Vocabulário de Termos e Listas de Postings

Prof. Dr. Leandro Balby Marinho

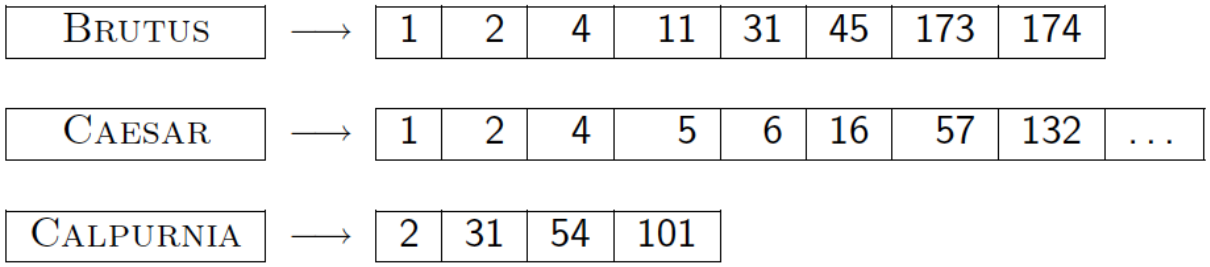
http//www.dsc.ufcg.edu.br/~lbmarinho

Sistemas de Recuperação da Informação

(Slides Adaptados de Cristopher D. Manning)

Na Aula Passada

- Índices invertidos básicos:
  - Estrutura: Dicionário e Postings



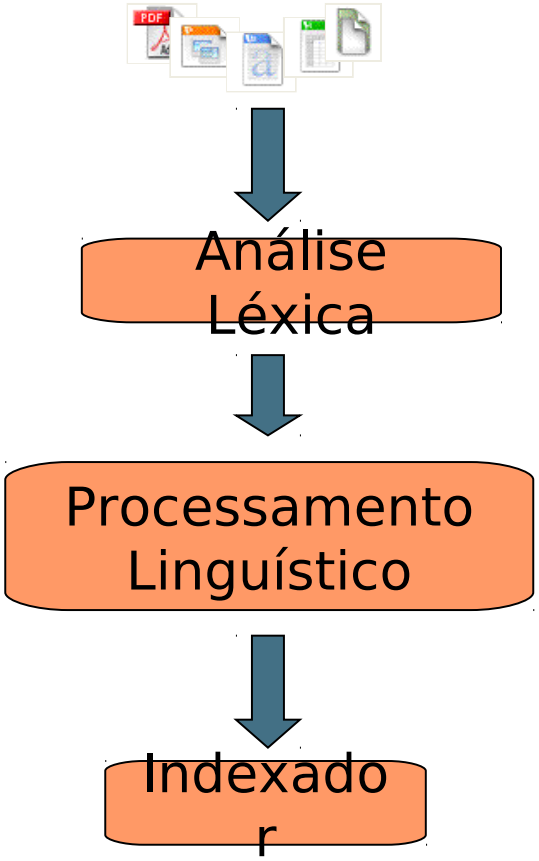
  - Etapa principal: Ordenação
- Processamento de consultas Booleanas
  - Interseção em tempo linear “merging”
  - Otimizações simples

