Recuperação Tolerante

Prof. Dr. Leandro Balby Marinho
http//www.dsc.ufcg.edu.br/~lbmarinho

UFCG CEEI Departamento de Sistemas e Computação

Sistemas de Recuperação da Informação

(Slides Adaptados de Cristopher D. Manning)

Na Aula Passada

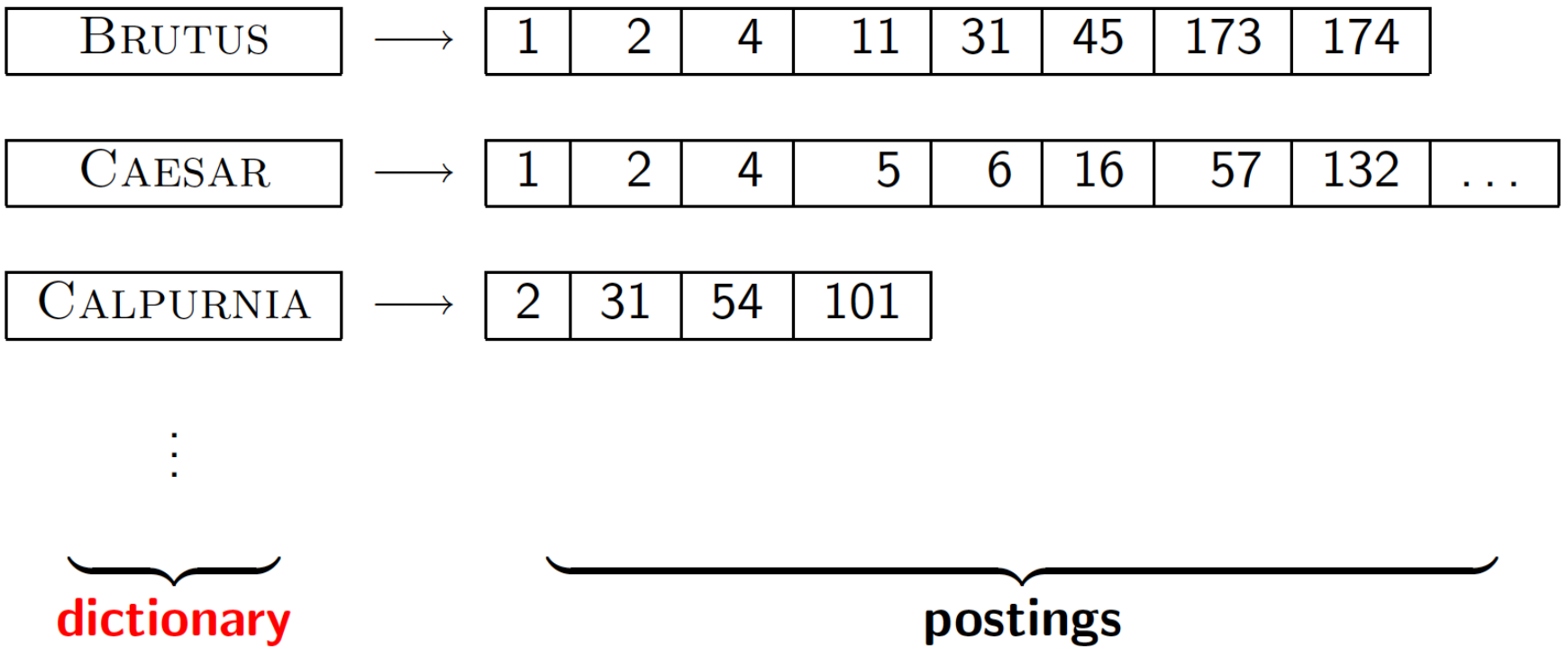
- Distinção termos/tokens
  - Termos são tokens normalizados colocados no dicionário
- Problemas de Análise Léxica
  - Hífens, apóstrofes, nomes compostos, idioma
- Classe de equivalência de termos:
  - Números, maiúsculas/minúsculas, stemming, lemmatização
- Ponteiros de Salto
  - Ponteiros para acelerar o “merging”
- Índices Biword para frases.
- Índices posicionais para frases/consultas de proximidade.

