

Estudos de caso:

- Jusbrasil
- Americanas.com
- OLX

Como combinar features ?

- Costumava-se normalizar resultados e assumir independência entre fontes (até a década de 90)
- Exemplos: Combinações lineares ou tratamento probabilístico
- Fala-se em percentual de impacto para cada fonte (não há artigos descrevendo ideia)
- Problema é mais complicado...

Problemas ligados à combinação

- Fontes nem sempre são independentes
- Diferenças entre pesos pode não ser linear
- Combinação pode mudar de acordo com tipo de consulta
- Afeta métodos de poda
- Como utilizar evidências independentes de consulta (Indegre, Pagerank, Nivel de URLs e etc...) ?

Problemas ligados à combinação

- Sistemas onde respostas têm estruturas distintas e representam objetos distintos.
 - Ex.: Sistemas como Méliuz, Ifood, UberEats têm lojas e produtos como resultados de busca
 - OLX e Ebay têm produtos com estruturas completamente distintas
 - Jusbrasil combina redes sociais, jurisprudência, leis, diários e etc...
 -

Alternativas de solução

- Utilizar algum método de combinação adhoc
- Utilização de métodos de aprendizagem automática
- Exemplos: GP, LambdaMart, Deep Learning

Métodos Adhoc

- Combinação linear dos graus de relevância dos documentos a uma consulta

$$SU(q,d) = P1 * A(q,d) + P2 * B(q,d) + P3 * C(q,d)$$

- Modelar cada fonte de evidência como uma probabilidade independente de relevância dos documentos

$$SU(q,d) = 1 - ((1 - A(q,d)) * (1 - B(q,d)))$$

*Silva, Ilmério, Ribeiro-Neto, Berthier, Calado, Pavel., Moura, Edleno, & Ziviani, Nivio. Link-based and content-based evidential information in a belief network model. SIGIR (2000)

Métodos Adhoc

- Vantagem

- Não precisam de treino

- Desvantagem

- ◆ Não utilizam propriedades das fontes de evidências para descobrir a melhor maneira de combinar
 - Dependências entre diferentes fontes de evidência
 - Variações na distribuição dos escores de cada evidência

Métodos de Aprendizagem de máquina

- Utilizam uma base de treino para “aprender” a melhor forma de combinar evidências
- Necessidade de treino é principal desvantagem
- Exemplo: GP, Redes Neurais, SVMRank e etc...
- Conhecidos como: Learn to rank, LTR ou L2R na literatura.

L2R (abordagem em 2 passos)

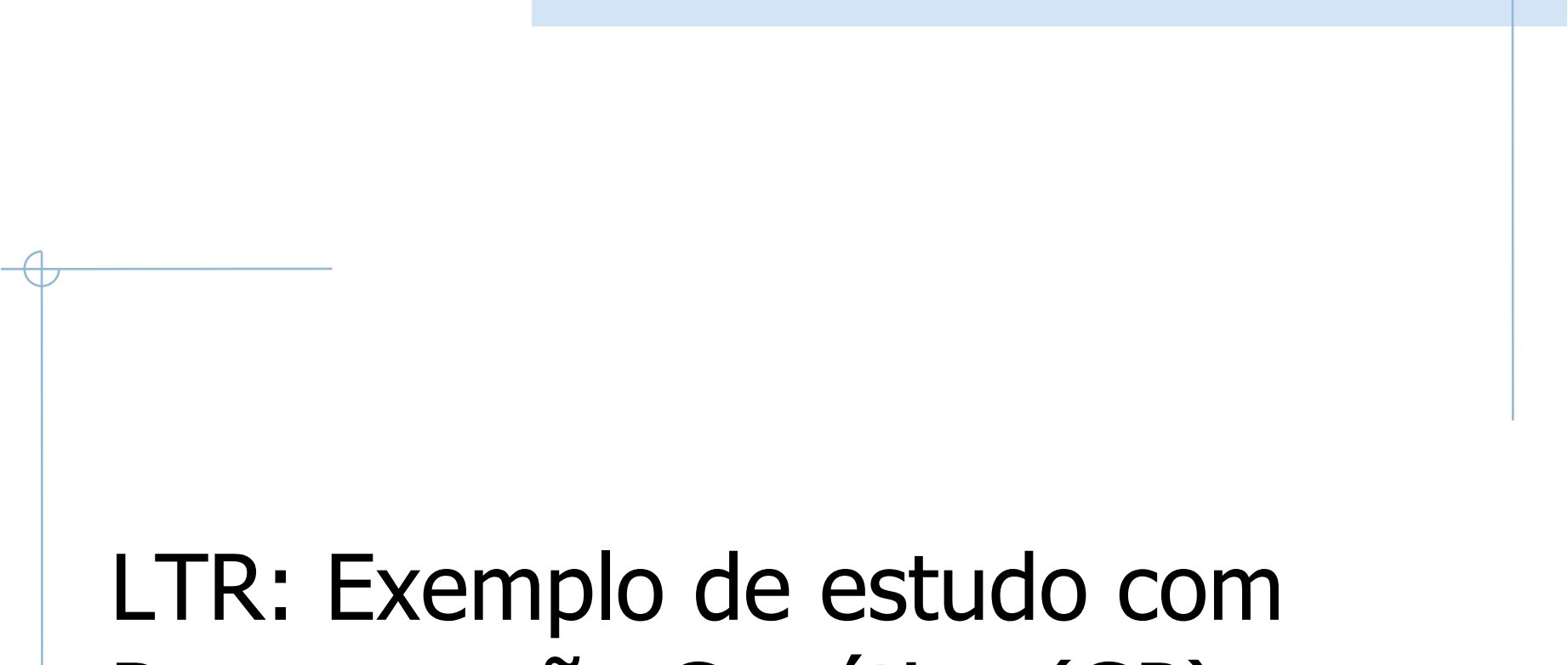
- Execute algum método mais leve de ranking para obter conjuntos iniciais de resposta.
- Toma-se os top-k resultados, por exemplo, os top-1000.
- Método base deve ter o melhor ranking possível, mas deve ser leve

L2R (abordagem em 2 passos)

- Aplique o L2R para reordenar os resultados do topo da resposta.
- Utilize os cuidados convencionais em ML:
 - Ter consultas de treino, validação e teste
 - Buscar conjunto de consultas que seja representativo do que será submetido ao sistema

L2R: Melhores alternativas atualmente

- **Lambdamart:** resultados muito bons e boa velocidade no treino e, principalmente, no teste
- Há vários outros métodos baseados em **redes neurais (neural ranking models)**: Resultados promissores, mas não muito superiores(ou até inferiores) aos do Lambdamart
- **GP:** pouco aceito na literatura, mas apresenta resultados bastante competitivos e é extremamente simples de usar e entender.



