ABORDAGENS DE PARTICIONAMENTO UTILIZANDO LOCALITY-SENSITIVE HASHING APLICADA A BUSCA HEURÍSTICA NA DETECÇÃO DE PLÁGIO EXTERNO

Michel Dias de Arruda

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientador: Geraldo Bonorino Xexéo

Rio de Janeiro
Setembro de 2019

ABORDAGENS DE PARTICIONAMENTO UTILIZANDO LOCALITY-SENSITIVE HASHING APLICADA A BUSCA HEURÍSTICA NA DETECÇÃO DE PLÁGIO EXTERNO

Michel Dias de Arruda

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

Prof. Geraldo Bonorino Xexéo, D.Sc.

Prof. Eduardo Bezerra da Silva, D.Sc.

Prof. Fellipe Ribeiro Duarte, D.Sc

Prof. Geraldo Zimbrão da Silva, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
SETEMBRO DE 2019

Arruda, Michel Dias de

Abordagens de Particionamento Utilizando Locality-Sensitive Hashing Aplicada a Busca Heurística na Detecção de Plágio Externo/Michel Dias de Arruda. – Rio de Janeiro: UFRJ/COPPE, 2019.

XV, 106 p.: il.; 29, 7cm.

Orientador: Geraldo Bonorino Xexéo

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2019.

Referências Bibliográficas: p. 82 – 88.

1. lsh. 2. partitions. 3. locality-sensitive hashing. 4. plagiarism detection. I. Xexéo, Geraldo Bonorino. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Dedicado a Deus
que me deu forças e sabedoria
para concluir com êxito esse
trabalho.

Agradecimentos

Agradeço primeiramente a Deus, por sempre me fortalecer e me guiar em todos meus objetivos.

A Laercio e Rosane, meus pais, por terem me gerado, educado, amado e auxiliado em toda vida.

A minha irmã Leticia, por sempre me incentivar a prosseguir.

A Airine Carmo, pelo companheirismo durante todo o curso, me incentivando e auxiliando.

Ao Professor Geraldo Xexéo, meu orientador, pela oportunidade e paciência na realização deste trabalho, que espero fazer jus a confiança dedicada a mim.

Ao Professor Fellipe Duarte, por aceitar estar presente na minha banca, pela amizade criada desde os tempos de Rural e pela paciência, orientação, confiança e incentivo durante todo o mestrado.

Ao Professor Eduardo Bezerra, por aceitar estar presente na minha banca e por toda a contribuição feita durante o desenvolvimento desse trabalho.

Ao professor Geraldo Zimbrão, por aceitar fazer parte da banca examinadora, abrindo mão de seus compromissos e sacrificando o tempo de sua agenda.

A todos os colegas de pesquisa do LINE, em especial a Hugo Rebelo e Joaquim Viana, pela amizade, pelo trabalho em conjunto e ajuda no desenvolvimento desse trabalho.

Ao Smartbank, pelo apoio desde a minha mudança para São Paulo.

Aos meus amigos Egberto Caetano, Ricardo Luiz, Geovani Celebrim, Débora Alcântara, Desirée Oliveira, Monique Marinari e Rodrigo Guarino pelo apoio e incentivo na fase final da minha dissertação.

Aos meus amigos de Rural, que mesmo de longe, me incentivaram sempre.

Ao Programa de Engenharia de Sistemas e Computação, em especial a linha de Engenharia de Dados e Conhecimento, pela oportunidade a mim dada para o desenvolvimento desse trabalho.

E a todos aqueles que me auxiliaram de alguma forma durante o curso.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

ABORDAGENS DE PARTICIONAMENTO UTILIZANDO LOCALITY-SENSITIVE HASHING APLICADA A BUSCA HEURÍSTICA NA DETECÇÃO DE PLÁGIO EXTERNO

Michel Dias de Arruda

Setembro/2019

Orientador: Geraldo Bonorino Xexéo

Programa: Engenharia de Sistemas e Computação

A Detecção de Plágio em Linguagem Natural (NLPD) visa identificar a evidência textual que um documento contém plágio, para gerar uma lista de documentos que são plagiados para posterior avaliação humana. A detecção de plagio externo (EPD) é uma tarefa da NLPD, a qual um conjunto de documentos está disponível para consulta por plágio. EPD é formada de algumas etapas, dentre elas a Busca Heurística (HR), que é a etapa de EPD que visa recuperar um conjunto de documentos candidatos a plágio de um grande corpus, reduzindo o carga de trabalho das etapas posteriores do EPD. A etapa de Busca Heurística é uma tarefa de Recuperação de Informação (IR) e contém duas subtarefas: a indexação e a busca. Foram propostos dois métodos de particionamento, das permutações e do vocabulário, com o objetivo de tornar mais rápida a execução das subtarefas de IR. Ambos utilizam Locality-Sensitive Hashing (LSH) e são baseados no conceito matemático conhecido como partição de um conjunto. O particionamento de qualquer conjunto pode gerar resto, e a partir disso foram propostas as estratégias de tratamento Remainder at End (RaE), Remainder at Cell (RaC) e Distributed at Cell (DaC). Nos dois métodos de particionamento, RaE, RaC e DaC foram aproximadamente 101% mais rápidos que o estado da arte da IR na subtarefa de busca. Além disso, ambos alcançaram melhores resultados na indexação de documentos, extração de consultas e tempo de recuperação quando comparados com o *baseline* padrão do LSH, o MinMax.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

PARTITIONING APPROACHES USING LOCALITY-SENSITIVE HASHING APPLIED TO HEURISTIC RETRIEVAL IN EXTERNAL PLAGIARISM DETECTION

Michel Dias de Arruda

September/2019

Advisor: Geraldo Bonorino Xexéo

Department: Systems Engineering and Computer Science

Natural Language Plagiarism Detection (NLPD) aims to seek textual evidence of plagiarism in documents, in order to generate a list of candidate documents of being plagiarized, to further be analysed by humans. External plagiarism detection (EPD) is a NLPD task in which a set of documents is available to be queried, seeking for plagiarism. DPE is comprised of a few steps, one of them being the Heuristic Search (HR), which is the EPD stage that retrieves a set of plagiarism candidate documents from a large corpus, reducing the workload of the later stages of the EPD. The HR stage is an Information Retrieval (IR) task, and comprises two subtasks, namely, Indexing and Source Retrieval. In order to speed up the execution of IR subtasks, two partitioning methods were proposed, the permutations and vocabulary partitioning. Both use Locality-Sensitive Hashing (LSH) and are based on the mathematical concept known as partition of a set. Partitioning any set can generate remainders, and to address this issue, the Remainder at End (RaE), Remainder at Cell (RaC) and Distributed at Cell (DaC) treatment strategies were proposed. Both partitioning methods were approximately 101% faster than the IR state of art. Moreover, RaE, RaC and DaC achieved better results in document indexing, query extraction and retrieval time in comparison to a standard LSH baseline, the Minmax.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Problema e Motivação

No ano de 2009, MCCABE (2009) realizou uma pesquisa com 24.000 alunos de 70 escolas secundárias[1] dos E.U.A e 58% dos alunos admitiram que praticavam plágios em seus trabalhos escolares. Outro dado interessante foi recuperado na pesquisa ETHICS (2012), onde 62% dos alunos em universidades admitem já terem plagiado em trabalhos escritos dentre os 71.300 entrevistados.

O plágio não é apenas uma questão acadêmica, pois pode desencadear crises políticas como em 2011, quando o secretário de defesa alemão renunciou após um escândalo de plágio (FAWZY, 2016; GUARDIAN, 2011). Essa prática também pode acarretar prejuízos financeiros, como em 2006, quando a DreamWorks comprou direitos de filmes baseados em um livro que teve treze passagens plagiadas descobertas por The Harvard Crimson (POSNER, 2007). Portanto, o plágio é um assunto multidisciplinar e existem várias maneiras de analisá-lo e compreendê-lo. Um exemplo é o auto-plágio, que é uma prática relacionada ao plágio de linguagem natural (MARTIN, 1994). Outro exemplo que pode ser dado é o plágio em código-fonte de *softwares* (CLOUGH, 2000).

Diante disso, surge a necessidade de mecanismos para identificação de plágio. A pesquisa de detecção de plágio (PD) visa identificar evidências de que o plágio foi cometido. As estratégias para o PD podem ser externas ou internas (ALZAHRANI *et al.*, 2012). Nas estratégias externas de PD, um documento suspeito é comparado a outros documentos oriundos de diferentes fontes. Por outro lado, estratégias internas consideram que a única informação disponível vem do documento suspeito. A pesquisa de detecção de plágio de linguagem natural (NLPD) visa identificar evidências textuais de que um determinado documento textual contém plágio, a fim de gerar uma lista de documentos que são candidatos a serem plagiados para posterior

[1] equivalente a parte final do ensino fundamental e ao ensino médio brasileiro

avaliação humana. A evidência textual de plágio pode ser produzida diretamente a partir de informações lexicais ou sintáticas (por exemplo, palavras ou frases) extraídas de um texto suspeito (ALZAHRANI *et al.*, 2012).