Na Aula de Hoje

- Estruturas de dados para o dicionário.
- Recuperação “Tolerante”
  - Consultas com curingas
  - Correção ortográfica
  - Consulta fonética

Estruturas de Dados do Dicionário

A estrutura de dados do dicionário guarda o vocabulário de termos, frequência dos documentos, ponteiros para listas invertidas ... qual estrutura de dados?



Um Dicionário Ingênuo

- Um array de estruturas:

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

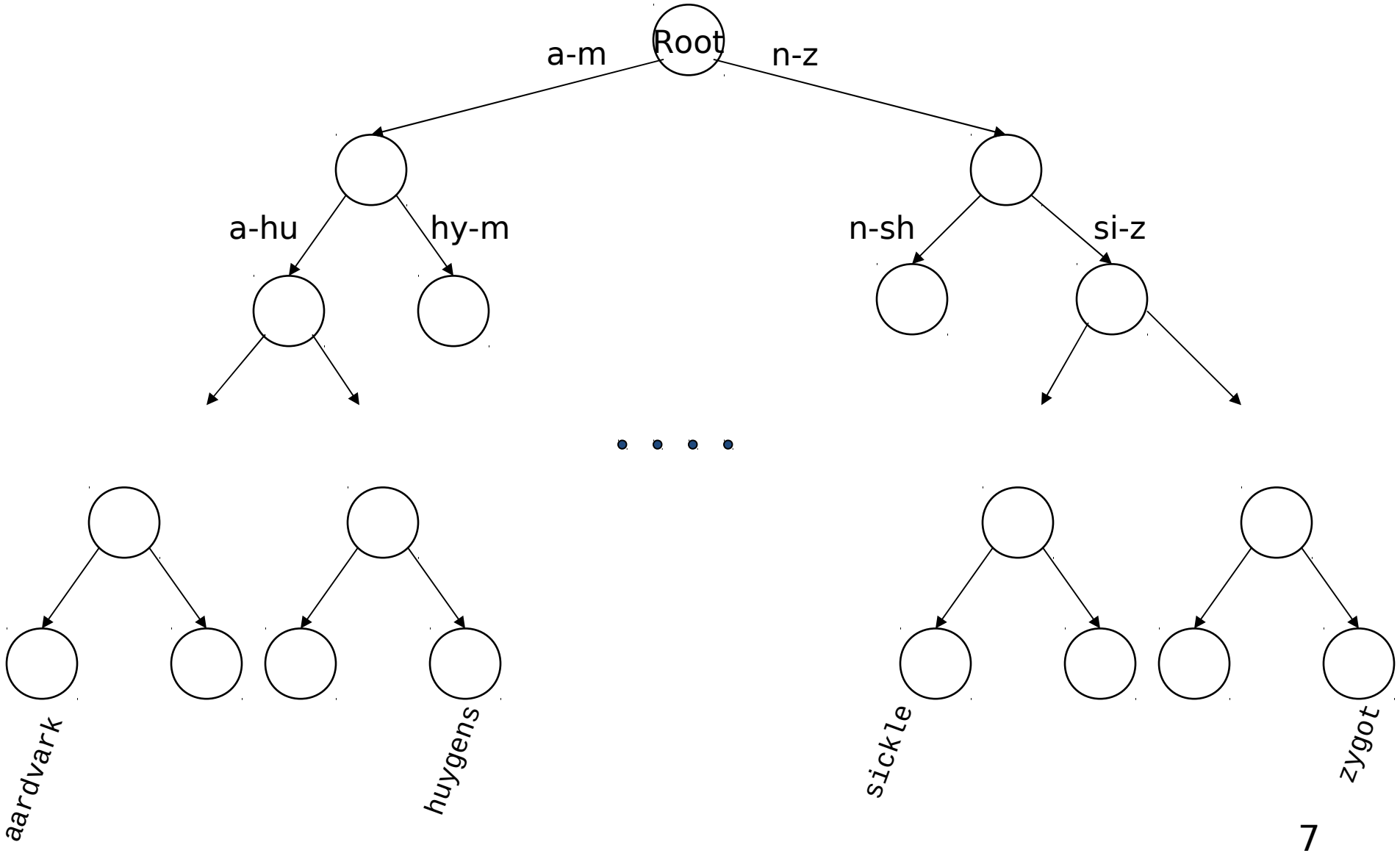
char[20]	int	Postings *
20 bytes	4/8 bytes	4/8 bytes

- Como armazenamos o dicionário eficientemente na memória?
- Como buscar rapidamente elementos em tempo de consulta?