LTR: Exemplo de estudo com Programação Genética (GP)

Combinação Utilizando GP

- Programação Genética pode ser usada para descobrir funções de combinação que:
 - ◆ aumentem a qualidade das respostas de sistemas de busca
 - ◆ utilizem as propriedades contidas nas evidências de relevância (features) para gerar boas funções
 - ◆ generalizem a tarefa de combinação para um número qualquer de evidências de relevância
 - ◆ Há várias libs prontas para usar GP

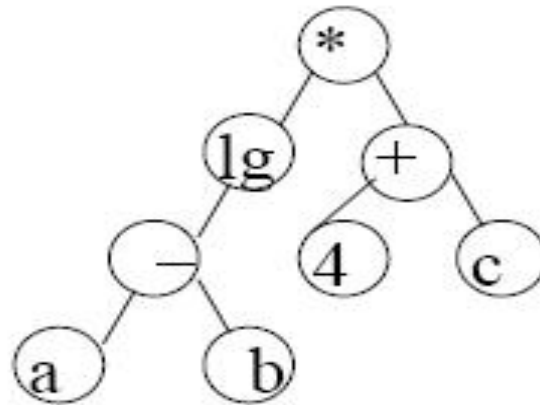
Programação Genética (PG)

- Baseado no conceito de *Seleção Natural*
- Técnica de aprendizagem automática (*machine learning*)
- Evolução de indivíduos (soluções para determinado problema)

Programação Genética

- Por que usar PG?
 - Tem sido usada com sucesso em várias aplicações, inclusive na derivação de funções de ranking.

Programação Genética



$$\text{Arvore}(a,b,c)=(\lg(a-b))*(4+c)$$

- Terminais: a,b,c e 4
- Funções: multiplicação, subtração, adição e logarítmo

Exemplo

- Aplicação para gerar ranking em máquinas de busca para a Web utilizando Gp

*Silva, Thomaz. P. C., de Moura, Edleno S., Cavalcanti, João. M. B., da Silva, Altigran. S., de Carvalho, Marcos. G., & Gonçalves, M. A. Information Systems (2009)

PG para Combinação de Evidências

■ Entrada

■ Consultas pré-avaliadas

- ◆ Processa consulta
- ◆ Avalia as 50 primeiras respostas do topo de cada um dos ranks gerados por cada evidência

■ Fases da combinação

- Treinamento
- Validação

Configuração dos experimentos

- Evidências de relevância (features) utilizadas
 - Texto presente nas páginas
 - ◆ Cosseno vetorial aplicado sobre os textos
 - Textos de âncora dos documentos
 - ◆ Cosseno vetorial aplicado sobre os textos de âncora
 - Estruturas de ligação entre as páginas da Web
 - ◆ Pagerank

Configuração dos experimentos

- Classes de consultas

- As consultas foram divididas segundo tipo de informação procurada e popularidade dos documentos

- ◆ Consultas por tópico de informação (informacionais)

- Populares e não-populares

- ◆ Consultas por sites específicos (navegacionais)

- Populares e não-populares

- Classificadas manualmente

Configuração dos experimentos

- Função de Fitness
 - Consultas informacionais: BPref
 - Consultas Navegacionais - MRR

Configuração dos experimentos

- Coleção

- Base de dados do *TodoBr*
- Mais de 12 milhões de páginas

- Fases dos experimentos

- Treinamento
- Validação
- Teste

Configuração dos experimentos

■ Informacionais

- ◆ 62 consultas extraídas do log do *TodoBr* composto por mais de 11 milhões de consultas
 - 70% utilizadas para o treinamento
 - 30% utilizadas para validação
 - Executamos 20 evoluções e escolhemos a que obteve melhor resultado na validação
 - 30 consultas para fase de teste em que foram avaliadas os 10 documentos do topo de cada evidência, inclusive as funções de combinação estudadas
- ◆ Os resultados apresentados se referem ao conjunto de teste

Configuração dos experimentos

- Navegacionais
 - 90 consultas extraídas do log
 - ◆ 50% para treino
 - ◆ 20% para validação
 - ◆ 30% para teste
- Executamos 20 evoluções e escolhemos a função que obteve melhor resultado quando aplicada à função de fitness para passar a fase de testes

Configuração dos experimentos

- Comparação de desempenho
 - Modelo de combinação de evidências por redes Bayesianas proposto por Silva
 - Modelo proposto por Craswell (baseado em treino)
 - Melhor combinação Linear (combinação linear utilizando treino)

Experimentos(1)

■ Consultas Informacionais

Query Type	Method	RankEff	Bpref-10	MAP
Informational Popular	GP	0.744	0.562	0.485
	BN	0.406	0.393	0.361
	BLC	0.731	0.563	0.312
	SIGM	0.704	0.520	0.462
Informational Non-Popular	GP	0.724	0.538	0.401
	BN	0.476	0.245	0.237
	BLC	0.648	0.365	0.312
	SIGM	0.632	0.336	0.300

Query Type	Evidence	RankEff	Bpref-10	MAP
Informational Popular	Text	0.715	0.414	0.336
	Anchor	0.632	0.348	0.259
	Pagerank	0.320	0.117	0.097
Informational Non-popular	Text	0.726	0.549	0.444
	Anchor	0.511	0.203	0.159
	Pagerank	0.317	0.190	0.134

Experimentos(2)

■ Consultas Informacionais

Query Type	Method	RankEff	Bpref-10	MAP
Informational Popular	GP	0.744	0.562	0.485
	BN	0.406	0.393	0.361
	BLC	0.731	0.563	0.312
	SIGM	0.704	0.520	0.462
Informational Non-Popular	GP	0.724	0.538	0.401
	BN	0.476	0.245	0.237
	BLC	0.648	0.365	0.312
	SIGM	0.632	0.336	0.300

Query Type	Evidence	RankEff	Bpref-10	MAP
Informational Popular	Text	0.715	0.414	0.336
	Anchor	0.632	0.348	0.259
	Pagerank	0.320	0.117	0.097
Informational Non-popular	Text	0.726	0.549	0.444
	Anchor	0.511	0.203	0.159
	Pagerank	0.317	0.190	0.134