A identificação de plágio externo é realizada em três etapas: A Busca Heurística, que seleciona possíveis candidatos a plágio; A Comparação Detalhada, que visa comparar trechos dos documentos candidatos com o documento suspeito em busca de evidências de plágio; e a etapa de pós-processamento, onde um ser humano vai verificar se os trechos selecionados configuram plágio (POTTHAST *et al.*, 2014).

A Busca Heurística (HR) tem como objetivo principal a seleção de documentos candidatos a fonte de plágio a partir de um documento suspeito, diminuindo o escopo de busca para as etapas posteriores. Portanto, a etapa de HR deve minimizar o custo de recuperação, enquanto recupera um conjunto pequeno de documentos candidatos dentre todos os documentos fontes possíveis(POSNER, 2007). Em outras palavras, é uma etapa que deve ser realizada de forma rápida, recuperando um conjunto de documentos semelhantes ao documento suspeito.

Neste contexto, modelos de busca aproximada por vizinhos mais próximos (ANN), mais especificamente os métodos Locality-Sensitive Hashing (LSH) podem ser utilizados. LSH são modelos que representam documentos como conjuntos de identificadores conhecidos como assinaturas. Esses identificadores são valores inteiros não-nulos que normalmente são atribuídos a características extraídas do texto, como o conjunto de palavras ou n-gramas. Como o conjunto das assinaturas é numérico, uma forma de se medir similaridade entre documentos é utilizando a interseção entre esses conjuntos, assim, possibilitando o uso da medida de similaridade de Jaccard (ALZAHRANI *et al.*, 2012; BRODER, 1997).

Entre os modelos LSH, as permutações aleatórias podem ser utilizadas, onde um documento é *tokenizado*, os *tokens* gerados são representados por assinaturas, essas sofrem etapas de permutação, e em cada permutação, são selecionadas assinaturas que representem o documento na avaliação de similaridade. Essa seleção é realizada através da aplicação de uma função de seleção.

Dentre as funções de seleção de assinaturas possíveis, Minwise Hashing é a mais conhecida, onde em cada permutação é selecionado um valor de mínimo para representar o documento (BRODER, 1997). A partir disso, JI *et al.* (2013) propôs o Minmaxwise Hashing (Minmax), que além do valor de mínimo, seleciona o valor de máximo do conjunto, visando representar os documentos de forma mais rápida que o Minwise Hashing. Com base nisso, DUARTE *et al.* (2017) propôs as técnicas CSA e CSA_L com o objetivo de diminuir o tempo de permutação do Minmax, selecionando mais valores representativos, pois ao se permutar várias vezes aumenta-se o tempo para indexar os documentos e o tempo de busca (DUARTE *et al.*, 2017).

Com base na mesma ideia de diminuir o tempo de indexação e busca usando o Minmax, a partir da diminuição da quantidade de vezes que a etapa de permutação é realizada, este trabalho propõe duas abordagens de particionamento. Uma das abordagens é focada em particionar o conjunto de assinaturas de cada permutação e a outra, o conjunto de itens do vocabulário dos documentos. Ambas, ao dividir, geram resto que necessitam de tratamento. Foram propostas três técnicas de tratamento deste resto: *Remainder at End* (RaE), *Remainder at Cell* (RaC) e *Distributed at Cell* (DaC).

1.2 Objetivo

O objetivo geral deste trabalho é propor algumas estratégias para execução da tarefa de Busca Heurística do problema de plágio externo. Essa estratégia deve visar o aumento da eficiência da execução da Busca Heurística através da geração de mais valores, por permutação. A partir disso, são propostas estratégias de particionamento, das permutações e do vocabulário, e estratégias de tratamento de restos.

Além de propor, serão avaliadas as propostas quanto a eficiência nas tarefas de indexação e busca da etapa de Busca Heurística. Para tal, devem ser executados experimentos entre todas as combinações possíveis de abordagens de particionamento (permutação e vocabulário) e as estratégias de tratamento de resto (RaE, RaC e DaC). Os experimentos são divididos em avaliar as estratégias propostas com relação a estimar a similaridade de Jaccard e avaliar as estratégias com relação a tarefa de Busca Heurística.

A partir da execução e recuperação dos resultados desses experimentos, as estratégias serão comparadas com o método mais utilizado na área de Busca e Recuperação de Informação (BAEZA-YATES & RIBEIRO-NETO, 2011), o BM25, e com métodos propostos por outros autores para a execução da Busca Heurística, como Minmax, CSA e CSA_L (DUARTE *et al.*, 2017).

1.3 Contribuições

Este trabalho é desenvolvido na área de detecção de Plágio Externo, com as seguintes contribuições:

(i) Aplicação e avaliação do BM25 na tarefa de indexação e busca da etapa de Busca Heurística.

(ii) Criação de estratégia de particionamento das permutações para diminuir o tempo de indexação e busca do Minmax.

(iii) Criação de estratégia de particionamento do vocabulário para diminuir o tempo de indexação e busca do Minmax.

(iv) Criação das estratégias RaE, RaC e DaC para tratamento do resto da divisão dos conjuntos.

(v) Avaliação e comparação do BM25 e estratégias de particionamento combinadas com as estratégias RaE, RaC e DaC com relação a CSA, CSA_L e Minmax.

1.4 Estrutura deste trabalho

O trabalho está estruturado da seguinte forma. O capítulo 2 apresenta as definições de plágio, os tipos de identificação, o plágio externo e a etapa de Busca Heurística. No capítulo 3, é apresentada uma visão geral dos métodos LSH. O capítulo 4 descreve trabalhos relacionados a Busca Heurística no plágio externo. As estratégias de particionamento propostas, sua análise de probabilidade de colisão e seus algoritmos são descritas no capítulo 5, enquanto o capítulo 6 apresenta os experimentos realizados e os resultados obtidos. Finalmente, conclusões e trabalhos futuros são apresentados no capítulo 7.

Capítulo 2

Plágio externo

Plágio pode ser definido como a apresentação de uma obra intelectual, contendo partes de outra obra, que pertença a outro autor sem a sua devida permissão. Ao plagiar, o autor do plágio assume a autoria da obra de outra pessoa indevidamente (ALZAHRANI *et al.*, 2012; MAURER *et al.*, 2006).

Nos dicionários, a palavra plágio possui significados levemente diferentes, como "Ato ou efeito de plagiar; Imitação ou cópia fraudulenta" (PRIBERAM, 2019), "Imitação de trabalho, geralmente intelectual, produzido por outrem" (MICHAELIS, 2019) e "É a prática de tomar o trabalho ou as ideias de outrem e fazê-las passar como próprias" (OXFORD, 2019).

Segundo MAURER *et al.* (2006), a definição de plágio como roubo de propriedade intelectual existe desde que os humanos produziram obras de arte e pesquisa. No entanto, com o fácil acesso à web, os problemas com plágio cresceram e se tornaram críticos para editores, pesquisadores e instituições educacionais. Sendo assim, para o site PLAGIARISM.ORG (2017), mantido por Turnitin, uma empresa importante no mercado de detecção de plágio, o ato de plagiar é uma fraude, envolvendo roubo de trabalho alheio, levando quem plagia a mentir sobre isso após o ato. Além disso, o site lista uma série de práticas que levam ao plágio, como: "transformar o trabalho de outra pessoa como seu", "copiar palavras ou ideias de outra pessoa sem dar crédito", "deixar de colocar uma citação entre aspas", "fornecer informações incorretas sobre a origem de uma citação", "mudar palavras, mas copiando a estrutura da sentença de uma fonte sem dar crédito" e "copiar tantas palavras ou ideias de uma fonte que compõe a maioria do seu trabalho, quer você dê crédito ou não" (PLAGIARISM.ORG, 2017).

Apesar das definições citadas, o plágio nem sempre é intencional ou se apropria de trecho de obra alheia, podendo ocorrer de algumas outras maneiras. Uma delas é quando ocorre de forma **acidental**, onde o autor desconhece sobre o ato de plagiar ou não compreende a forma de citar ou referenciar as fontes utilizadas. Além disso, um plágio pode ocorrer de forma **não intencional**. Esta forma ocorre quando um

autor está imerso em uma quantidade de informações disponíveis e essa acaba por influenciar seus pensamentos, fazendo surgir ideias que outras pessoas já tiveram. A partir disso, o autor se apropria destas ideias e as reproduzem, de forma falada ou materializada, como se fossem suas. Por fim, outra maneira de um plágio ocorrer de forma não intencional é o **auto-plágio**. Este ocorre quando o autor usa algum trecho de um trabalho de sua autoria, publicado anteriormente, sem referenciá-lo (MAURER *et al.*, 2006).

Levando em consideração o plágio textual, MAURER *et al.* (2006) cita diversos métodos empregados no ato do plágio, como:

(i) copiar e colar - conteúdo textual é copiado palavra por palavra;

(ii) plágio de ideias - um conceito ou ideia, que não é de conhecimento comum, é utilizado sem referências;

(iii) paráfrase - conteúdo textual sofre modificação de gramática, substituição de palavras por outras palavras com significado semelhante ou reordenação de frases;

(iv) plágio artístico - autor apresenta um trabalho de outra pessoa usando diferentes mídias, como imagens, voz ou vídeo;

(v) uso inadequado de aspas - ocorre quando há falha na identificação de partes exatas do conteúdo emprestado;

(vi) referências erradas - ocorre quando há adição de referências a fontes incorretas ou inexistentes, e

(vii) plágio de tradução - texto é traduzido de outros idiomas.