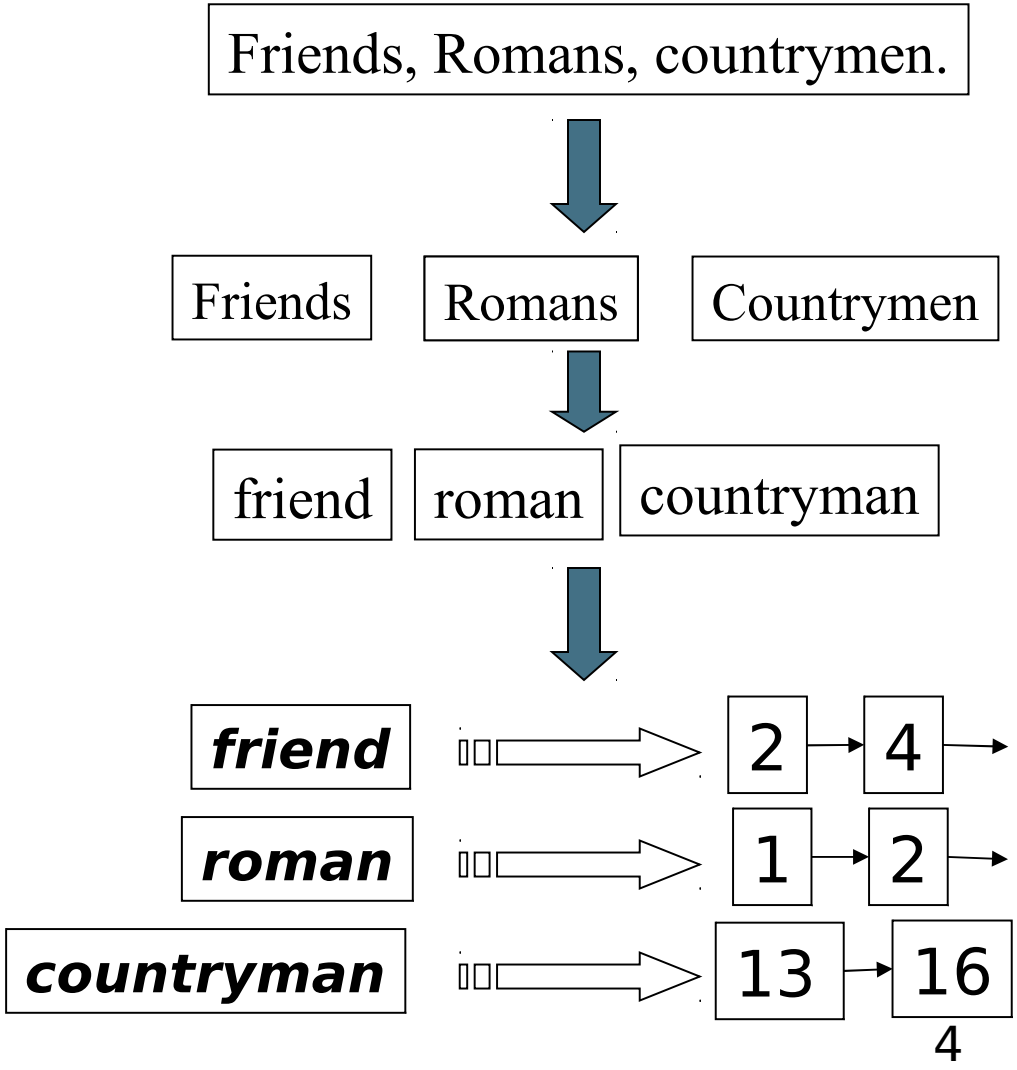
Na Aula de Hoje...

Elaborar indexação básica

- Pré-processamento para formar o vocabulário de termos
  - Documentos
  - Análise Léxica
  - Quais *termos* colocamos no indíce?
- Postings
  - Merges rápidos: skip lists
  - Postings posicionais e consultas de frases

Relembre as Etapas de Indexação





Análise Sintática de um Documento

- Qual o formato do documento?
  - pdf/word/excel/html?
- Qual o idioma?
- Que codificação de caracteres é utilizada?

Cada um desses problemas pode ser definido como um problema de classificação. Mas essas tarefas são geralmente feitas de forma heurística ...

Complicações: Formato/idioma

- Coleções podem incluir documentos em muitos idiomas diferentes
  - Um único índice pode conter termos de vários idiomas.
- Algumas vezes um documento ou seus componentes podem conter múltiplos idiomas/formatos
  - Email em Francês com um anexo pdf em Alemão.
- O que constitui uma unidade de documento?
  - Um arquivo?
  - Um email? (diferentes arquivos em um folder)
  - Um email com 5 anexos?
  - Um grupo de arquivos (PPT ou LaTeX como páginas HTML)

TERMOS E TOKENS

Análise Léxica

- Input: "***Friends, Romans and Countrymen***"
- Output: Tokens
  - ***Friends***
  - ***Romans***
  - ***Countrymen***
- Um token é uma instância de uma sequência de caracteres representando uma unidade semântica
- Cada token é um candidato para uma entrada no índice, depois de devidamente processado
- Como escolher os "melhores" candidatos?