Experimentos(3)

- Consultas Navegacionais

Query Type	Method	MRR
Navigational Popular	GP	0.920
	BN	0.405
	BLC	0.581
	SIGM	0.479
Navigational Non-Popular	GP	0.803
	BN	0.408
	BLC	0.367
	SIGM	0.325

Query Type	Evidence	MRR
Navigational Popular	Text	0.153
	Anchor	0.178
	Pagerank	0.266
Navigational Non-popular	Text	0.209
	Anchor	0.364
	Pagerank	0.178

Experimentos(4)

- Consultas navegacionais

Query Type	Method	MRR
Navigational Popular	GP	0.920
	BN	0.405
	BLC	0.581
	SIGM	0.479
Navigational Non-Popular	GP	0.803
	BN	0.408
	BLC	0.367
	SIGM	0.325

Query Type	Evidence	MRR
Navigational Popular	Text	0.153
	Anchor	0.178
	Pagerank	0.266
Navigational Non-popular	Text	0.209
	Anchor	0.364
	Pagerank	0.178

Análise das Fórmulas

t-texto; a-âncora; p-pagerank

■ Informacionais Populares

$$Comb(a, p, t) = \begin{cases} t(3t + 2tp^2 + 2ap^2 + 6a + 4p) + 8ap + \\ p \ln(p)(6t^2p^2 + 2tap^2 + 9t + 3ta + 12tp + 4ap) \end{cases}$$

Informacionais não-populares

$$Comb(a, p, t) = t^5(2p + 6t^2p + 4tp + 3tp + 2p^2 + 6t^3 + 6t^5 + 4t^4 + 3t^4p + 2t^3p)$$

Análise das Fórmulas

- Navegacionais populares

$$Comb(a, p, t) = \begin{cases} ap^2(8 + 7p - p^2 - a - \\ ap \ln(a)(8p - p^2 - \ln(p - 8 \ln(p - 8) + a \ln(p - 8)) - ap + 6)) \\ + ap(6 - \ln(p - 8 \ln(p - 8) + a \ln(p - 8)))) \end{cases}$$

- Navegacionais não-populares

$$Comb(a, p, t) = a^3 p(2t + p)$$

LambdaMart

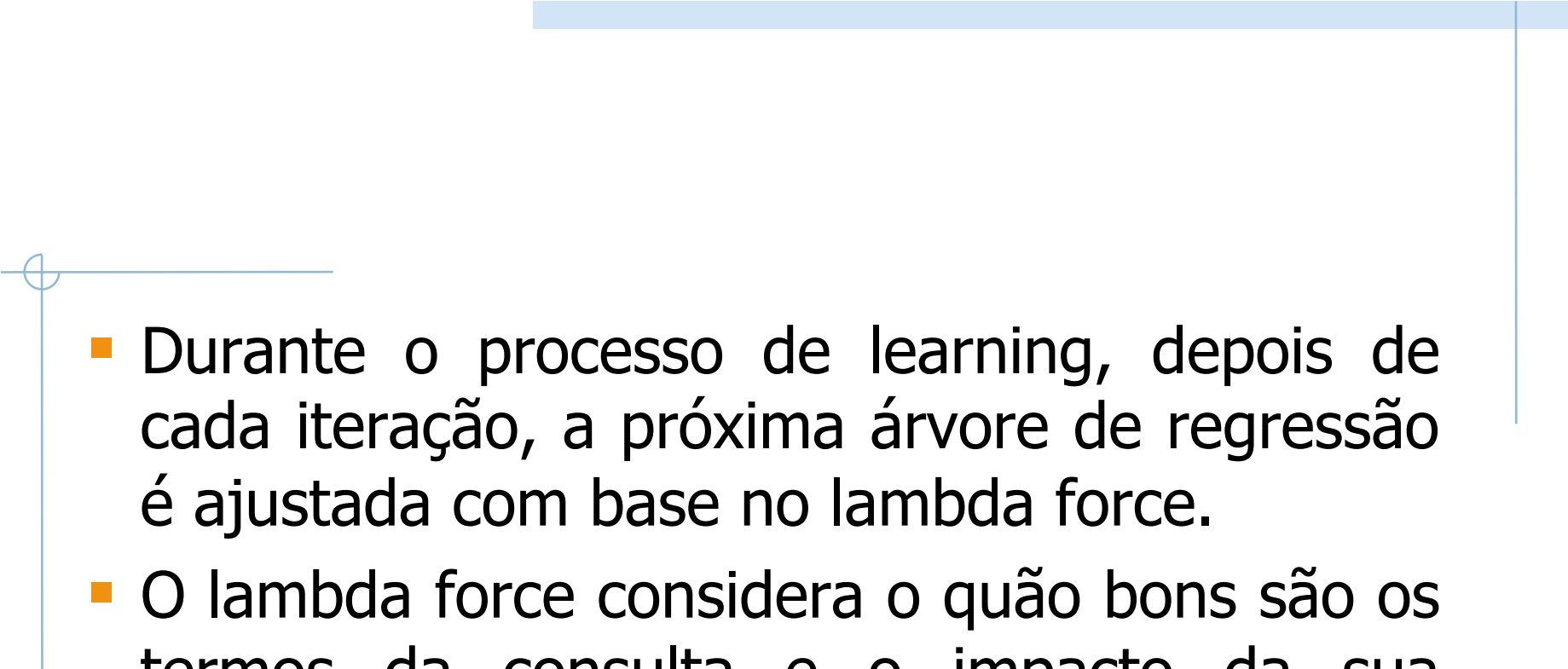
- Desenvolvido por Burges et al, microsoft research (ver relatório técnico: From RankNet to LambdaRank to LambdaMART: An Overview, *MSR-TR-2010-82*)
- Combinação do método MART, que utiliza boosted trees, and LambdaRank, baseado em redes neurais. Você aprende mais sobre essas técnicas na disciplina de ML
- Assim como Gp, produz uma função de score que pode ser usada na hora de computar o ranking final.

LambdaMart

- Será mais detalhado no seminário sobre LTR

- Cria N árvores.
- Usa os scores computados para ordenar os documentos
- Podemos estimar a diferença em NDCG obtida pela troca da ordem de cada par de documentos (d_j, d_k) num resultado (ΔNDCG) , sempre que o score de relevancia d_j for maior que o score de d_k .