O foco deste trabalho é o plágio textual e para isso, na seção 2.1, são apresentadas diferentes estratégias de análise e identificação de plágio. Na seção 2.2, o plágio externo é explorado e um *workflow* de passos é proposto. Por fim, na seção 2.3, a etapa da busca heurística é definida e apresentada.

2.1 Estratégias para identificação de plágio

Para identificar plágio, é necessário um método que identifique a ocorrência de plágio a partir de um documento. Para tal, MAURER *et al.* (2006) apresenta três possíveis métodos para análise e detecção de plágio:

(i) Análise de estilo - conhecida também como estilometria, é fundamentada na hipótese de que cada pessoa tem um estilo de escrita único, podendo ser analisada por meio de suas características.

(ii) Comparação de documentos fonte - comparação de um documento suspeito de ser plagiado com uma coleção conhecida de documentos em busca de suas fontes textuais plagiadas;

(iii) Busca por fragmentos - procura por fragmentos de texto, em vez de todo o documento, supostamente plagiados pelos documentos suspeitos.

Em vista das diferentes formas de se identificar plágio, pode haver ou não a existência de um conjunto de documentos fontes para se buscar por trechos plagiados. Os tipos de plágio que não apresentam esse conjunto de documentos utilizam da estratégia de "análise de estilo", mencionada anteriormente. Em contrapartida, documentos que possuem o conjunto de documentos fontes, podem ser analisados pelas demais estratégias citadas, "comparação de documento fonte"ou de "busca por fragmentos" (BARRÓN-CEDEÑO, 2010).

Quando não existe um conjunto de documentos, são explorados atributos internos do texto, onde a análise se baseia em mudanças e inconsistências do documento. Assim, ao se detectar o plágio, usa-se somente as irregularidades do documento, sem considerar outro documento para análise (STAMATATOS, 2009). Para tanto, um documento é analisado buscando inconsistências como, mudanças no vocabulário, complexidade da escrita e mal fluxo do texto (BARRÓN-CEDEÑO, 2010). Essa forma de identificação é nomeada como plágio interno ou intrínseco.

A tarefa de identificação de plágio interno pode ser definida em alguns passos, como em POTTHAST *et al.* (2011):

(i) Realização da segmentação do texto em parte menores, como frases ou parágrafos;

(ii) Aplicação de um modelo para indexação desses segmentos de texto.

(iii) Execução de algoritmo para detecção de divergências no estilo de escrita entre os segmentos;

(iv) Consolidação e retorno dos resultados da detecção como um conjunto de segmentos com potencial de plágio.

Em contrapartida, quando se possuí documentos de referência, podem se extrair atributos externos, ou seja, atributos que não pertencem ao texto analisado, mas que são utilizados para a detecção do plágio externo. Exemplo desses atributos são os documentos, ou trechos deles, oriundos do conjunto de documentos, geralmente escritos por outro autor. O cenário do plágio externo ainda se divide entre identificar trechos plagiados em documentos inteiros ou em fragmentos de texto (BARRÓN-CEDEÑO, 2010). Essa forma de identificação é nomeada como plágio externo ou extrínseco.

A tarefa de identificar plágio externo pode ser dividida em quatro passos (POTTHAST *et al.*, 2011):

(i) Conjunto D de documentos fontes é pré-processado, onde o texto é segmentado em *tokens*, sendo esses *tokens* indexados em uma estrutura.

(ii) Um processo de seleção de documentos candidatos a plágio é executado utilizando os documentos fonte indexados e o documento suspeito. Esses candidatos são recuperados e indexados em uma nova estrutura.

(iii) O documento suspeito é comparado de forma mais detalhada com os documentos candidatos de plágio. Ao comparar, são extraídos fragmentos que possam ter sido plagiados.

(iv) Os fragmentos recuperados são pós-processados com o objetivo de consolidar o conjunto de fragmentos a serem apresentados como potenciais de plágio, eliminando fragmentos falso positivos.

A seção 2.2 apresenta de forma mais detalhada a identificação de plágio externo, definindo e explanando um *workflow* para identificação do plágio.

2.2 Identificação de plágio externo

A detecção de plagio externo é uma tarefa na qual a região de busca é uma coleção de documentos conhecida, isto é, um conjunto de documentos fontes de plágio está disponível e é alcançável. Assim, um documento suspeito d_{susp} pode ser comparado e verificado, quanto a ocorrência de plágio, utilizando esse conjunto de documentos. Com base nisso, a identificação de plágio externo pode ser vista como uma tarefa de Busca e Recuperação de Informações. Na Recuperação de Informações, um conjunto de documentos é recuperado a partir de uma consulta, podendo ser ou não ordenado pela similaridade entre os documentos. De forma análoga, a detecção de plágio externo pode ser definida, onde um documento d_{susp} é considerado a consulta para um sistema de busca em documentos origem D, sendo retornado um conjunto de documentos semelhantes candidatos a plágio (ALZAHRANI *et al.*, 2012).

Para construção de algum mecanismo automatizado de identificação do plágio externo, baseado em Busca e Recuperação de Informação, podem ser utilizados atributos externos extraídos do textos presentes nos documentos. Estes atributos são utilizados para representar o texto e podem ser classificados como léxicos, sintáticos, semânticos e estruturais. Os léxicos operam no nível de caracteres ou palavras do texto, onde o documento é representado por um conjunto de caracteres ou palavras. Já os sintáticos operam no nível de sentenças ou blocos de texto, onde cada palavra

é classificada com base na sua função sintática na sentença. As funções sintáticas básicas incluem verbos, substantivos, pronomes, adjetivos, advérbios, preposições, conjunções e interjeições. Além disso, o texto pode ser representado por características semânticas, que visam extrair maior visão sobre o significado do texto. Por fim, podem ser utilizadas características estruturais, que levam em consideração as informações contextuais e de organização do texto, capturando maiores informações semânticas (ALZAHRANI *et al.*, 2012).

Para execução da identificação do plágio externo, um *workflow* é proposto por EHSAN & SHAKERY (2016). A figura 2.1 apresenta o *workflow* que é dividido em três etapas: *Heuristic Retrieval*, *Detailed Comparison* e *Knowledge-Based post processing*. A etapa de *Heuristic Retrieval*, ou Busca Heurística, é a etapa inicial do *workflow* e que tem como objetivo principal a diminuição do escopo de busca para as etapas posteriores. Isso ocorre ao submeter um documento suspeito a comparação com todos documentos indexados selecionando um conjunto de documentos candidatos a plágio EHSAN & SHAKERY (2016); STEIN *et al.* (2007). A Busca Heurística é definida e apresentada com mais detalhes na seção 2.3.

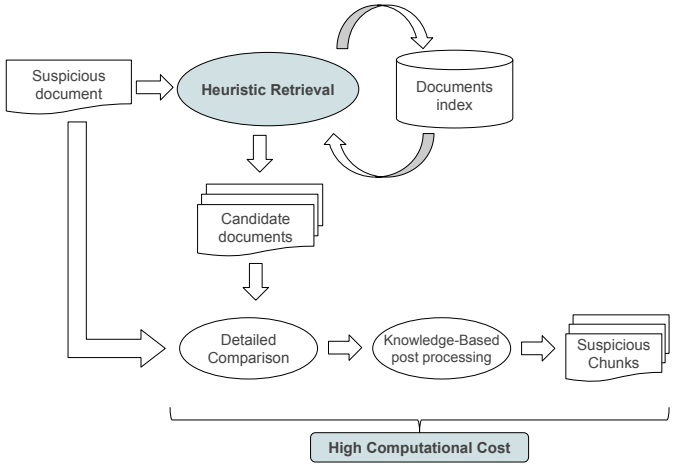


Figura 2.1 – Sequência de etapas para a detecção de plágio externo. Adaptado de EHSAN & SHAKERY (2016)

A segunda etapa da identificação de plágio externo é denominada como *Detailed Comparison*, ou Comparação Detalhada, e tem como objetivo selecionar trechos de texto plagiados, pelo documento suspeito, a partir do conjunto de documentos candidatos. Essa seleção é realizada através de uma comparação exaustiva par-a-par entre documento suspeito e cada documento candidato EHSAN & SHAKERY (2016); STEIN *et al.* (2007).

Por fim, na etapa compreendida como *Knowledge-Based post processing*, ou pós-processamento baseado no conhecimento, todos os trechos de texto, destacados como texto plagiado, são apresentados a um ser humano que decide se o trecho foi plagiado ou não. Essa analise deve levar em consideração se os trechos possuem ou não

citações, caso possuam, são retirados da lista de plágio EHSAN & SHAKERY (2016); STEIN *et al.* (2007).

