	**3** / **3**	3 **2** / 4 **2**	**2** 3 / 3 **2**	**3** 4 / **3** **3**	**4** **4** / **4** **4**
o	**4** / **4**	4 **3** / 5 **3**	**3** **3** / 4 **3**	**2** 4 / 4 **2**	4 5 / **3** **3**

cost	operation	input	output
1	delete	o	*
0	(copy)	s	s
1	replace	l	n
0	(copy)	o	o
1	insert	*	w

			c		a		t		c		a		t	
		0	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**	**5**	**5**	**6**	**6**
c	**1**	**1**	**0**	2	2	3	3	4	**3**	5	5	6	6	7
	1		2	**0**	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**	**5**	**5**
a	**2**	**2**	2	**1**	**0**	2	2	3	3	4	**3**	5	5	6
	2		3	**1**	2	**0**	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
t	**3**	**3**	3	**2**	2	**1**	**0**	2	2	3	3	4	**3**	5
	3		4	**2**	3	**1**	2	**0**	**1**	**1**	**2**	**2**	**3**	**3**

		c	a	t	c	a	t
	0	**1** **1**	**2** **2**	**3** **3**	**4** **4**	**5** **5**	**6** **6**
c	**1** **1**	0 2 2 **0**	2 3 **1** **1**	3 4 **2** **2**	**3** 5 **3** **3**	5 6 **4** **4**	6 7 **5** **5**
a	**2** **2**	2 **1** 3 **1**	**0** 2 2 **0**	2 3 **1** **1**	3 4 **2** **2**	**3** 5 **3** **3**	5 6 **4** **4**
t	**3** **3**	3 **2** 4 **2**	2 **1** 3 **1**	**0** 2 2 **0**	2 3 **1** **1**	3 4 **2** **2**	**3** 5 **3** **3**

cost	operation	input	output
1	insert	*	c
1	insert	*	a
1	insert	*	t
0	(copy)	c	c
0	(copy)	a	a
0	(copy)	t	t

		c	a	t	c	a	t
	0	**1** **1**	**2** **2**	**3** **3**	**4** **4**	**5** **5**	**6** **6**
c	**1** **1**	0 2 / 2 **0**	2 3 / **1** **1**	3 4 / **2** **2**	**3** 5 / **3** **3**	5 6 / **4** **4**	6 7 / **5** **5**
a	**2** **2**	2 **1** / 3 **1**	**0** 2 / 2 **0**	2 3 / **1** **1**	3 4 / **2** **2**	**3** 5 / **3** **3**	5 6 / **4** **4**
t	**3** **3**	3 **2** / 4 **2**	2 **1** / 3 **1**	**0** 2 / 2 **0**	2 3 / **1** **1**	3 4 / **2** **2**	**3** 5 / **3** **3**

cost	operation	input	output
0	(copy)	c	c
1	insert	*	a
1	insert	*	t
1	insert	*	c
0	(copy)	a	a
0	(copy)	t	t

		c	a	t	c	a	t
	0	**1** 1	**2** 2	**3** 3	**4** 4	**5** 5	**6** 6
c	**1** / **1**	0 2 / 2 **0**	2 3 / **1** **1**	3 4 / **2** **2**	**3** 5 / **3** **3**	5 6 / **4** **4**	6 7 / **5** **5**
a	**2** / **2**	2 **1** / 3 **1**	0 2 / 2 **0**	2 3 / **1** **1**	3 4 / **2** **2**	**3** 5 / **3** **3**	5 6 / **4** **4**
t	**3** / **3**	3 **2** / 4 **2**	2 **1** / 3 **1**	**0** 2 / 2 **0**	2 3 / **1** **1**	3 4 / **2** **2**	**3** 5 / **3** **3**

cost	operation	input	output
0	(copy)	c	c
0	(copy)	a	a
1	insert	*	t
1	insert	*	c
1	insert	*	a
0	(copy)	t	t