- Δ NDCG é usado para calcular o λ e o peso associado a cada consulta. ($\lambda[d, q]$ e $w[d, q]$, respectivamente)
- Finalmente é realizado um fitting para as árvores de regressão e o modelo é atualizado.
- As árvores de regressão mapeiam consultas e features dos documentos.

- 
- Durante o processo de learning, depois de cada iteração, a próxima árvore de regressão é ajustada com base no lambda force.
 - O lambda force considera o quão bons são os termos da consulta e o impacto da sua ausência para cada documento.

Muitos trabalhos com machine learning nos últimos anos

- Metrikov, P., Wu, J., Anderton, J., Pavlu, V., & Aslam, J. A. A modification of lambdamart to handle noisy crowdsourced assessments (ICTIR, 2013)
- Wang, L., Bennett, P. N., & Collins-Thompson, K. Robust ranking models via risk-sensitive optimization (SIGIR, 2012). ***Risk sensitive = tomar cuidado com features e modificações que possam tornar o ranking mais instavel***

Mais referências

- Hu, Z., Wang, Y., Peng, Q., & Li, H. Unbiased LambdaMART: An Unbiased Pairwise Learning-to-Rank Algorithm. (WWW, 2019)
- Guo, J., Fan, Y., Pang, L., Yang, L., Ai, Q., Zamani, H., ... & Cheng, X. Croft, B.. A deep look into neural ranking models for information retrieval (IPM, 2019): survey recente que compara métodos de L2R que usam redes neuronais.

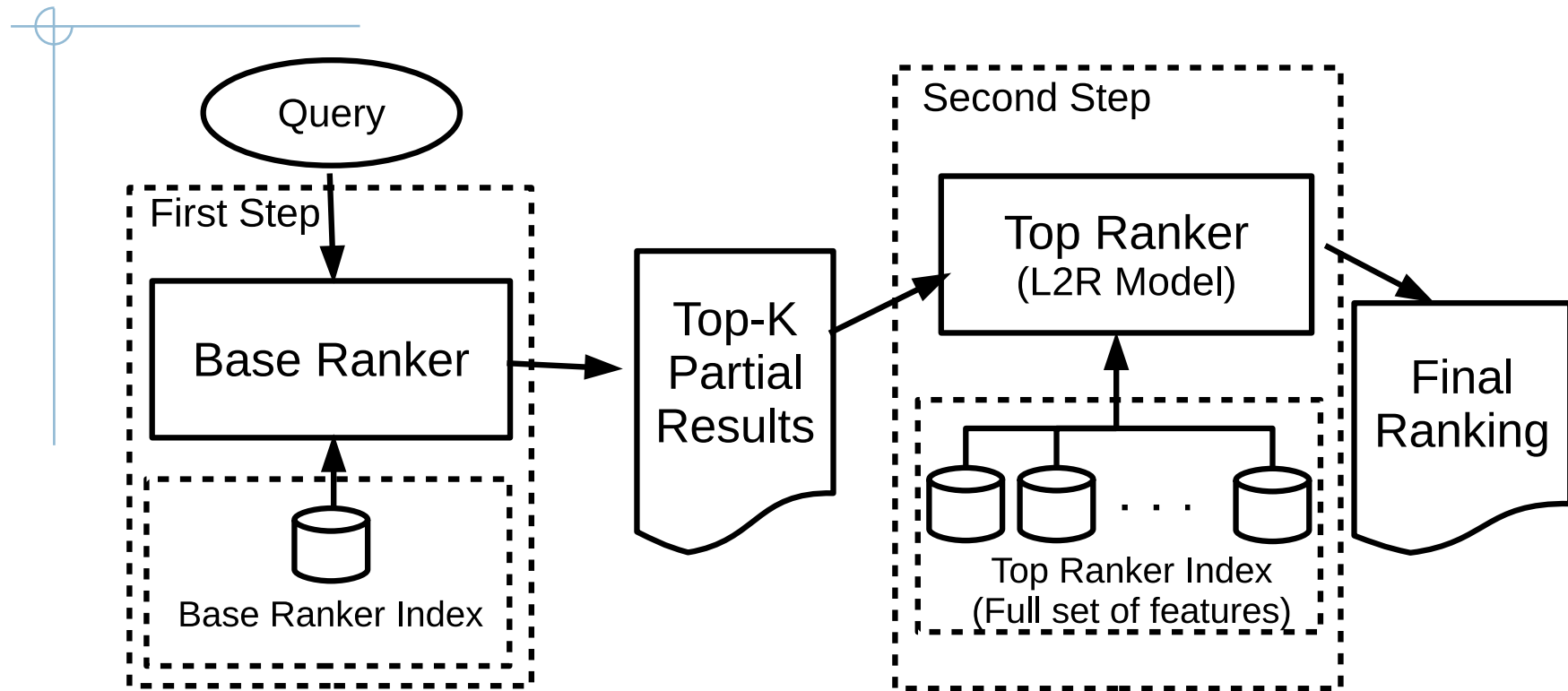
Trabalho em andamento aqui na UFAM

- Aluna de doutorado tem artigo submetido que mostra como usar LambdaMart para produzir resultados superiores a outros métodos de LTR existentes na literatura e, ao mesmo tempo, reduzir tempo de processamento.
- Tema de doutorado do MAX: teremos seminários mesmo após a disciplina para detalhar mais, todos serão convidados