Análise Léxica

- Complicações:
  - ***Brazil's capital*** $\rightarrow$ ***Brazil? Brazils? Brazil's***?
  - ***Hewlett-Packard*** $\rightarrow$ ***Hewlett*** and ***Packard*** como dois tokens?
    - ***estado-da-arte***: quebra sequências hifenizadas.
    - ***co-education***
    - ***lowercase***, ***lower-case***, ***lower case*** ?
    - Pode ser efetivo fazer com que o usuário ponha os hifens.
  - ***Campina Grande***: um ou dois tokens ?
    - Como decidimos que trata-se de um token?

Números

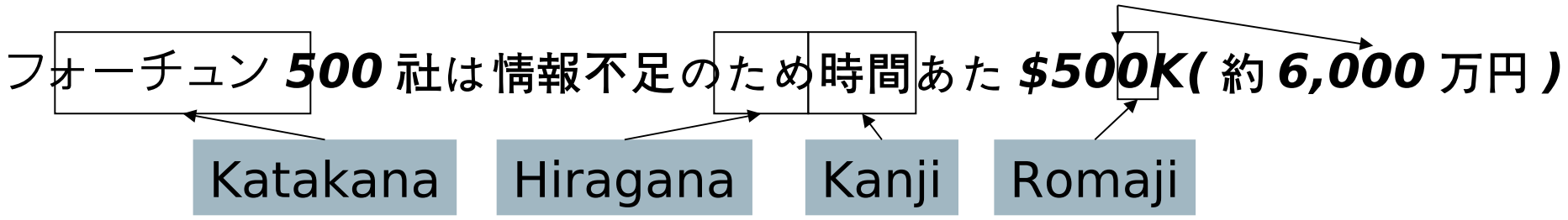
- ***3/20/91 Mar. 12, 1991 20/3/91***
- ***55 B.C.***
- ***B-52***
- ***Meu número de protocolo é 324a3df234cb23e***
- ***(800) 234-2333***
  - Muitas vezes possuem espaços incorporados
  - Sistemas antigos de RI não indexavam números
    - Geralmente útil: considere a busca por linhas de código com erro/rastreamento de objetos no correio
  - Muitas vezes “metadados” são indexados separadamente
    - Data de criação, formato, etc.

Complicações de Idioma

- Francês
  - ***L'ensemble*** → um token ou dois?
    - ***L*** ? ***L'*** ? ***Le*** ?
    - Queremos documentos mencionando tanto ***l'ensemble*** como ***un ensemble*** indexados sob ***ensemble***
      - Até pelo menos 2003, não era possível no Google
        - Internacionalização!
- Substantivos compostos em Alemão não são segmentados
  - ***Lebensversicherungsgesellschaftsangestellter***
  - 'empregado de compania de seguro'
  - Sistemas de RI alemãos se beneficiam de um módulo **divisor de compostos**
    - Pode dar um ganho de 15% para busca por docs em Alemão

Complicações de Idioma

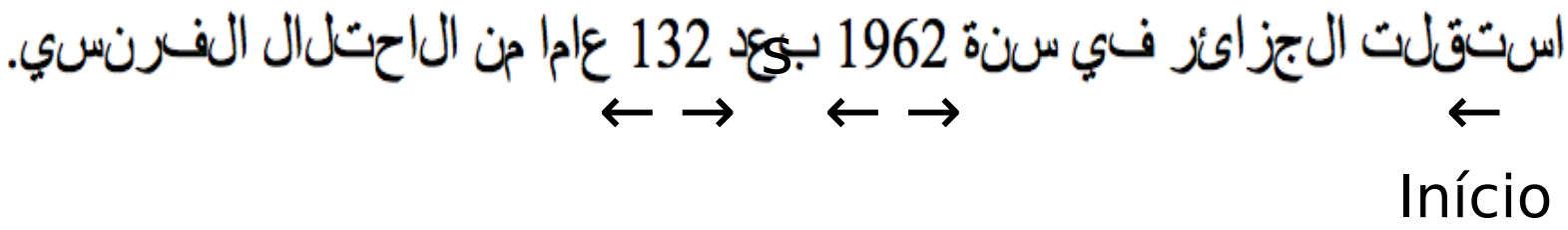
- Chinês e Japonês não possuem espaço entre palavras:
  - 莎拉波娃现在居住在美国东南部的佛罗里达。
  - Não se pode garantir tokens consistentes
- Ainda mais complicada em japonês, com vários alfabetos misturados
  - Datas/quantidades em múltiplos formatos



Usuário final pode expressar consultas inteiramente em hiragana!