		c	a	t	c	a	t
	0	**1** **1**	**2** **2**	**3** **3**	**4** **4**	**5** **5**	**6** **6**
c	**1** **1**	**0** 2 / 2 **0**	2 3 / **1** **1**	3 4 / **2** **2**	**3** 5 / **3** **3**	5 6 / **4** **4**	6 7 / **5** **5**
a	**2** **2**	2 **1** / 3 **1**	**0** 2 / 2 **0**	2 3 / **1** **1**	3 4 / **2** **2**	**3** 5 / **3** **3**	5 6 / **4** **4**
t	**3** **3**	3 **2** / 4 **2**	2 **1** / 3 **1**	**0** 2 / 2 **0**	2 3 / **1** **1**	3 4 / **2** **2**	**3** 5 / **3** **3**

cost	operation	input	output
0	(copy)	c	c
0	(copy)	a	a
0	(copy)	t	t
1	insert	*	c
1	insert	*	a
1	insert	*	t

تصحیح املایی

- حالا که ما می‌توانیم فاصله ویرایشی را محاسبه کنیم:
  - چطور این فاصله را برای تصحیح املایی کلمه مجزا استفاده کنیم؟

- شاخص‌های k-گرمی برای تصحیح املای کلمه مجزا
- تصحیح املایی حساس به متن
- مسائل عمومی

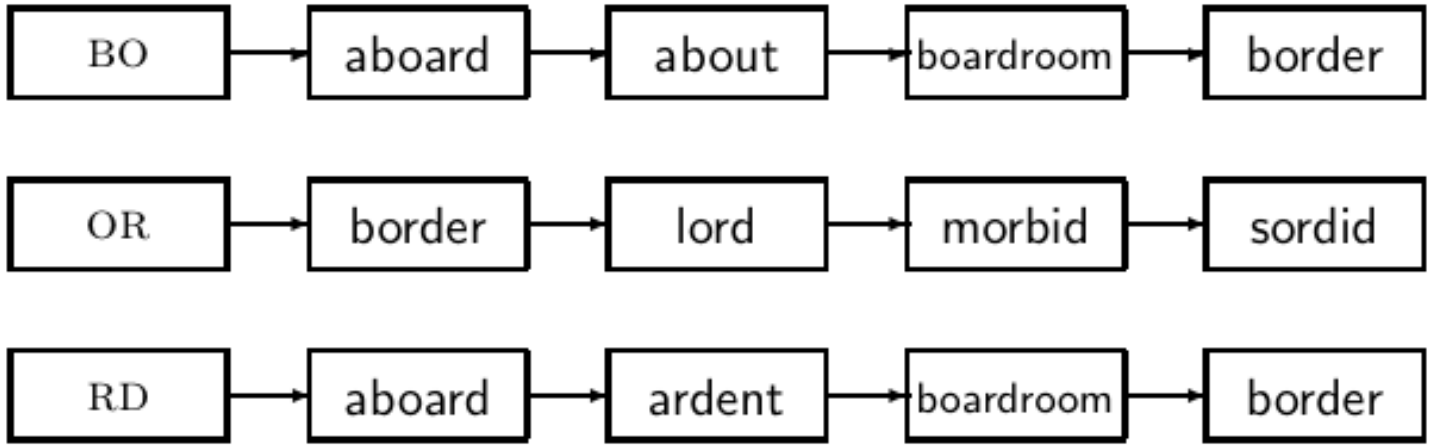
شاخص‌های k–گرمی برای تصحیح املایی

- همه‌ی k-گرمی‌ها را در عبارت پرس‌وجو محاسبه و ثبت می‌کنیم.