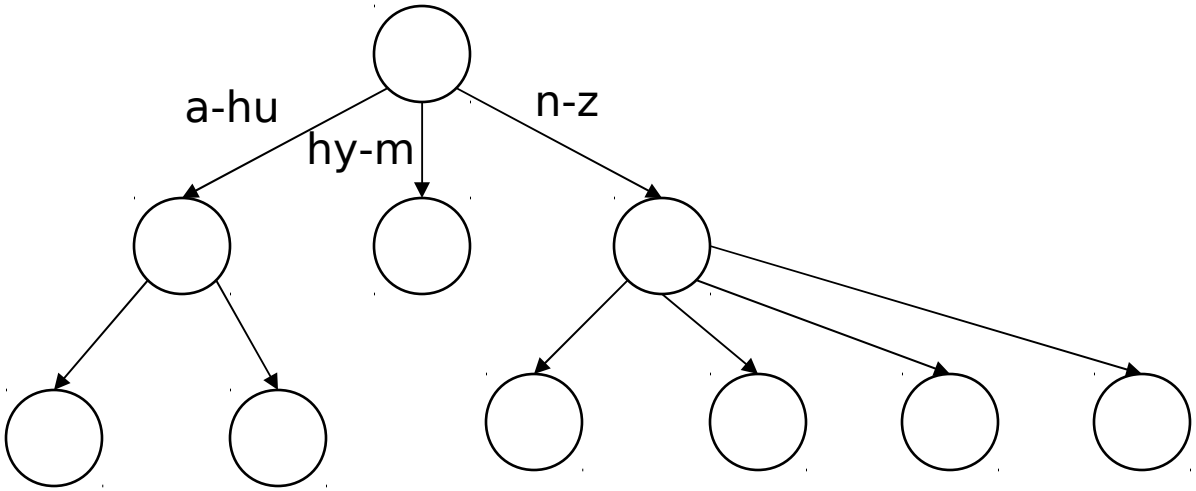
Tabelas Hash

- Cada termo é mapeado para um inteiro (chave).
- Prós:
  - Busca é feita em tempo constante: O(1)
- Contras:
  - Não é fácil de capturar variações ortográficas:
    - julgamento/julgmento
  - Sem busca por prefixos [recuperação tolerante]
  - Se o vocabulário cresce muito, é preciso, ocasionalmente, refazer o hash de todo o dicionário

Árvore: Árvore Binária



Árvore: Árvore B



- Definição: Cada nó interno tem um número de filhos no intervalo [*a*,*b*] onde *a, b* são números naturais apropriados, e.g., [2,4].

Árvores

- Mais simples: árvore binária.
- Mais usual: árvores B.
- Árvores requerem uma ordem padrão de caracteres e strings.
- Prós:
  - Resolve o problema do prefixo (e.g. termos começando com *hiper*)
- Contras:
  - Mais lento: O(log *M*) [requer que árvore esteja *balanceada*]
  - Rebalanceamento de árvores binárias é caro
    - Mas árvores B resolvem esse problema

CONSULTAS CURINGA

Consultas Curinga *

- ***mon*:*** ache todos os docs. contendo qualquer palavra começando com “mon”.

- Simples com dicionários em árvore binária (ou árvore B): recupere todas as palavras **t** no intervalo: ***mon ≤ t < moo***

- ****mon:*** ache todas as palavras terminadas em “mon”: mais difícil
  - Manter uma árvore B inversa para termos. Retorna todas as palavras **t** no intervalo: ***nom ≤ t < non***.

Exercício: como podemos enumerar todos os termos relacionados a consulta curinga **uni*dade** ?

* no Meio dos Termos de Consulta

- Como podemos lidar com * no meio de termos de consulta?
  - ***uni*dade***
- Poderíamos procurar por **uni*** AND ***dade** em uma árvore B e fazer a interseção dos dois conjuntos de termos
  - Caro
- Solução: transformar consultas curinga tais que o * ocorra no fim.
- Índices com permutação de termos.

Árvore de Permutação

- Adicione um símbolo especial, $, para marcar o final do termo.