2.3 Busca Heurística

A etapa de busca heurística (HR) visa reduzir a carga de trabalho das etapas de Comparação Detalhada e Pós-processamento Baseado em Conhecimento, reduzindo a quantidade de dados de entrada para essas etapas MEUSCHKE & GIPP (2014); POTTHAST *et al.* (2014); THOMPSON *et al.* (2015). Portanto, o objetivo da etapa de HR é minimizar o custo de recuperação, enquanto recupera todos os documentos de origem, ou uma pequena coleção de candidatos para documentos de origem, de um texto plagiado (POTTHAST *et al.*, 2014).

Um exemplo prático de redução ocasionada pelo HR pode ser descrito por uma coleção de documentos $D = \{d_1, d_2, d_3, d_4, \boldsymbol{d_5}, d_6, d_7, d_8, d_9, \boldsymbol{d_{10}}\}$, os documentos d_5 e d_{10} são documentos fontes de $d_q \in D_{susp}$ e um método de HR retorna um conjunto de documentos candidatos $d_{src} \in D$, menor que $|D| = 10$, onde d_5 e d_{10} estão inclusos. Como exemplo de retorno podem ser citados inúmeros conjuntos: $d_{src} = \{d_1, d_2, \boldsymbol{d_5}, \boldsymbol{d_{10}}\}$, $d_{src} = \{d_3, d_4, \boldsymbol{d_5}, d_7, \boldsymbol{d_{10}}\}$ ou $d_{src} = \{d_1, \boldsymbol{d_5}, d_7, \boldsymbol{d_{10}}\}$ (DUARTE, 2017).

A etapa de HR é composta de duas tarefas: (i) indexação da região de busca por plágio, ou seja, da coleção de documentos fontes de plágio, e (ii) busca e recuperação de documentos candidatos, conhecido como busca por origem de plágio.

A tarefa de indexação (i) é composta por um Modelo de Recuperação de Informações e duas etapas, conforme mostrado na figura 2.2. Na etapa de pré-processamento, a normalização do texto é aplicada para garantir que o índice do documento seja consistente. Além disso, na etapa de pré-processamento, as palavras do texto são convertidas em minúsculas, e espaços em branco extras, datas e pontuação podem ser removidos. Então, na etapa de extração de atributos, os documentos são tokenizados, por exemplo, dividido em palavras, frases, n-gramas e etc, e os tokens, que podem ser usados como evidência de plágio, são selecionados como atributos do Modelo de Recuperação de Informações (EHSAN & SHAKERY, 2016).

A tarefa de Busca de Origem de plágio (ii) visa selecionar a coleção mais provável de documentos que foram plagiados pelo documento suspeito (POTTHAST *et al.*, 2014). A Figura 2.3 mostra como três etapas permitem que o Modelo de Recuperação de Informações encontre a evidência textual do plágio, do documento suspeito, em documentos do índice gerado pela tarefa de indexação mencionada anteriormente. Nas etapas de pré-processamento e extração de consulta, semelhantes às duas etapas da figura 2.2, cada documento suspeito é pré-processado, dividido em uma ou mais

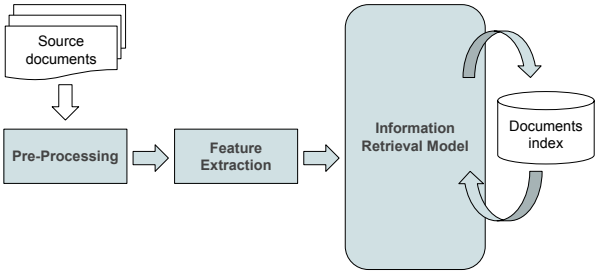


Figura 2.2 – Sequência de etapas para a tarefa de indexação.

consultas, onde atributos são extraídos para cada consulta (DUARTE *et al.*, 2017). Em seguida, a etapa de expansão da consulta aprimora as consultas substituindo os recursos léxico, sintático ou semântico e as consultas são enviadas ao Modelo de Recuperação de Informações. Por fim, a etapa de pós-processamento remove os documentos recuperados, com conteúdo semelhante, que não são plagiados pelo documento suspeito (DUARTE *et al.*, 2017; EHSAN & SHAKERY, 2016).

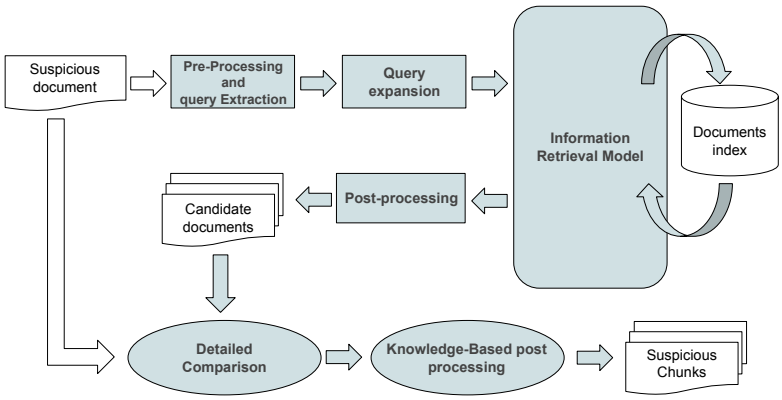


Figura 2.3 – Sequência de etapas para tarefa de busca pela fonte de plágio.

Capítulo 3

Locality-Sensitive Hashing

Com o massivo crescimento na geração de dados oriundos de diversas fontes e formatos, a busca por conteúdo relevante, em conjuntos grandes de documentos, tornou-se uma tarefa bastante desafiadora (WANG *et al.*, 2016). A partir disso, uma área que visa estudar e suprir essa necessidade é a Nearest Neighbor Search (NNS).

O Nearest Neighbor Search (NNS) ou busca pelo vizinho mais próximo é definido como um problema de otimização onde o objetivo é encontrar o ponto mais próximo, ou mais similar, de um dado ponto dentro de um conjunto de dados. É formalizado como: Dado um conjunto P de n pontos em algum espaço (X, F), definido em um conjunto X com função de distância ou similaridade F, é construída uma estrutura de dados qualquer, que permita que dado um ponto q, seja encontrado em P um ponto próximo, ou similar, n_1 a q (ANDONI *et al.*, 2018). Como exemplificado na figura 3.1, dado o ponto q, um ponto n_1 é o vizinho mais próximo de q, por possuir a menor distância para q e, sendo o ponto em P mais similar.

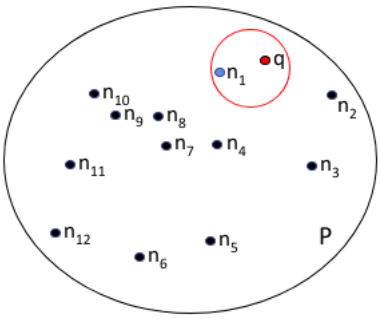


Figura 3.1 – Busca de vizinho mais próximo de q em P usando NNS

Abordagens de NNS são aplicadas em diversas áreas da computação, dentre essas podem ser destacadas a utilização para recuperação de informação e para detecção de plágio externo (SHAKHNAROVICH *et al.*, 2006). De acordo com MOHAMED & MARCHAND-MAILLET (2015), o NNS é categorizado em três tipos: *Exhaustive*

(XNN) ou exaustivo, *Exact* (ENN) ou exato, e *Approximated* (ANN) ou aproximado (MOHAMED & MARCHAND-MAILLET, 2015).

O primeiro tipo é a busca exaustiva (XNN), onde para dado uma consulta q qualquer em um conjunto de documentos D, são calculadas e armazenadas a similaridade par-a-par entre a consulta e cada documento do conjunto de documentos. Assim, XNN tem como principal desvantagem o custo em tempo para computar a similaridade par-a-par para cada consulta com toda base de documento, levando a um aumento exponencial com o aumento do conjunto de dados. Além disso, o armazenamento desses dados se torna bem custoso e o uso dessa técnica se torna proibitiva para grandes conjuntos de documentos (LI & KÖNIG, 2011; MOHAMED & MARCHAND-MAILLET, 2015).

O tipo busca exata (ENN) também faz uma comparação exaustiva entre pares, se diferenciando do XNN por ter um número menor de comparações entre documentos, pois é baseado em partições de espaço ou em técnicas de decomposição (MOHAMED & MARCHAND-MAILLET, 2015; WANG *et al.*, 2014). Entretanto, é apenas uma solução eficiente para casos de baixa dimensão, pois para dados dimensionais altos, o desempenho da busca ENN pode ser menor que o da busca XNN (MOHAMED & MARCHAND-MAILLET, 2015).

Um exemplo de aplicação do ENN pode ser visto na figura 3.2, onde uma técnica de decomposição é aplicada sobre o conjunto P, levando a diminuição da quantidade de comparações par-a-par entre q e P. Para que essa decomposição do espaço seja realizada, a indexação é realizada utilizando alguma técnica de índice espacial. No exemplo é aplicado o *quadtree*, que utiliza uma estrutura de dados para realizar a indexação espacial. O *quadtree*, considera que cada nó corresponda a uma região quadrada do espaço e esta região podendo ser subdividida, novamente em quatro partes, gerando mais um nível na árvore, e assim sucessivamente. As subdivisões terminam quando cada quadrado tiver um ou nenhum documento dentro (CASANOVA *et al.*, 2005). Toda essa subdivisão leva em conta as dimensões do documento espacial, no caso do texto, cada *token* do documento.