Implementação de Sistemas de busca

Arquitetura típica



Estrutura de Dados: Arquivo Invertido

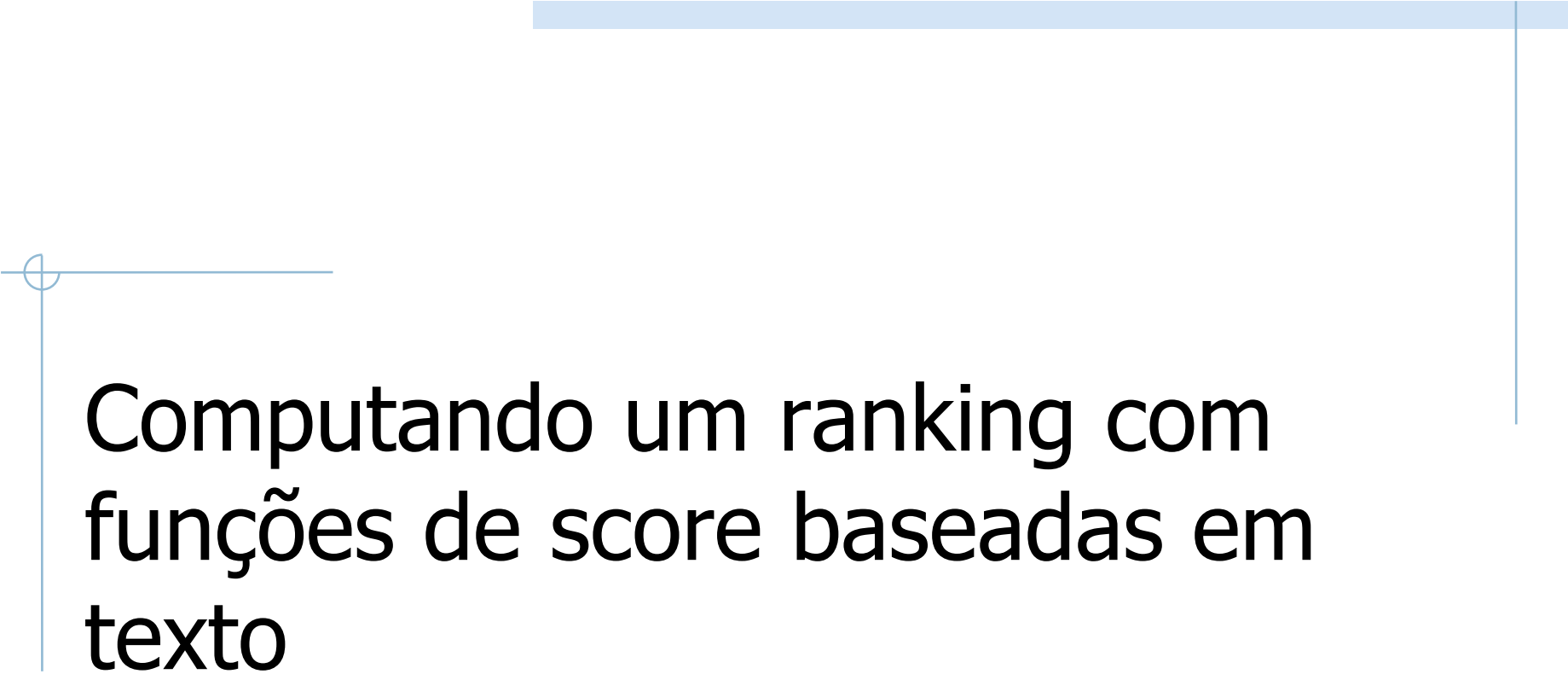
- Composto de:
 - Vocabulário - contém cada termo (t) distinto da coleção;
 - Listas Invertidas - Para cada termo da coleção há uma lista invertida que indica a frequência com que o termo ocorre em cada documento da coleção;

Ranking básico

- Primeiro corte nos resultados, tipicamente usa algum modelo clássico, como o BM25 ou o vetorial, para calcular um ranking inicial.
- Topo do resultado é passado para um método mais sofisticado que leva em conta todas as features (fontes de evidência de relevância, tais como links, cliques, personalização e etc...)

Posso calcular o ranking de uma vez só ?

- Sim, arquitetura mostrada é a que indicaria hoje, mas pode-se pensar em aplicar diretamente a função de learning sobre toda a base para cada consulta
- Isso pode trazer desafios ligados a desempenho.
- Estratégia mencionada aqui é bastante citada na literatura como uma boa alternativa



Computando um ranking com
funções de score baseadas em
texto

Calculando funções de score em modelos clássicos

- Funções de score usadas são tipicamente baseadas em valores como frequência do documento na coleção (**tf**), tamanho do documento (**norma**) e raridade das palavras na coleção (**idf**)
- **Vide BM25, VSM ou LM de Zhay e Laffert, entre outros.**

Indexadores

- Componente mais simples de uma máquina de busca
- Em geral funcionam offline, atualizando os índices de tempos em tempos em uma cópia que não é acessada pelo usuário
- Pode ser interessante ter um indexador integrado ao processador para atualizar informações online!

IDF e a Norma

- IDF e norma são valores independente de consulta, devem ser calculados em tempo de indexação
- IDF é um vetor de valores numéricos do tamanho do vocabulário da coleção. Tipicamente tem milhões de entradas. Em sistemas como o Google, deve ter bilhões.
- Norma é um vetor com um entrada para cada documento da coleção. Se tenho 1 bilhão de documentos, vetor terá 1 bilhão de entradas.

IDF e Norma

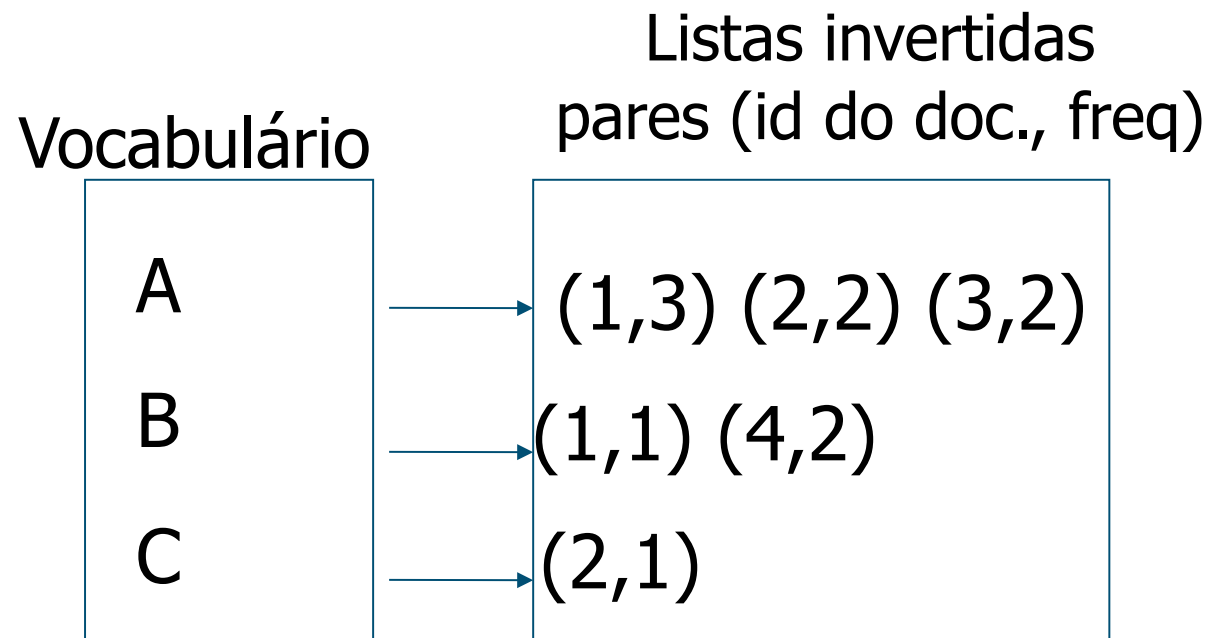
- Podem ser recalculados apenas de tempos em tempos para termos e documentos onde os valores já foram computados
- Novos termos e novos documentos devem ter suas normas calculadas
- Uso de tamanho do documento como norma gera grande economia de tempo na indexação, pode ser considerado no caso do VSM. LM e MB25 usam tamanho!