- Construa um índice que mostre todas as rotações de cada termo, e.g., :
  - ***hello$***
  - ***ello$h***
  - ***llo$he***
  - ***lo$hel***
  - ***o$hell***
  - ***$hello***

- Construa uma árvore B com essas permutações

Processando Consultas na Ávore de Permutações

- Exemplo: Consulta: he*o.
- Rotacione a consulta tal que o curinga vá para o final o$he*.
- Use a árvore de permutações para achar todos os termos que casam com o$he*, e.g., o$hell.
- Problema: quadriplica o tamanho do vocabulário!

Observações empíricas para o inglês

Índices N-gramas

- Enumere todas as *n*-gramas (sequencia de *n* caracteres) ocorrendo em qualquer termo. E.*g.*, o texto "***Hoje tem jogo***" geraria as 2-gramas (*bigramas*)

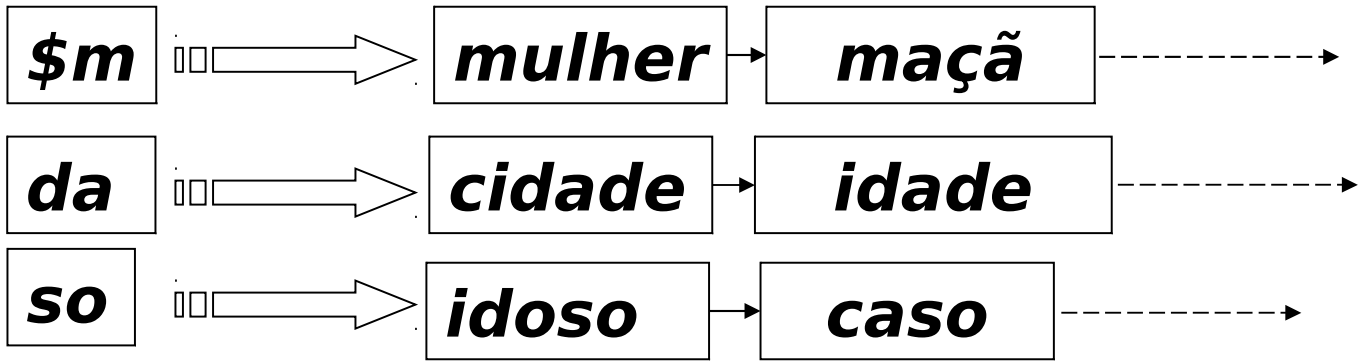
  h,ho,oj,je,e,te,em,m$,jo,og,go,o$

  - $ é um símbolo especial delimitador

- Crie um *segundo* índice invertido *de bigramas para termos do dicionário* que casam com cada bigrama.

Exemplo de Índice Bigrama

- O índice *n*-grama encontra *termos* baseado na consulta consistindo de n-gramas (aqui n=2).



Consultas Curinga com N-*gramas*

- A consulta **cas*** pode ser executada como
  - ***$c*** *AND* ***ca*** *AND* ***as***

- Retorna termos que casam com a versão booleana da consulta curinga.

- Rápida e eficiente em espaço (comparada a árvore de permutações).

Consultas Curinga com N-*gramas*

- Precisamos executar uma consulta Booleana para cada termo enumerado.
- Curingas podem resultar em consultas caras (disjunções longas...)
  - pyth* AND prog*
- Alguns motores de busca encorajam a “preguiça do usuário” através de consultas curingas escondidas atrás de interfaces de consultas avançada!

CORREÇÃO ORTOGRÀFICA

Correção Ortográfica

- Dois usos principais:
  - Correção de documento(s) sendo indexados
  - Correção de consultas para melhorar os resultados

- Dois tipos principais:
  - Palavras isoladas
    - Corrige uma palavra de cada vez
    - Não consegue capturar erros em palavras ortograficamente corretas
    - e.g., ***from*** $\rightarrow$ ***form***
  - Sensível ao contexto
    - Leva em consideração palavras adjacentes,
    - e.g., ***I flew form Salvador.***

Correção Ortográfica de Consultas

- Nosso principal foco nesta aula
  - E.g., a consulta ***Bruce Dickenson***

- Há várias formas de expor o corretor ortográfico aos usuários, por exemplo:
  1. O sistema recupera tanto os documentos contendo ***Bruce Dickenson*** quanto os documentos contendo a grafia correta.
  2. Recupera os docs. com a grafia correta só quando ***Bruce Dickenson*** não está no dicionário.
  3. Mesmo que (1), mas só quando a consulta original retorna um número de documentos menor que um limiar pré-especificado.
  4. Mesmo que (3) mas o sistema sugere consultas com a grafia correnta oo usuário, e.g., Você quis dizer: ***Bruce Dickinson?***

Correção de Palavras Isoladas

- Premissa fundamental – há um dicionário contendo a grafia correta.