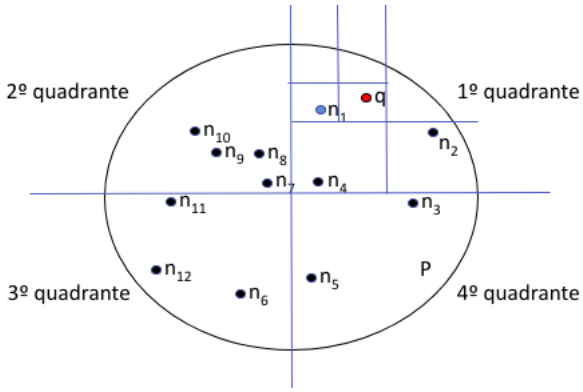


Figura 3.2 – Busca de vizinho mais próximo usando ENN

O que explica o baixo desempenho de XNN e ENN em altas dimensões é o fenômeno da *"maldição da dimensionalidade"* (INDYK & MOTWANI, 1998), expressão criada por BELLMAN *et al.* (1957), referindo-se aos desafios que surgem com a necessidade de se analisar dados em espaço de alta dimensão. Nesse problema, quando a dimensionalidade aumenta, o volume do espaço aumenta rapidamente, tornando os dados disponíveis esparsos, dificultando a análise e organização (MARIMONT & SHAPIRO, 1979).

Um exemplo prático da ineficiência desses métodos, quando aplicados a altas dimensões, é imaginar uma situação de busca textual na Wikipédia, que possui mais de 40 milhões de artigos indexados (WIKIMEDIA, 2019). Neste caso, ao realizar uma busca por documentos próximos utilizando XNN, cada documento é comparado aos 40 milhões de documentos previamente indexados. Considerando que a Wikipédia recebe 7489 novos documentos diários, seriam gerados 299 bilhões de comparações por dia.

Em WIKIPEDIA (2019) pode ser visto que a Wikipédia tem 27 bilhões de palavras nos 40 milhões de documentos indexados, assim, tendo uma média de 562 palavras por documento. Se o ENN for aplicado na indexação dos 7489 novos documentos diários, utilizando o *quadtree*, seriam necessárias diversas comparações e aplicações da decomposição até que a estrutura de dados seja formada. Nesse caso, deve-se levar em consideração que cada etapa da decomposição é realizada utilizando as 562 palavras de cada um dos 40 milhões de documentos.

A fim de evitar *"a maldição da dimensionalidade"* os métodos de busca aproximada (ANN) podem ser utilizados (INDYK & MOTWANI, 1998). Estes visam estimar os vizinhos mais próximos de um ponto de dados sem realizar uma comparação exaustiva sobre uma coleção de documentos. Assim, em cenários de grande dimensão e grande escala, a ANN pode realizar a tarefa de NNS eficientemente, reduzindo tempo de busca e quantidade de memória utilizada (ANDONI *et al.*, 2018; MOHAMED & MARCHAND-MAILLET, 2015; WANG *et al.*, 2014).

O ANN é baseado na premissa de que objetos similares estão mais próximos do que objetos não similares, assim, o algoritmo aproximado pode retornar o mesmo resultado de um algoritmo exato (GIONIS *et al.*, 1999). Contudo, existe uma pequena probabilidade de não se conseguir encontrar o vizinho mais próximo ao objeto consultado pois é uma técnica de aproximação que depende de um fator de aproximação (SLANEY & CASEY, 2008). Um exemplo de método ANN é o *Locality-Sensitive Hashing* (LSH), que mapeia cada ponto do espaço em um conjunto de números conhecidos como assinaturas (WANG *et al.*, 2014). Além disso, WANG *et al.* (2014) destaca que LSH é um dos métodos mais utilizados de ANN, e portanto, será discutido nas próximas seções.

3.1 Definição

O LSH foi desenvolvido por INDYK & MOTWANI (1998) e aprimorado por GIONIS *et al.* (1999). A ideia do LSH é gerar assinaturas para diferentes objetos usando diferentes funções de *hash*, a fim de garantir que, em cada função, a probabilidade de se recuperar objetos similares em uma busca seja maior que recuperar objetos dissimilares (GIONIS *et al.*, 1999). Para o LSH, se dois objetos são próximos, a sua projeção vetorial em um subespaço de menor dimensionalidade leva a dois pontos vizinhos, assim, podendo se calcular assinaturas dos objetos a fim de compará-los em outro subespaço (GIONIS *et al.*, 1999; SLANEY & CASEY, 2008).

Um exemplo desse conceito pode ser visto na figura 3.3. O conjunto de documentos $\{d_1, d_2, d_3, d_4, d_5, d_6, d_7\}$ é representado por pontos em um espaço e nestes pontos é aplicada uma função *hash* qualquer $h(\cdot)$. No LSH, quando documentos são próximos, como no conjunto de documentos $\{d_1, d_2, d_3\}$, continuam próximos em outro subespaço, em um mesmo *bucket*, após a aplicação da função $h(\cdot)$.

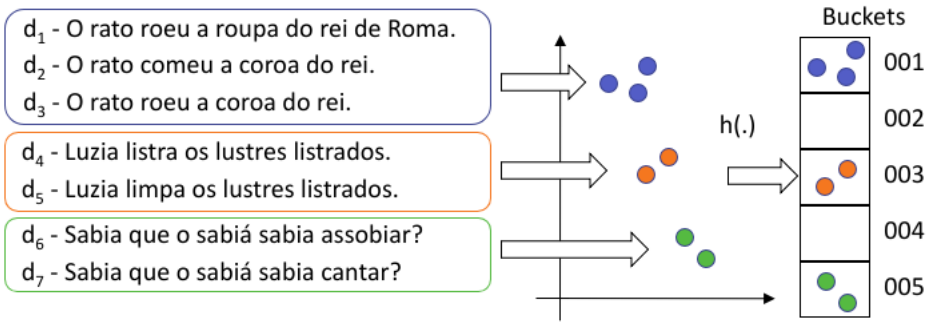


Figura 3.3 – Aplicação de *hash* garantindo proximidade entre documentos

Em outras palavras, a família de métodos LSH, com uma função de *hash* $h(\cdot)$, apresenta uma probabilidade de colisão proporcional à sua similaridade, quando dois pontos x e y tem pelo menos um *hash* em comum (WANG *et al.*, 2014, 2016). Além disso, como visto na equação 3.1, a probabilidade de colisão da *hash* é avaliada pela similaridade par-a-par $sim(\cdot, \cdot)$, tal como as similaridade do cosseno, *Hamming* ou *Jaccard* (WANG *et al.*, 2016).

$$Pr[h(x) = h(y)] = sim(x, y) \tag{3.1}$$

Por ser uma busca aproximada (ANN), pode produzir resultados um pouco menos precisos do que os obtidos por métodos de pesquisa exaustivos. No entanto, permite pesquisar grandes coleções de itens de alta dimensão, mapeando cada item em um conjunto de assinaturas, melhorando o tempo de indexação e recuperação (DUARTE *et al.*, 2017; WANG *et al.*, 2016).

Um método da família LSH pode estar em duas categorias: projeções aleatórias e permutações aleatórias. O método de projeção aleatória visa mapear pontos próximos de um espaço original para o mesmo *bucket* em outro espaço, preservando a localidade dos elementos. Segundo WANG *et al.* (2016), a partir de dois pontos x_i e x_j quaisquer, quando aplicada a projeção $h_k(x) = sgn(w_k^T x + b_k)$, dois vetores são gerados em um subespaço. Na figura 3.4 pode ser visto um exemplo dos vetores x_i e x_j gerados a partir de uma projeção qualquer $w^T x + b$. Para o cálculo de similaridade entre esses dois elementos, diferentes projeções são criadas. Para cada projeção, w recebe um valor aleatório de uma distribuição Gaussiana padrão para distância de cosseno e b recebe um valor aleatório qualquer.

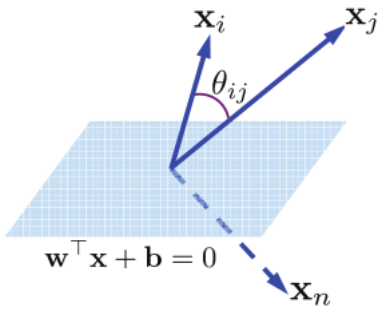


Figura 3.4 – Projeção aleatória. Adaptado de WANG *et al.* (2016)

A probabilidade dos elementos x_i e x_j estarem próximos, após a aplicação da projeção, é determinada pelo ângulo $\theta_{i,j}$ entre os vetores, como na equação 3.2. No entanto, as funções da projeção aleatória são independentes dos dados e ignoram propriedades importantes destes dados, podendo gerar funções menos efetivas em comparação a métodos de aprendizado de máquina que aprendem com os dados da coleção (WANG *et al.*, 2016).

$$Pr[h_k(x_i) = h_k(x_j)] = 1 - \frac{\theta_{i,j}}{\pi} = 1 - \frac{1}{\pi} cos^{-1} \frac{x_i^T x_j}{||x_i||\,||x_j||} \tag{3.2}$$

Outra categoria é a de permutações aleatórias. A ideia básica da permutação aleatória é ordenar aleatoriamente um conjunto de objetos. Por exemplo, a partir de um conjunto de números $D = \{1, 2, 3, 4, 5, 6\}$, cada ordenação possível deste conjunto produz uma nova lista dos números, sem repetições. Um exemplo de aplicação da permutação neste conjunto D é $\{1, 2, 6, 3, 5, 4\}$ (PEMMARAJU & SKIENA, 2003).