TF

- Os valores de TF são mapeamentos entre palavras do vocabulário (palavras distintas de coleção) e cada documento da coleção
- Forma uma matriz esparsa, que é processada a cada consulta para calcular o score de cada documento que contém pelo menos uma palavra da consulta
- Precisamos de uma boa estrutura de dados para representar tal mapeamento

Estruturas de Dados: Arquivo Invertido

D1	A A A B
D2	A A C
D3	A A
D4	B B



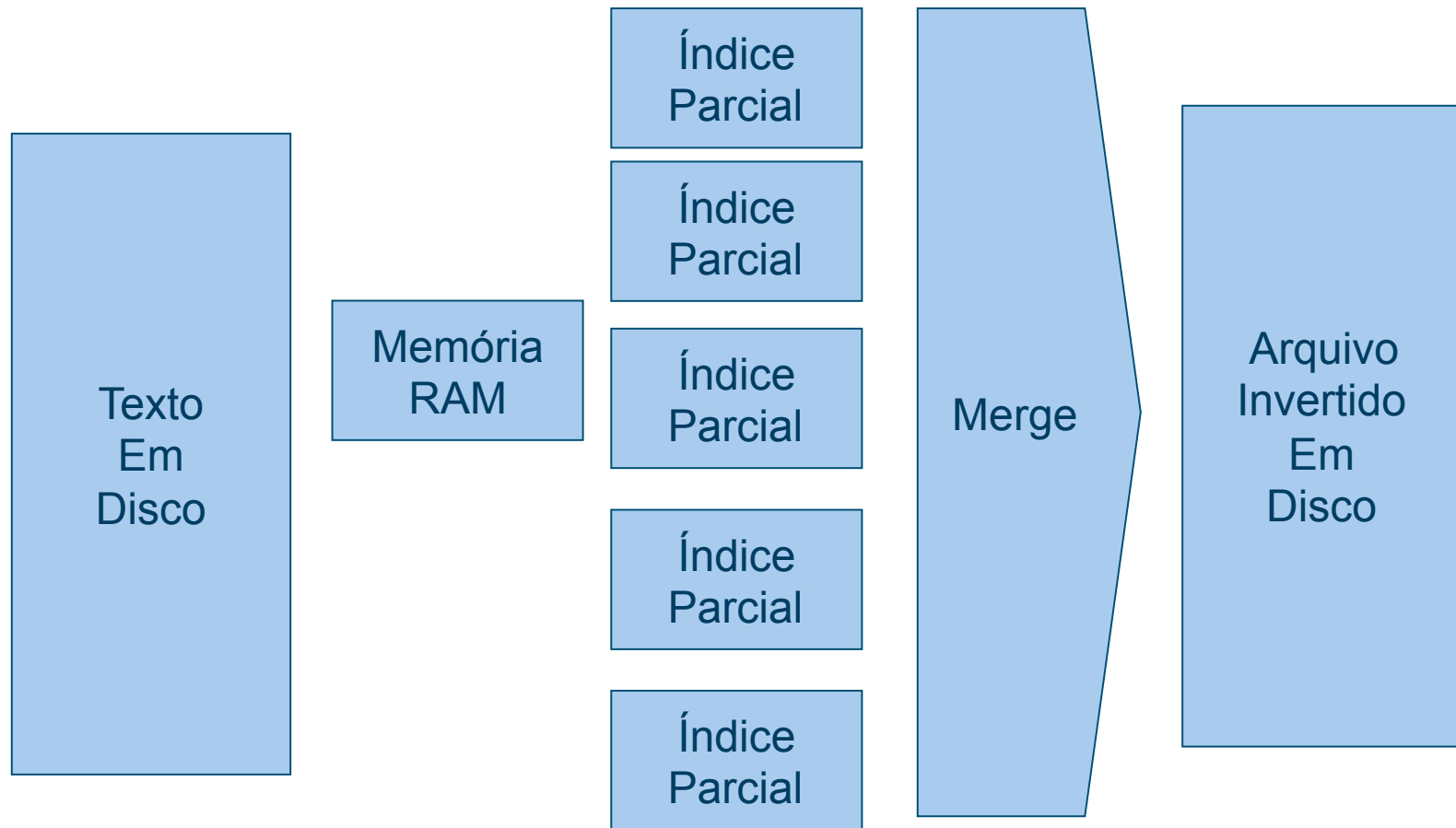
Arquivos invertidos

- Para cada palavra (termo) encontrada na coleção (conjunto de documentos), contém a lista de documentos onde a mesma ocorre $\langle d1, freq \rangle \langle d2, freq \rangle \dots$
- Tal lista é conhecida como **lista invertida**
- O índice completo é conhecido como arquivo invertido