Complicações de Idioma

- Árabe (ou Hebraico) é basicamente escrito da direita para esquerda, mas certos itens como números são escritos da esquerda para direita.
- Palavras são separadas, mas as letras dentro de uma palavra formam ligaduras complexas.



- ‘Argélia conseguiu sua independência em 1962, após 132 anos de ocupação francesa.’
- Com Unicode, a apresentação é complexa na superfície, mas o formato de armazenamento é simples.

Lista de Stop Words

- Cria-se uma lista contendo as palavras mais frequentes, de forma a excluí-las do dicionário. Intuição:
  - Possuem fraco conteúdo semântico: a*, o, e, para, ser*
  - Há muitas delas: ~30% dos postings para as 30 palavras mais comuns
- Entretanto, estão caindo em desuso:
  - Técnicas eficientes de compressão reduzem bastante o espaço para o armazenamento de stop words.
  - Técnicas eficientes de consulta não sofrem significativamente com a inclusão de stop words.
  - São importantes para:
    - Consultas de frase: “Presidente do Brasil”
    - Títulos de músicas, etc.: “Let it be”, “To be or not to be”
    - Consultas “relacionais”: “voos para São Luís”

Normalização de termos

- Algumas vezes os termos da consulta não casam exatamente com os do dicionário:
  - ***E.g., E.U.A.*** e ***EUA***
- Precisamos “normalizar” os termos no diconário e na consulta para o mesmo formato.
- Termo é um tipo de palavra (normalizada), que é uma entrada no dicionário do sistema de RI.
- Podemos definir classes de equivalência, e.g.,
  - deletando pontuação
    - ***U.S.A., USA*** ⟨ ***USA***
  - deletando hifens
    - ***anti-terrorista, antiterrorista*** ⟨ ***antiterrorista***

Normalização: outros idiomas

- Acentos: e.g., Francês ***résumé*** vs. ***resume*.**
- Tremas: e.g., Alemão: ***Tuebingen*** vs. ***Tübingen***
  - deveriam ser equivalentes

- Critério mais importante:
  - como seus usuários gostam de escrever suas consultas para essas palavras?

- Usuários podem eventualmente não acentuar as palavras nas suas consultas
  - na maioria das vezes é melhor normalizar desacentuando a palavra
    - ***Tuebingen, Tübingen, Tubingen*** ⌊ ***Tubingen***

Alteração de Maiúsculas e Minúsculas

- Alterar todas as letras para caixa baixa
  - Exceção: caixa alta no meio da sentença?
    - e.g., ***General Motors***
    - ***Salgado*** vs. ***salgado***
    - ***Brasileiro*** vs. ***brasileiro***
  - Na maioria das vezes, os usuários digitam suas consultas em caixa baixa ...
- Exemplo do Google:
  - Consulta ***C.A.T.***
  - Docs ocorrendo “cat” e não Caterpillar Inc.

Tesauro

- E quanto a sinônimos e homônimos?
  - E.g., através de classes de equivalência construídas manualmente
    - ***carro*** = ***automóvel,*** ***color*** = ***colour***
  - Podemos reescrever para formar classes de equivalência de termos
    - Se o documento contém ***automóvel***, indexe-o com ***carro-automóvel*** (e vice-versa)
  - Ou podemos expandir a consulta
    - Se uma consulta contém ***automóvel***, procure por ***carro*** também

Lematização e Stemming

- Reduzem inflexões/variações para uma base comum
- Exemplo,
  - *sou, estou, estava* $\rightarrow$ *ser*
  - *car, cars, car's, cars'* $\rightarrow$ *car*
- *Os carros do garoto são de cores diferentes* $\rightarrow$ o carr do garot *ser de cor diferent*
- Lematização implica em fazer a redução mofológica "adequada" de acordo com o dicionário.

Stemming

- Heurística para cortar o final das palavras de forma “correta”.