As permutações aleatórias são utilizadas em tarefas de aproximação da similaridade de Jaccard entre conjuntos, sendo Jaccard comumente usada para medir a similaridade entre documentos textuais (WANG *et al.*, 2016). Para tal, nas permutações aleatórias, *tokens* de um conjunto de documentos textuais são representados por números inteiros não nulos, denominados assinaturas. O conjunto de assinatu-

ras v de cada documento é submetido a funções de permutação h_i, transformando cada assinatura de um espaço R^D em um espaço Z^M. A etapa de permutação é repetida diversas vezes, utilizando funções de permutações independentes. A partir de cada conjunto resultante de cada permutação, é selecionado o valor mínimo para representar v. O conjunto de valores selecionados em $min(h_i(v))$ é utilizado para o cálculo da similaridade de Jaccard (DUARTE *et al.*, 2017; WANG *et al.*, 2016).

Nas seções a seguir, é apresentado um *workflow* baseado em busca aproximada utilizando permutação aleatória, que é a base para a proposta apresentada no capítulo 5.

3.2 Busca aproximada por vizinho mais próximo usando LSH por permutação

Existem diferentes métodos de *hash* aleatórios para realizar pesquisa de ANN. WANG *et al.* (2014) propuseram um *workflow* genérico para executar qualquer método de *hash* aleatório com base em três etapas básicas:

(i) Projetar a função *hash* - escolha uma função *hash* $h(.)$ para cada permutação π_n para gerar os códigos *hash*;

(ii) Indexar o conjunto de dados - gere uma tabela de *hash* organizando os códigos de *hash* como uma lista invertida;

(iii) Consulta online - após a etapa ii, pesquise cada código *hash*, de uma determinada consulta, na tabela de *hash* procurando os vizinhos mais próximos da consulta.

Posteriormente, um *workflow* mais detalhado foi proposto em DUARTE *et al.* (2017). A sequência de etapas no *workflow* desta tarefa é: (i) *Tokenização*, (ii) Geração de assinaturas, (iii) Permutação, (iv) Aplicação de função de seleção e (v) Avaliação de similaridade. Diferentes estratégias podem ser utilizadas e comparadas para cada etapa do *workflow*, por exemplo, diversas funções de seleção, como: Minwise, Minmaxwise, Minmax Circular Sector Arcs Lower Bound (CSA_L) and Minmax Circular Sector Arcs Full Bound (CSA). Outro exemplo é a possibilidade de comparações entre diversas formas de *tokenizar* e seus pré-processamentos. Baseadas nesse *workflow*, as subseções seguintes apresentam as etapas (i), (ii), (iii), (iv) e (v).

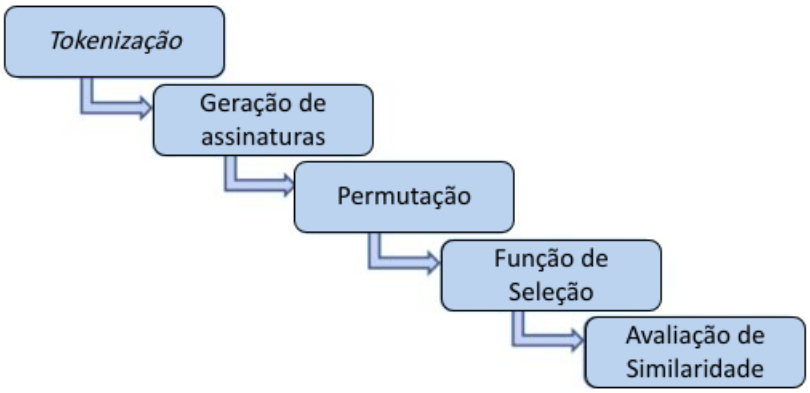
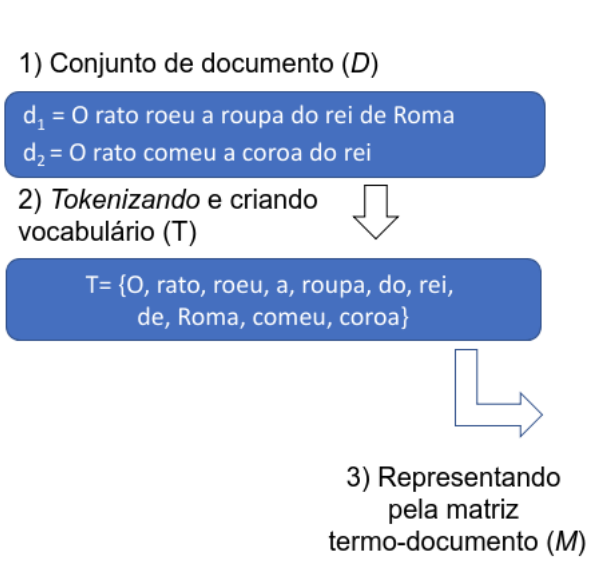


Figura 3.5 – *Workflow* adaptado de DUARTE *et al.* (2017)

3.2.1 *Tokenização*

Esta etapa permite transformar um documento de texto em um conjunto de *tokens*. Especificamente, para cada documento d de uma coleção de documentos D, o conjunto correspondente de termos é extraído e representado como uma matriz binária termo-documento $M \in \{0,1\}^{j \times 1}$, onde j é o número de termos e 1 representa um documento.

Na figura 3.6 é apresentado um exemplo de matriz termo-documento criada a partir dos documentos d_1 e d_2, onde d_1 = "O rato roeu a roupa do rei de Roma"e d_2 = "O rato comeu a coroa do rei". Os documentos citados são processados e as palavras são extraídas como *tokens* formando um vocabulário T para coleção D. A partir disso, cada item do vocabulário é representado conforme sua ocorrência ou não no documento, onde a ausência é representada por 0 (zero) e a ocorrência por 1 (um).



	d_1	d_2
o	1	1
rato	1	1
roeu	1	0
a	1	1
roupa	1	0
do	1	1
rei	1	1
de	1	0
roma	1	0
comeu	0	1
coroa	0	1

Figura 3.6 – Representação do documento na matriz termo-documento

3.2.2 Geração de assinaturas

Nesta etapa, uma assinatura g_k é gerada para cada *token* t_k do vocabulário T da coleção de *tokens* T. Uma assinatura é sempre um inteiro não negativo que é gerada após a definição e aplicação de uma função de transformação $\psi(t_k)$ para cada termo t_k. Então, cada documento $d_j \in D$ é representado como um conjunto de assinaturas L_j gerado pela aplicação de ψ. Observe que a mesma assinatura g_k pode aparecer no conjunto de assinaturas de dois ou mais documentos, caso aconteça de os documentos possuírem o termo t_k em comum.

No exemplo da figura 3.7 é apresentada a aplicação de uma função $\psi()$ sobre os termos da matriz termo-documento M gerando uma matriz M' e o conjunto de assinaturas $L_j = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$ que representam cada termo do vocabulário T.

Matriz termo-documento *M*

	d_1	d_2
Ψ(o)	1	1
Ψ(rato)	1	1
Ψ(roeu)	1	0
Ψ(a)	1	1
Ψ(roupa)	1	0
Ψ(do)	1	1
Ψ(rei)	1	1
Ψ(de)	1	0
Ψ(roma)	1	0
Ψ(comeu)	0	1
Ψ(coroa)	0	1

Aplicação de Ψ em cada termo para geração de assinaturas

Matriz *M'* resultante da aplicação de Ψ

	d_1	d_2
1	1	1
2	1	1
3	1	0
4	1	1
5	1	0
6	1	1
7	1	1
8	1	0
9	1	0
10	0	1
11	0	1

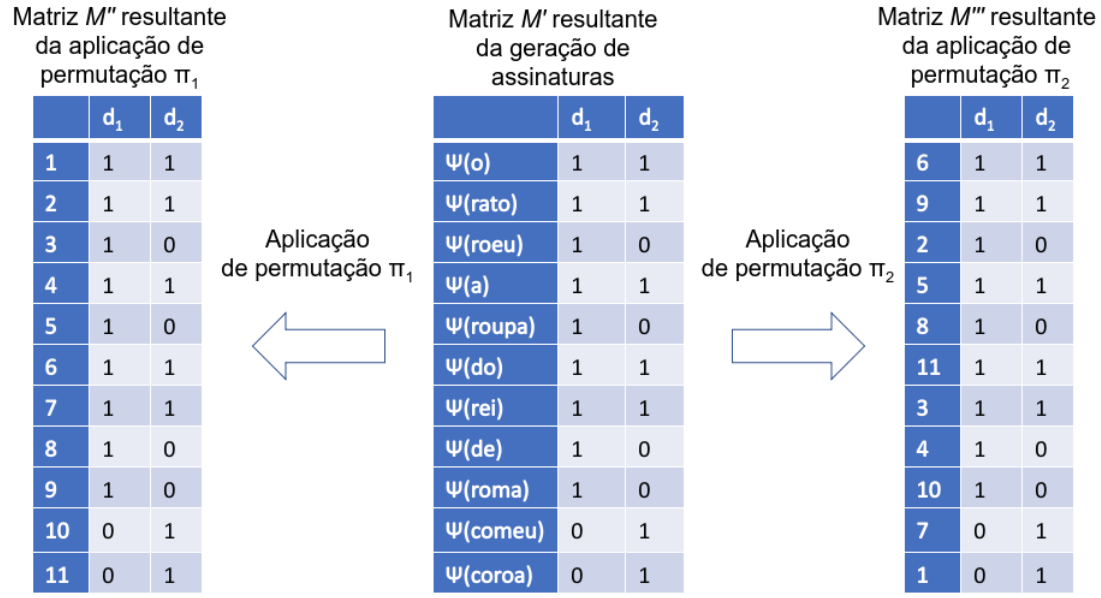
Figura 3.7 – Geração de assinaturas a partir de aplicação de $\psi()$ em t_k