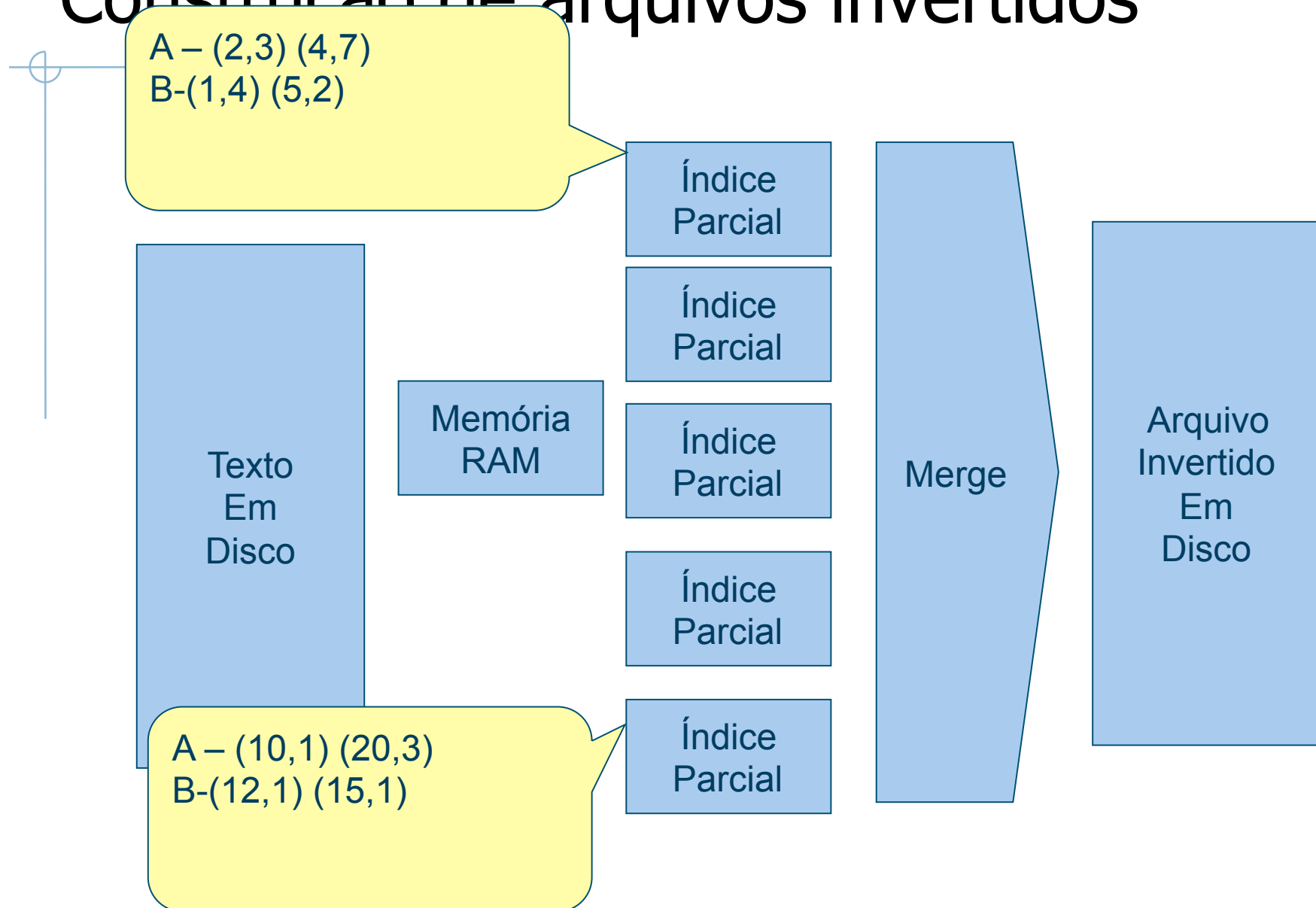
Arquivos Invertidos

- Construção:
 - Fácil de construir quando há memória RAM suficiente;
 - Algoritmos mais sofisticados são necessários quando a base de dados é muito grande;