- نمونه: شاخص دو-گرمی (bigram) کلمه با اشتباه املایی bordroom به صورت زیر می‌باشد.
  - Bigrams: bo, or, rd, dr, ro, oo, om

- از شاخص k-گرمی برای بازیابی کلمات "صحیح" که با k-گرمی‌های عبارت پرس‌وجو انطباق پیدا می‌کنند استفاده می‌شود.
- حدآستانه با تعداد تطبیقات k-گرمی ها
- به عنوان مثال: فقط عبارات واژگانی که بیشتر از ۳، k-گرم‌ها متفاوت هستند

شاخص‌های k–گرمی برای تصحیح املایی: *bordroom*



- شاخص ۲-گرمی- پست ها (بخشی از آنها) را برای سه ۲-گرمی در پرس و جوی board را نشان می دهد.
- فرض کنید می خواستیم عبارات مجموعه واژگانی را که شامل حداقل دو تا از این سه ۲-گرمی است، بازیابی کنیم.
- یک اسکن واحد از پست ها به ما اجازه می دهد تا تمامی چنین عباراتی را محاسبه کنیم.
- در این مثال، عبارات boardroom، aboard و border در نظر گرفته می‌شوند.

تصحیح املایی حساس به متن

- نمونه: *an asteroid that fell form the sky*
- چگونه می‌توان form را اینجا تصحیح کرد؟
- ایده اول: تصحیح املایی مبتنی بر ضربه (hit-based)
  - در این روش بازیابی عبارات "صحیح" به مانند بازیابی هر عبارت پرس‌وجو (بخش قبل)
  - در این روش اسناد اندکی بازیابی می‌شوند.
  - برای flew form munich:
    - flea for flew
    - from for form
    - munch for munich

تصحیح املایی حساس به متن

- حالا همه‌ی نتایج عبارات ممکن را به عنوان پرس‌وجوهایی با یک کلمه "ثابت" در یک زمان امتحان کنید
  - Try query “flea form munich”
  - Try query “flew from munich”
  - Try query “flew form munch”
  - The correct query “flew from munich” has the most hits.
- فرض کنید ما ۷ گزینه برای flew، ۲۰ برای form و ۳ تا برای munich
- چند عبارت تصحیح شده باید شمارش شود؟
- اگر برای یک عبارت واحد تصحیح های بسیاری پیدا شود، این محاسبات می تواند بسیار پرهزینه باشد چون ممکن است با تعداد زیادی از ترکیب های عبارات جایگزین مواجه شویم.

تصحیح املایی حساس به متن

- الگوریتم مبتنی بر ضربه (hit-based) در اینجا فقط معرفی شده است و کارآمدی خوبی ندارد.
- جایگزین کارآمدتر: نگاهی به "مجموعه" پرس‌وجوها ، نه اسناد

Soundex

- Soundex اساسا برای پیدا کردن آوایی (phonetic) می‌باشد
- نمونه: *chebyshev / tchebyscheff*
- الگوریتم:
  - هر عبارت را به صورت کاهش‌یافته ۴ کارکتری شاخص کنید.
  - با عبارات پرس‌وجو نیز همین کار را انجام دهید.
  - هنگامی که پرس‌وجو نیاز به تطبیق Soundex دارد، این شاخص را جستجو کنید.

الگوریتم Soundex

۱. اولین عبارت را حذف کنید.

۲. تمامی رخدادهای حروف A, E, I, O, U, H, W, Y را به ’0‘ تغییر دهید.

۳. حروف را به ترتیب زیر به ارقام تغییر دهید:

- B, F, P, V to 1
- C, G, J, K, Q, S, X, Z to 2
- D,T to 3
- L to 4
- M, N to 5
- R to 6

۴. به طور تکرار شونده، یکی از جفت ارقام یکسان متوالی را حذف کنید.

۵. تمامی صفرها را از رشته‌ی حاصل شده حذف کنید. رشته‌ی حاصل شده را با یک دنباله از صفر پر کنید و چهار موقعیت اول را برگردانید که شامل حرفی است که به دنبال آن سه رقم وجود دارد.

مثال Soundex برای HERMAN

- Retain H
- *ERMAN* → *0RM0N*
- *0RM0N* → *06505*
- *06505* → *06505*
- *06505* → *655*
- Return *H655*
- Note: *HERMANN* will generate the same code

102