3.2.3 Permutação

Esta etapa consiste na aplicação de uma função de permutação $h(.) = \pi_n(.)$ para ordenar o conjunto de assinaturas L_j para cada $d_j \in D$. Sendo L_j um conjunto numérico, quando a função $h(.)$ é aplicada, cada assinatura g_k assume um novo valor, de acordo com a ordenação de $h(.)$. Além disso, para um vocabulário T, existem $|T|!$ possíveis permutações e $N \leq T!$ permutações resultam em $L'_{j,N} = \{\pi_1(L_j), \pi_2(L_j) \ldots, \pi_N(L_j)\}$ assinaturas a partir de N funções de ordenação.

No exemplo da figura 3.8 é apresentada a aplicação de funções de permutação para o conjunto de assinaturas L_j representadas por M' no exemplo 3.7. Tais funções geram as assinaturas $L_1 = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$ e $L_2 =$

$\{6, 9, 2, 5, 8, 11, 3, 4, 10, 7, 1\}$ através das funções $\pi_1(.)$ e $\pi_2(.)$, respectivamente. Assim, os documentos d_1 e d_2 podem ser representados por M'' para aplicação de $\pi_1(.)$ e M''' para aplicação de $\pi_2(.)$.



	d_1	d_2
1	1	1
2	1	1
3	1	0
4	1	1
5	1	0
6	1	1
7	1	1
8	1	0
9	1	0
10	0	1
11	0	1

	d_1	d_2
Ψ(o)	1	1
Ψ(rato)	1	1
Ψ(roeu)	1	0
Ψ(a)	1	1
Ψ(roupa)	1	0
Ψ(do)	1	1
Ψ(rei)	1	1
Ψ(de)	1	0
Ψ(roma)	1	0
Ψ(comeu)	0	1
Ψ(coroa)	0	1

	d_1	d_2
6	1	1
9	1	1
2	1	0
5	1	1
8	1	0
11	1	1
3	1	1
4	1	0
10	1	0
7	0	1
1	0	1

Figura 3.8 – Aplicação de permutações em L_j

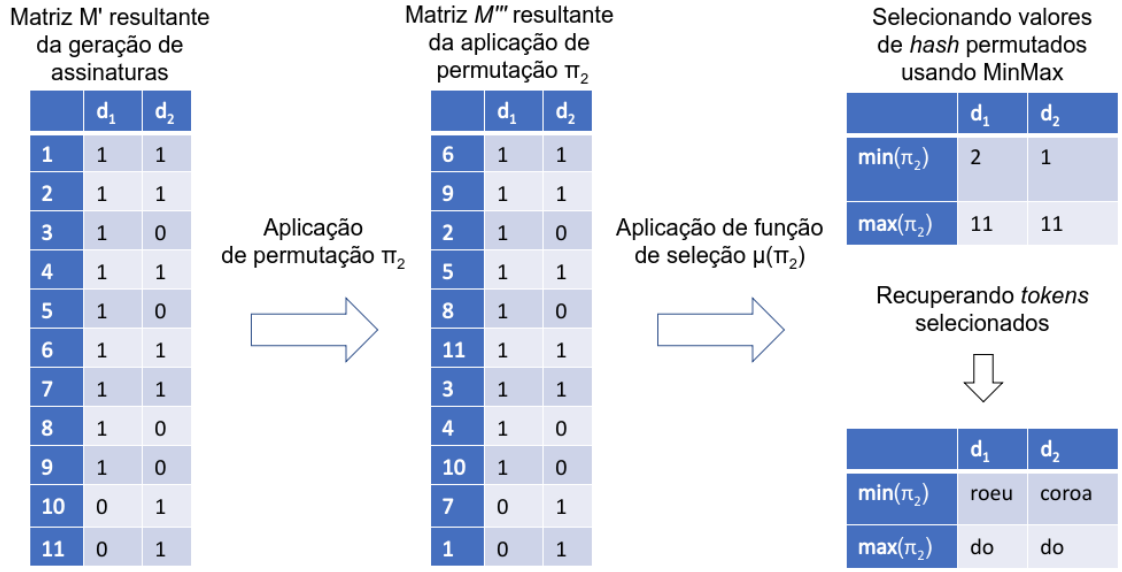
3.2.4 Aplicação de função de seleção

Um documento d_j com um número $|L_j|$ de *tokens* tem $|L_j| \times N$ assinaturas em $L'_{j,N}$. Cada conjunto de assinaturas cria um custo de armazenamento e comparação quando realizada uma busca qualquer em D. A fim de amenizar isso, uma função de seleção é utilizada permitindo a sumarização de $L'_{j,N}$ assinaturas, selecionando um subconjunto $S_{i,j}$ para cada permutação π_i. Além disso, essa sumarização deve garantir a possibilidade de se calcular a similaridade entre documentos. Isso acaba sendo possível pois esta sumarização seleciona os elementos mais representativos do documento para o subconjunto, levando a seleção de elementos na interseção entre eles, assim, aumentando a similaridade caso possuam muitos elementos em comum (BRODER, 1997).

A sumarização é realizada a partir da execução de uma função escolhida μ, sendo $\mu(d_i, \pi_n, O_k)$, onde d_i é o documento a ser sumarizado, π_n é a função de permutação e O_k é o operador, que pode ser, por exemplo, uma seleção de um valor de mínimo, máximo ou a junção desses. Como, a cada aplicação de permutação $\pi_n(.)$, o conjunto de assinaturas L é transformado em valores diferentes, é possível selecionar diferentes valores ao aplicar uma função de seleção μ após uma nova permutação $\pi_n(.)$.

No exemplo apresentado na figura 3.9 a função de seleção MinMaxwise Hashing é aplicada no conjunto resultante da permutação $\pi_2(L_j)$. Nessa função, os valores de mínimo e máximo do conjunto são recuperados. No exemplo, os valores $\{2, 11\}$

foi recuperado para d_1 e $\{1, 11\}$ para d_2. Esses valores, como mostrado no exemplo 3.8, referenciam os *tokens* $\{$"*roeu*", "*do*"$\}$ para d_1 e $\{$"*coroa*", "*do*"$\}$ para d_2.



	d_1	d_2
1	1	1
2	1	1
3	1	0
4	1	1
5	1	0
6	1	1
7	1	1
8	1	0
9	1	0
10	0	1
11	0	1

	d_1	d_2
6	1	1
9	1	1
2	1	0
5	1	1
8	1	0
11	1	1
3	1	1
4	1	0
10	1	0
7	0	1
1	0	1

	d_1	d_2
$min(\pi_2)$	2	1
$max(\pi_2)$	11	11

	d_1	d_2
$min(\pi_2)$	roeu	coroa
$max(\pi_2)$	do	do

Figura 3.9 – Aplicação de função de seleção MinMaxwise Hashing em $\pi_n(L_j)$

3.2.5 Avaliação de similaridade

A similaridade entre dois documentos d_i e d_j é calculada a partir do conjunto de assinaturas dos documento. Na equação 3.3 é apresentada a formalização do cálculo de similaridade $sim_m(d_i, d_j)$, entre d_i e d_j, onde m é a métrica utilizada.

$$sim_m(d_i, d_j) : Sig \times Sig \mapsto R \tag{3.3}$$

O método LSH usa a similaridade de Jaccard para medir a similaridade entre os conjuntos de assinatura de d_i e d_j, sendo o m da equação citada. Jaccard é utilizado por preservar a similaridade par-a-par (JI *et al.*, 2013). Jaccard é formalizado na equação 3.4, onde a similaridade é a diferença entre o número de elementos contidos na interseção dos conjuntos A e B, e o número de elementos contidos na união de A e B.