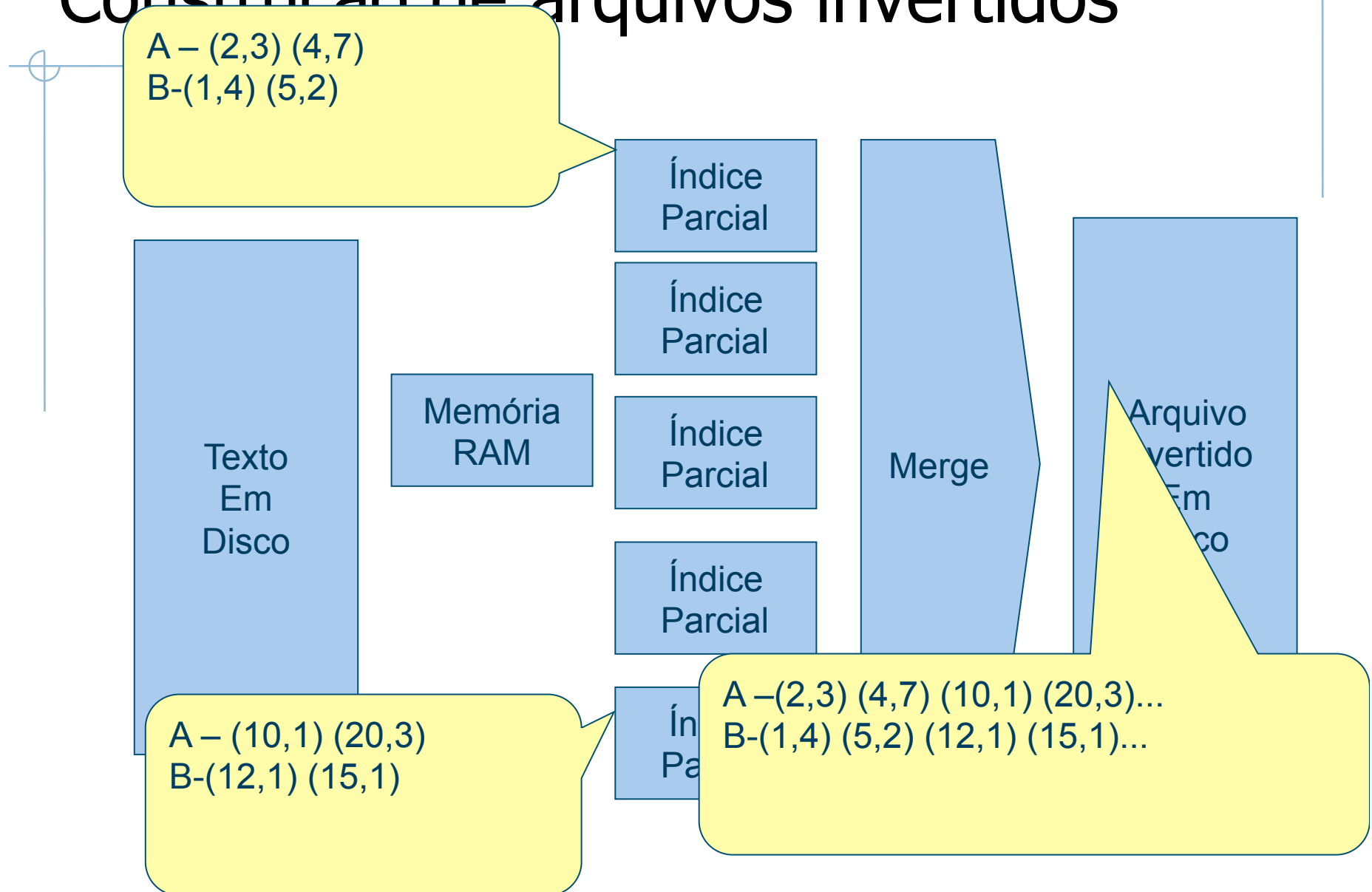
Construção de arquivos invertidos



Construção de arquivos invertidos



Construção de arquivos invertidos



Merge

- Merge utiliza memória como buffer para acelerar o processo de indexação
- Compressão de dados pode ser usada para reduzir o tamanho dos índices e assim aumentar a velocidade de indexação



Compressão de Listas Invertidas

Sim, elas são armazenadas em geral são comprimidas

Codificação de Inteiros

- Vários estudos foram apresentados por Moffat, Zobel e outros na década de 90
- Décadas de 60 e 70, mas úteis até hoje:
 - Unário
 - Elias γ
 - Elias δ
 - Golomb
- Mais recentes (quase padrões hoje)
 - Métodos do tipo PFor: PFOR = Patched Frame of Reference

Codificação de Inteiros

- Código Unário
- Seqüência de 1s seguida de um zero. x é codificado em $(x-1)$ 1s seguidos de 0

1	0
2	10
3	110
4	1110
5	11110

Uso Prático

- Os códigos unários normalmente não são utilizados diretamente na compressão de seqüências números, mas são empregados como parte de outros métodos de codificação.
- A codificação unária é empregada indiretamente na compressão de listas invertidas...

Codificação de Elias

- Elias γ :

- X é representado como $1 + \text{PISO}(\log x)$ em unário seguido de um código binário com $\text{PISO}(\log x)$ bits com o valor $X - 2^{\text{PISO}(\log x)}$ em binário.

Comparação Unário e Elias γ

Valor	Unário	Elias γ
1	0	0
2	10	100
3	110	101
4	1110	11000
5	11110	11001
6	111110	11010
7	1111110	11011

x é representado como $1 + \text{PISO}(\log x)$ em unário seguido de um código binário com $\text{PISO}(\log x)$ bits com o valor $X - 2^{\text{PISO}(\log x)}$ em binário.

Elias γ

- Codifique o número 25 utilizando Elias γ

Codificação de Elias

- Elias δ :

- X é representado como $1 + \text{PISO}(\log X)$ em Elias γ seguido de um código binário com $\text{PISO}(\log X)$ bits com o valor $X - 2^{\text{PISO}(\log x)}$ em binário.

Codifique o número 4 utilizando Elias δ

Comparação Unário e Elias δ

Valor	Unário	Elias δ
1	0	0
2	10	1000
3	110	1001
4	1110	10100
5	11110	10101
6	111110	10110
7	1111110	10111

x é representado como $1 + \text{PISO}(\log X)$ em Elias γ seguido de um código binário com $\text{PISO}(\log X)$ bits com o valor $X - 2^{\text{PISO}(\log x)}$ em binário.

Golomb

- Uma família de codificadores parametrizada
- Dado um parâmetro b , um número x é codificado como:
 - $q+1$ em unário, onde $q = \text{PISO}((x-1)/b)$
 - $r = x - qb - 1$ em binário, usando $\text{PISO}(\log b)$ ou $\text{TETO}(\log b)$ bits

Número de bits em r

B=6

00
01
100
101
110
111

B=7

00
010
011
100
101
110
111

B=8

000
001
010
011
100
101
110
111

B=12

000
001
010
011
1000
1001
1010
1011
1100
1101
1110
1111

Compressão de listas invertidas

- As frequências são codificadas diretamente utilizando-se Elias-Gama
- Armazena-se apenas as diferenças entre os documentos:
- Ex: $\langle 2, 1 \rangle \langle 4, 1 \rangle \langle 7, 2 \rangle$
 $\langle 2, 1 \rangle \langle 2, 1 \rangle \langle 3, 2 \rangle$
- As diferenças entre os documentos são codificadas com Elias-Delta ou Golomb
- Se listas estiverem ordenadas por documento, compressão melhora!!

Ganhos

- Frequências são armazenadas com uma média abaixo de 2 bits/símbolo
- Número dos documentos é armazenado em média com 6 bits/símbolo (se não ordenado por documento)
- Tamanho do índice é reduzido normalmente para 1/6 da representação convencional
- Velocidade aumenta de acordo com Hardware disponível.

Compressão FASTPFORDELTA (ICDE, 2013)

- Pacote de dados comprimindo uma determinada quantidade de números (tipicamente 128).
- 1 byte para b , 1 byte para $\max b$; 1 byte para C (número de exceções)
- C bytes para as posições das exceções

b	Max b	C	C bytes	Números de b bits	C Exceções
-----	---------	-----	-----------	---------------------	--------------

Compressão FASTPFORDELTA (LEMIRE et al, SPE, 2013)



1 b	2 Max b	3 C	4 C bytes	5 Números de b bits	6 C Exceções
---------------	-------------------	---------------	---------------------	-------------------------------	------------------------

1 **b**: tamanho em bits dos números codificados. Escolhido para representar os 90% menores números do bloco.

2 **Max b**: número de bits para guardar as exceções (número de bits do maior valor no bloco)

Ex: se o maior dos 90% é 30, pode-se escolher $b=5$.

3 **C**: número de exceções

4 **C bytes** que indicam as posições das exceções (bloco tem no máximo 256 números, logo cada posição cabe em 1 byte)

5 **Números de b bits**: cada (entre os 90%)

6 **C Exceções**: números que passaram de b bit

FastPforDelta

- Vantagem de poder descomprimir todos os números de um bloco em paralelo
- Muita velocidade!
- Comprime menos que outros códigos, como Elias, mas é muito efetivo

MILC (Wang et al, VLDB, 2017)

- Compara todo o histórico de métodos até então
- Uma nova variante de métodos pfor que autores advogam ser mais rápida e comprimir mais.
- Pequenos truques interessantes dão ganho em compressão
- Será explicada em seminário