- “Stemming” sugere o corte de afixos
  - dependente do idioma
  - e.g., **livro*, livrinho, livreco*** são reduzidos ao radical ***livr***.

for example compressed and compression are both accepted as equivalent to compress.

for exampl compress and compress ar both accept as equival to compress

Algoritmo de Porter

- Algoritmo mais comum para stemming no idioma Inglês
  - Resultados sugerem ser tão bom quanto qualquer outro método de stemming
- Convenções + 5 fases de reduções
  - fases aplicadas sequencialmene
  - cada fase consiste de um conjunto de comandos
  - exemplo de convenção: Das regras *em um comando composto, selecione a que se aplica ao sufixo mais longo.*

Regras Típicas do Algoritmo de Porter

- *sses* → *ss* — *caresses* → *caress*
- *ies* → I — *ponies* → *poni*
- *ss* → ss — *caress* → *caress*
- *s* → — *cats* → *cat*

- Peso das regras sensíveis às palavras
- *(m>1) EMENT* →
  - *replacement* → *replac*
  - *cement* → *cement*

Outros stemmers

- Stemmers para Português
  - http://snowball.tartarus.org/algorithms/portuguese/stemmer.html
  - http://code.google.com/p/ptstemmer/
- Análise morfológica completa – benefícios modestos para recuperação.
- Stemming e outras normalizações ajudam?
  - inglês: resultados diversos. Aumentam o recall em alguns casos mas pioram a precisção.
  - Definitivamente útil para Espanhol, Alemão, Finlandês, ...
    - 30% ganho de desempenho para Finlandês!

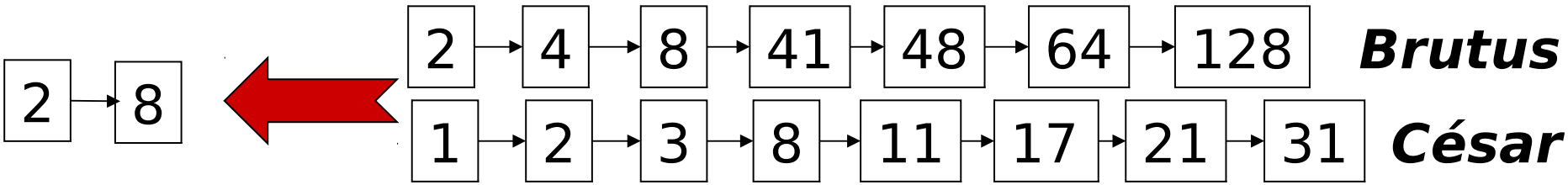
Especificidade do Idioma

- Muitas das técnicas descritas até agora requerem transformações que são:
  - específicas do idioma e
  - frequentemente, específicas da aplicação
- Elas são “plug-in” addenda ao processo de indexação.
- Tanto plug-ins open source quanto commerciais estão disponíveis para fazer isso.

MERGE RÁPIDO: PONTEIROS DE SALTO/LISTAS DE SALTO

Relembre o Merge Básico

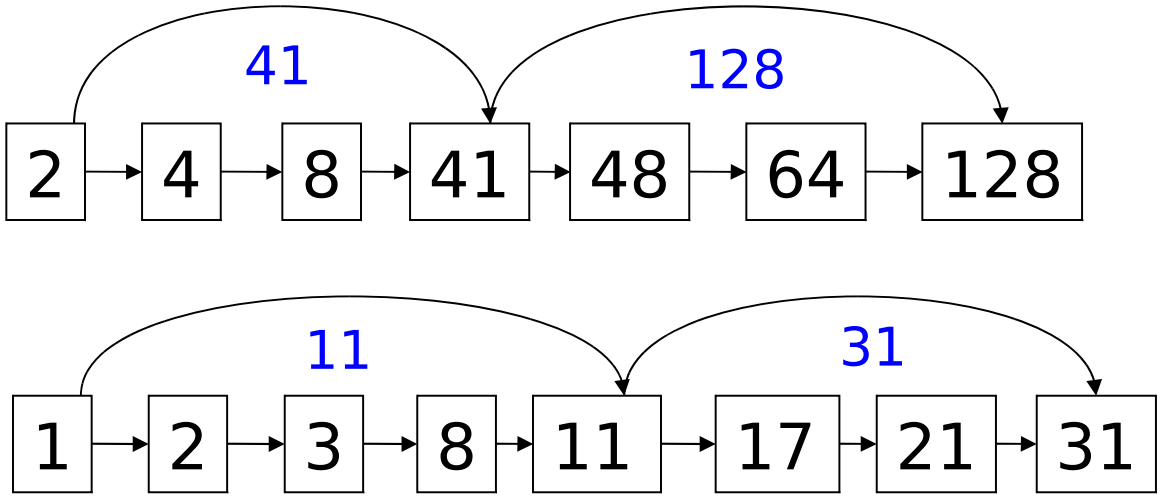
- Corra os postings simultaneamente, em tempo linear no número total de entradas



Se os tamanhos das listas são *m* e *n*, o merge realiza O(*m*+*n*) operações.