- Duas escolhas básicas aqui:
  - Um dicionário padrão
    - E.g., dicionário Webster em inglês
  - O dicionário do corpus indexado
    - E.g., todas as palavras da Web
    - Nomes, acrônimos, etc.
    - (Incluindo os erros de ortografia)

Correção de Palavras Isoladas

- Dado um dicionário e uma sequência de caracteres Q, retorne as palavras no dicionário mais próximas de Q.
- O que significa “próximo”?
- Há diversas formas de medir proximidade
  - Distância de Edição (Distância Levenshtein)
  - Distância de Edição ponderada
  - Interseção em *n-gramas*

Distância de Edição

- Dadas duas strings S_1 e S_2, o número mínimo de operações para transformar uma na outra.
- Operações são tipicamente a nível de carácter
  - Inserção, Deleção, Substituição, (Transposição)
- E.g., distância de edição de ***dof*** para ***dog*** é 1
  - De ***cat*** para ***act*** is 2 (apenas uma transposição.)
  - De ***cat*** para ***dog*** é 3.
- Implementações disponíveis para várias linguagens de programação.

Distância de Edição Ponderada

- O peso de uma operação depende nos caracteres envolvidos:
  - E.g. é mais provável que ***m*** seja trocado por ***n*** do que por ***q***
  - Dessa forma, substituindo ***m*** por ***n*** representa uma distância de edição menor do que por ***q***
  - Pode ser formulado como um modelo probabilístico
- Requer uma matriz de pesos como entrada.

Usando Distância de Edição

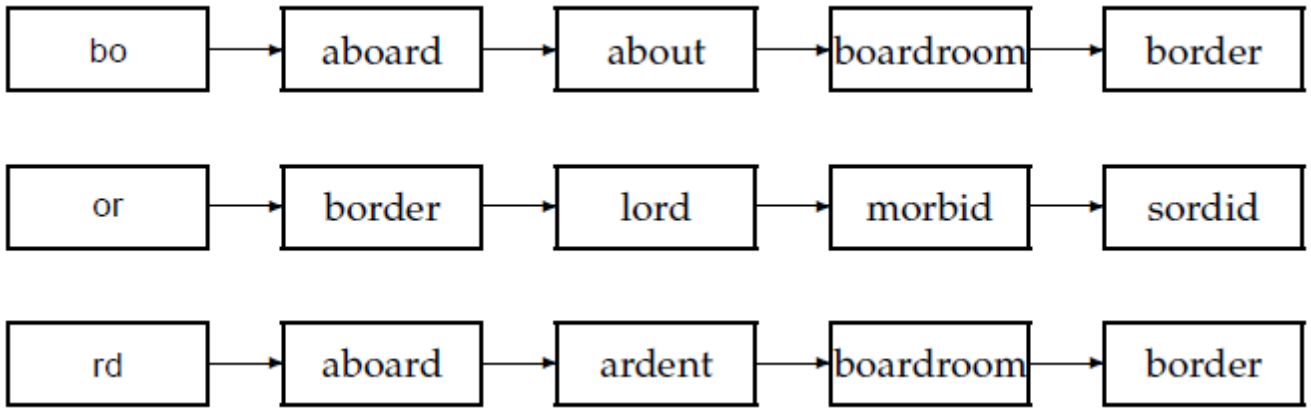
- Enumere todas as strings apresentando uma distância de edição (ponderada) mínima com a consulta (e.g., 2).
- Sugira os termos encontrados aos usuários como possíveis correções.
- Alternativamente,
  - Retorna todos os documentos relacionados as possíveis correções encontradas ... devagar
  - Usa a correção única mais provável (e.g., menor distância de edição)
- As alternativas tiram o poder de decisão do usuário, mas poupam eventuais rodadas de interações.