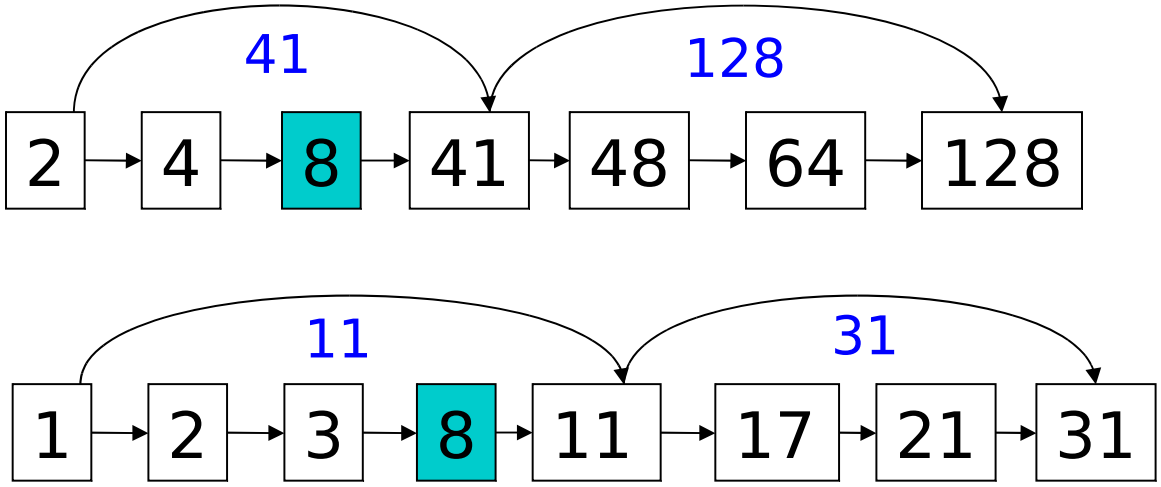
Podemos fazer melhor?
Sim (se o índice não mudar muito rápido).

Aumentar os postings com ponteiros de salto



- Por que?
- Para pular postings que não estarão nos resultados da busca.
- Como?
- Onde colocamos os ponteiros?

Processamento de consultas com listas de salto



Suponha uma varredura nas listas na qual o **8** é encontrada em cada lista.

No próximo passo teremos **41** (lista acima) e **11** (lista de baixo).

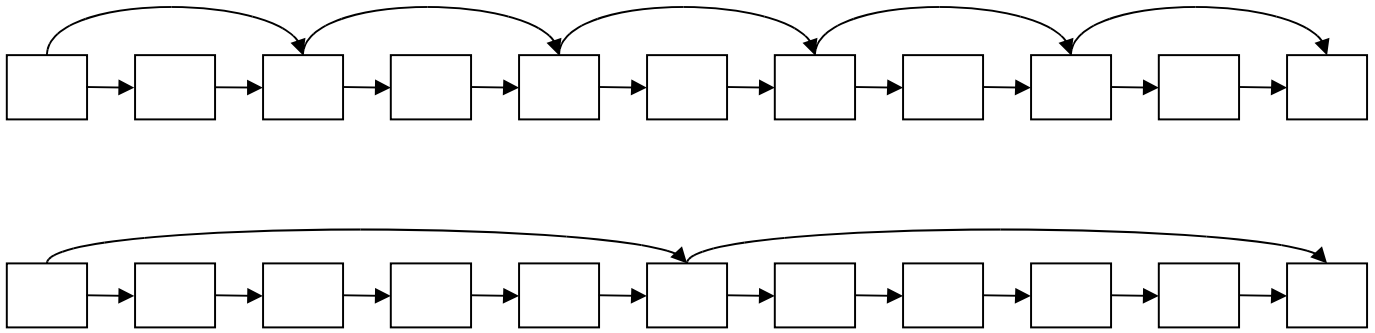
Como o ponteiro de salto de **11** é **31**, podemos saltar o **17** e **21**.