Distância de Edição: Problemas

- Dada uma consulta com erro de grafia – calcula-se a distância de edição para cada termo do dicionário?
  - Caro e devagar
  - Alternativa?
- Como restringimos o conjunto de candidatos do dicionário?

Interseção de n-gramas

- Enumere todas as *n*-gramas na string de consulta assim como no dicionário.
- Use o índice *n*-grama para recuperar todos os termos do dicionário casando com qualquer uma das *n*-gramas da consulta.
- Índice n-grama para a consulta “bord”.



Uma opção: Coeficiente Jaccard

- Sejam *X* e *Y* dois conjuntos; então o C.J. é

$$\left|X \cap Y\right| / \left|X \cup Y\right|$$

- Igual a 1 quando *X* e *Y* possuem os mesmos elementos e zero quando são disjuntos
- *X* e *Y* não precisam ser do mesmo tamanho
- Sempre retorna entre 0 e 1
  - Escolha limiar para decidir casamento
  - E.g., se C.J. > 0.8, declare casamento

Correção Sensível ao Contexto

- Precisa do contexto adjacente.
- Primeira idéia: recupere termos do dicionário próximos a cada termo de consulta.
- Agora tente todas as frases resultantes com uma palavra “fixa” por vez
  - ***flew from Salvador***
  - ***fled form Salvador***
  - ***flea form Salvador***
- **Correção baseada no nr. de resultados:** Sugira a alternativa que tem o maior número de resultados.

Exercício

- Suponha que ***“flew form Salvador” possui*** 7 alternativas para flew, 19 para form, 3 para Salvador. Quantas frases “corretas” serão enumeradas?

Complicações da Correção Ortográfica

- Enumeramos diversas alternativas para “Você quis dizer:?”
- Qual apresentar ao usuário?
- Heurísticas
  - A alternativa que gerar maior número de resultados
  - Análise de logs de consulta
- Correção ortográfica é cara computacionalmente
  - Rodar somente nas consultas retornando poucos documentos?

CONSULTAS FONÉTICAS

Soundex

- Classe de heurísticas para expandir consultas em equivalencias fonéticas
  - Específico da linguagem – principalmente para nomes
  - E.g., ***chebyshev*** $\rightarrow$ ***tchebycheff***
- Inventedo para o censo nos E.U.A em 1918.

Soundex – Algoritmo típico

- Transforma cada token a ser indexado numa forma reduzida de 4-caracteres.
- O mesmo com termos de consulta.
- Construa um índice de busca com as formas reduzidas.

Soundex – Algoritmo típico

1. Mantenha a primeira letra da palavra.
2. Mude todas as occorrências das seguintes letras para '0' (zero):
   'A', E', 'I', 'O', 'U', 'H', 'W', 'Y'.
3. Mude letras para dígitos da seguinte forma:

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

Soundex Continuação...

4. Remova todos os pares consecutivos de dígitos.
5. Remova todos os zeros da string resultante.
6. Complete a string com zeros, caso menor que 4, e retorne a string da seguinte forma <uppercase letter> <digit> <digit> <digit>.

E.g., ***Herman*** pode ser escrito como H655.

E quanto a ***hermann***?

Soundex

- Soundex é o algoritmo clássico disponível em muitos SGBD (Oracle, Microsoft, ...)
- Quão útil?
- Não muito – para recuperação da informação

O que temos até agora?

- Índices posicionais com ponteiros de salto
- Índices curinga
- Correção Ortográfica
- Soundex

Recursos

- Livro Texto cap. 3
- Recuperação ortográfica eficiente:
  - K. Kukich. Techniques for automatically correcting words in text. ACM Computing Surveys 24(4), Dec 1992.
  - J. Zobel and P. Dart. Finding approximate matches in large lexicons. Software - practice and experience 25(3), March 1995. http://citeseer.ist.psu.edu/zobel95finding.html
  - Mikael Tillenius: Efficient Generation and Ranking of Spelling Error Corrections. Master's thesis at Sweden's Royal Institute of Technology. http://citeseer.ist.psu.edu/179155.html
- **Leitura interessante e fácil sobre correção ortográfica:**
  - Peter Norvig: How to write a spelling corrector

  http://norvig.com/spell-correct.html