Merge com Listas de Salto

```
INTERSECTWITHSKIPS(p1, p2)
 1  answer ← ⟨ ⟩
 2  while p1 ≠ NIL and p2 ≠ NIL
 3  do if docID(p1) = docID(p2)
 4        then ADD(answer, docID(p1))
 5             p1 ← next(p1)
 6             p2 ← next(p2)
 7        else if docID(p1) < docID(p2)
 8                then if hasSkip(p1) and (docID(skip(p1)) ≤ docID(p2))
 9                     then while hasSkip(p1) and (docID(skip(p1)) ≤ docID(p2))
10                          do p1 ← skip(p1)
11                     else p1 ← next(p1)
12                else if hasSkip(p2) and (docID(skip(p2)) ≤ docID(p1))
13                     then while hasSkip(p2) and (docID(skip(p2)) ≤ docID(p1))
14                          do p2 ← skip(p2)
15                     else p2 ← next(p2)
16  return answer
```

Onde colocamos os ponteiro de salto?

- Tradeoff:
  - Mais ponteiros → saltos mais curtos ⇒ mais provável de saltar. Mas muitas comparações de ponteiros de salto.
  - Menos ponteiros → menos comparações de ponteiro, mas saltos mais longos ⇒ poucos saltos de sucesso.



Atribuindo Ponteiros

- Heurística simples: para postings de tamanho L, use $\sqrt{L}$ ponteiros espaçados uniformemente.
- Fácil se o índice é relativamente estático; mas difícil se L muda muito por causa de updates.
- Costumava ajudar; mas com hardwares modernos pode não ser o caso
  - O custo de I/O para carregar uma lista de postings grande pode superar os ganhos com o merge mais rápido na memória!

CONSULTAS DE FRASE E ÍNDICES POSICIONAIS

Consultas de Frase

- Queremos poder responder consultas do tipo **“Universidade Federal de Campina Grande”** – como uma frase.
- Dessa forma, *“Eu fui para a Universidade em Campina Grande”* não seria uma resposta.
  - O conceito de consultas de frase se provou fácil de entender pelos usuários; uma das poucas idéias de “pesquisa avançada” que funcionam
  - Muitas consultas determinam implicitamente frases
- Sendo assim, não é mais suficiente armazenar apenas registros do tipo <*termo* : *docs*>.

Uma primeira tentativa: Índices Biword

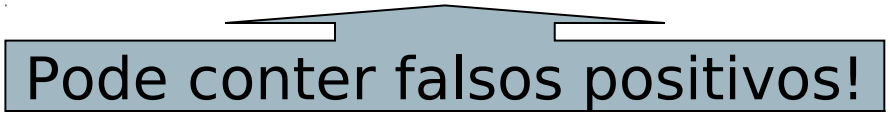
- Indexe todos os pares consecutivos de termos no texto como uma frase.
- Por examplo, o texto “Universidade Federal Campina Grande” geraria as biwords
  - ***universidade federal***
  - ***federal campina***
  - ***campina grande***
- Cada uma dessas biwords se torna um termo no dicionário.
- Processamento de consultas baseadas em frases de duas palavras é agora trivial.

Consultas de frases mais longas

- ***Universidade Federal Campina Grande*** pode ser quebrada em consultas Booleanas em biwords:

  universidade federal *AND* ***federal campina*** *AND* ***campina grande***

Sem examinar os docs, não podemos verificar se os docs casando com a consulta Booleana contém a frase.



Biwords estendidas

- Analisar o texto indexado e realizar part-of-speech-tagging (POST).
- Separe os termos em Substantivos (S) e artigos/preposições (X).
- Considere cada frase da forma SX*S uma biword estendida.
  - Cada uma dessas biwords estendidas é agora um termo no dicionário.
- Examplo: **promessa para o povo**
  S X X S
- Processamento de consulta: classifique as palavras em S's and X's
  - Segmente as consulas em biwords estendidas
  - Procure no índice: **promessa *povo***

Complicações com Biwords

- Falsos positivos podem ocorrer.
- “Explosão” do índice por causa do dicionário maior.
  - Inviável para frases maiores que duas palavras

- Índices baseados em biwords não são a solução padrão (para todas as biwords) mas pode ser parte de uma estratégia composta.

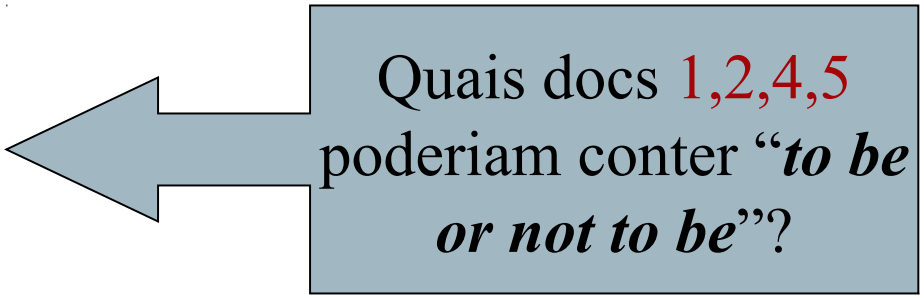
Índices Posicionais

- Armazene nos postings, para cada ***termo,*** as posições em que os termos aparecem:

  <***termo***, *nr. de documentos contendo o* ***termo***;
  doc1: posição1, posição2 ... ;
  doc2: posição1, posição2 ... ;
  etc.>

Exemplo de Índice Posicional

<***be***: 993427;
1: 7, 18, 33, 72, 86, 231;
2: 3, 149;
4: 17, 191, 291, 430, 434;
5: 363, 367, …>



- Para uma consulta de frase, utilizamos um merge recursivo a nível de documento.
- Mas agora precisamos lidar com mais do que apenas igualdade entre docIDs.

Processando uma Consulta de Frase

- Extrai entradas de índices invertidos para cada termo distinto: ***to, be, or, not.***
- Faz o merge nas listas *doc:posição para enumerar todas as* posições com "***to be or not to be***".
  - ***to****:*
    - *2*:1,17,74,222,551; *4*:8,16,190,429,433; *7*:13,23,191; ...
  - ***be****:*
    - *1*:17,19; *4*:17,191,291,430,434; *5*:14,19,101; ...
- Mesmo método para pesquisa por proximidade genérica.

Consultas de Proximidade

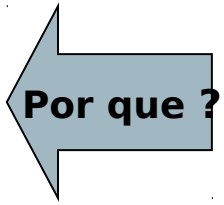
- “universidade /3 federal”
  - Aqui */k* significa “*k* palavras de separação”.
- Índices posicionais podem ser usadas para tal; índices baseados em biword não.
- Exercício: Adapte o merge linear de postings para lidar com consultas de proximidade. É possível fazê-lo para qualquer valor de *k*?
  - É um pouco complicado de fazê-lo correta e eficientemente.
  - Olhe a Figure 2.12 do livro texto
  - É provável haver um problema aí!

Tamanho do Índice Posicional

- Podemos comprimir valores/offsets: mais sobre isso depois.
- Mesmo assim, um índice posicional aumenta o armazenamento de postings substancialmente.
- Apesar disso, índices posicionais são utilizados de forma padrão por causa do poder e utilidade de consultas de frase/proximidade.

Tamanho do Índice Posicional

- Precisa de uma entrada para cada ocorrência, não apenas por documento
- O tamano do índice depende da média do tamanho dos documentos
  - Em média páginas da Web tem <1000 termos
  - Livros, poemas épicos … facilmente 100,000 termos
- Considere um termo com frequência 0.1%



Tamanho do doc.	Postings	Postings posicionais
1000	1	1
100,000	1	100

Regras de Ouro

- Um índice posicional é de 2–4 maior que um índice não posicional.
- Índices posicionais possuem 35–50% do volume dos textos originais
- **Ressalva: Isso corresponde à linguas baseadas no Inglês.**

Combinando Esquemas

- Essas duas abordagens podem ser combinadas
  - Para algumas frases (***“Michael Jackson”, “Britney Spears”***) não é eficiente fazer o merge em listas de postings posicionais
    - Mais ainda para frases como ***“The Who”***

Material da Aula de Hoje

- Livro texto capítulo 2
- Porter's stemmer:
  http://www.tartarus.org/~martin/PorterStemmer/
- Teoria das listas de salto: Pugh (1990)
  - Multilevel skip lists give same O(log *n*) efficiency as trees
- H.E. Williams, J. Zobel, and D. Bahle. 2004. "Fast Phrase Querying with Combined Indexes", ACM Transactions on Information Systems.
  http://www.seg.rmit.edu.au/research/research.php?author=4
- D. Bahle, H. Williams, and J. Zobel. Efficient phrase querying with an auxiliary index. SIGIR 2002, pp. 215-221.