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{3.4}$$

Se adaptado para o cálculo de similaridade no LSH, Jaccard poderia ser representado como na equação 3.5, onde A e B são substituídos por $\mu(\pi_n(S_1))$ e $\mu(\pi_n(S_2))$, respectivamente.

$$J(d_1, d_2) = \frac{|\mu(\pi_n(S_1)) \cap \mu(\pi_n(S_2))|}{|\mu(\pi_n(S_1)) \cup \mu(\pi_n(S_2))|} \tag{3.5}$$

No exemplo 3.10, as permutações $\{\pi_1, pi_2, pi_3, pi_4\}$ são aplicadas de forma independentes nos documentos d_1 e d_2 e, a partir de cada conjunto gerado, são selecionados valores de mínimo e máximo.

	π_1		π_2		π_3		π_4	
d_1	2	11	3	10	4	9	5	8
d_2	1	11	3	10	4	9	6	8
	min	max	min	max	min	max	min	max

Figura 3.10 – Exemplo de aplicação de múltiplas permutações e seleção de mínimos e máximos

A partir dos valores selecionados, é realizado um exercício de cálculo de similaridade entre os documentos, como no exemplo da equação 3.6, onde é aplicada a função de Jaccard sobre os conjuntos oriundos de d_1 e d_2.

$$J(d_1, d_2) = \frac{|\{2,3,4,5,8,9,10,11\} \cap \{1,3,4,6,8,9,10,11\}|}{|\{2,3,4,5,8,9,10,11\} \cup \{1,3,4,6,8,9,10,11\}|} = \frac{6}{8} = 0,75 \tag{3.6}$$

3.3 Tarefas de indexação e busca

Na tarefa de indexação é criado um índice invertido orientado a *hash*, armazenando os documentos da coleção D. Com base nisso, o algoritmo 1 apresenta o processo de indexação, onde o conjunto D é dado como entrada junto do número de permutações a serem utilizadas. Para cada documento da coleção D um conjunto de funções de permutação π_P é executado e para cada permutação, uma função de seleção μ é utilizada para extração de *hashes* significativas que representem o documento. Com essas *hashes* extraídas, é criado um índice invertido, onde o índice são as *hashes* selecionadas e estas apontam para cada documento que representam.

Esse índice é armazenado e fica disponível para a tarefa de consulta.

Algoritmo 1: Indexação da coleção D usando LSH

Entrada: $D = \{d_1, d_2, d_3, ..., d_{|D|}\}$, P = número de permutações
Saída: *indice*

```
1 Inicialização:  indice = índice invertido
2 início
3     para i ← 1 até |D| faça
4         para n ← 1 até P faça
5             S_{i,n,k} ← μ(d_i, π_n, O_k)
6             para sig ∈ S_{i,n,k} faça
7                 indice.sig ← indice.sig ∪ {d_i}
```

Na tarefa de consulta, um conjunto de documentos suspeitos d_{susp} é utilizado com objetivo de encontrar os documentos que mais se aproximam de cada documento q_i e que poderiam ser as possíveis fontes de plágio. O algoritmo 2 apresenta o processo de busca, onde cada documento suspeito q_i passa pelo mesmo processo de permutações e seleção de assinatura que os documentos fonte D. Após isso, as assinaturas selecionadas são utilizadas para a busca pelos documentos fontes utilizando o índice invertido indexado na etapa anterior.

Através do conjunto D^*_{susp}, que representa os documentos q_i, e de seus documentos relacionados d_{src}, é realizada a avaliação de similaridade, como apresentado na seção 3.2.5. Esse processo é realizado utilizando as assinaturas dos documentos envolvidos, e após isso, tais documentos são *rankeados* com relação a similaridade

com o documento suspeito q_i.

Algoritmo 2: Consulta de um conjunto suspeito d_{susp} na coleção D usando LSH

Entrada: $d_{susp} = \{q_1, q_2, q_3, ..., q_{|d_{susp}|}\}$,
$indice$ = índice invertido,
P = número de permutações
Saída: $indice$

```
1 Inicialização:  D*_susp = {}
2 início
3     para i ← 1 até |d_susp| faça
4         para n ← 1 até P faça
5             S_{i,n,k} ← μ(q_i, π_n, O_k)
6             para sig ∈ S_{i,n,k} faça
7                 para d_src ∈ indice.sig faça
8                     D*_susp ← D*_susp ∪ {d_src}
```

Capítulo 4

Trabalhos Relacionados

Devido à maldição da dimensionalidade e às grandes coleções de documentos encontradas em cenários reais, a etapa de busca heurística, para resolução do problema de plágio, pode ser definida como métodos de busca aproximada orientados por *token* (DUARTE *et al.*, 2017). De fato, muitas pesquisas sobre pré-processamento, extração e expansão de consultas foram feitas em competições da PAN[1] BRASCHLER *et al.* (2010); CAPPELLATO *et al.* (2015); FORNER *et al.* (2013); PAN & MANOCHA (2012); POTTHAST *et al.* (2011, 2014). No entanto, toda a pesquisa do PAN manipula a tarefa de recuperação de documento fonte de plágio, com base em um mecanismo de pesquisa BM25, sem se preocupar com a tarefa de indexação.

Esse capítulo visa apresentar alguns trabalhos e métodos relacionados com a etapa de busca heurística na resolução do problema de plágio externo, com foco na indexação e recuperação de documentos candidatos de plágio utilizando estratégias de busca aproximada. Na seção 4.1, o método BM25 é definido e apresentado, junto de trabalhos realizados em competições PAN. Na seção 4.2, funções de seleção para LSH são apresentadas, sendo elas: Minwise hashing, Minmax hashing, Minmax Circular Sector Arcs Lower Bound ($\mathrm{CSA_L}$) e Minmax Circular Sector Arcs Full Bound (CSA).

4.1 BM25

O BM25 (Best Match 25) é um modelo probabilístico que classifica a relevância dos documentos para uma determinada consulta, isto é, o ato de buscar a informação é baseado na probabilidade de um documento ser relevante para uma determinada consulta. Foi criado a partir de diversos experimentos em variações de modelos probabilísticos, sendo resultante da combinação de BM11 e BM15 (BAEZA-YATES & RIBEIRO-NETO, 2011). Segundo BAEZA-YATES & RIBEIRO-NETO (2011),

[1]Plagiarism Analysis, Authorship Identification, and Near-Duplicate Detection

na área de recuperação de informação, o BM25 é um dos mais populares e efetivos algoritmos de ranqueamento, apresentando resultados melhores para ranqueamento, se comparado com o modelo vetorial clássico, servindo como base de comparação para novos métodos de ranqueamento. Na equação 4.1 é apresentada a fórmula do BM25.

$$BM25(d) = \sum_{i=1}^{n} IDF(q_i) * \frac{f(q_i) * (k_1 + 1)}{f(q_i) + k_1 * (1 - b + b * \frac{|D|}{d_{avg}})} \tag{4.1}$$

Onde:

- $f(q_i)$ é o número de vezes que o termo q_i ocorre em um documento d;
- $|d|$ é o número de palavras em um documento d;
- d_{avg} é o número médio de palavras por documento;
- b e k_1 são parâmetros livres, geralmente escolhidos como: $k_1 \in [1.2, 2.0]$ e $b = 0,75$;
- $IDF(q_i) = \log \frac{|D|}{n(q_i)}$, onde $|D|$ é o número total de documentos na coleção e $n(q_i)$ é o número de documentos que contêm q_i.

4.2 Funções de seleção para LSH

Em DUARTE *et al.* (2017), quatro funções de seleção foram avaliadas em tarefas de indexação e recuperação de fontes na etapa de busca heurística do problema de plágio externo. Essas funções de seleção foram utilizadas em uma etapa do *workflow* LSH, com utilização de permutação aleatória, proposto em DUARTE *et al.* (2017) e apresentado na seção 3.2. As funções avaliadas foram: Minwise, Minmaxwise, Minmax Circular Sector Arcs Lower Bound (CSA_L) e Minmax Circular Sector Arcs Full Bound (CSA). Além disso, o *workflow* foi executado utilizando representação de documentos como um conjunto de *token*, isto é, abordagens Bag-Of-Words. Nesta seção são apresentadas essas funções de seleção.

4.2.1 Minwise Hashing

O método Minwise Hashing, proposto por BRODER (1997); BRODER *et al.* (1997), é uma função que seleciona um valor mínimo de um conjunto de assinaturas em cada aplicação de uma função de permutação independente π_i. Minwise Hashing está baseado na propriedade 4.2, onde a probabilidade do valor de mínimo de um conjunto A ser igual ao mínimo de um conjunto B é igual a similaridade de Jaccard.

$$Pr[min(\pi_i(A)) = min(\pi_i(B))] = J(A, B) \tag{4.2}$$

Um exemplo prático do uso do Minwise Hashing, pode ser descrito como: a partir de um documento A, de conjunto de assinaturas $L_j(A) = \{1, 2, 3, 4, 5\}$, uma função de permutações π_3 é aplicada gerando o conjunto $\pi_3(A) = \{5, 3, 1, 2, 4\}$, onde é selecionado o valor de mínimo $min(\pi_3(A)) = 1$, que representa o item 3 do conjunto de assinaturas de $L_j(A)$ nesta permutação.

Como visto no capítulo 3, a aplicação do Minwise Hashing visa amenizar o custo com armazenamento e comparação entre documentos, pois seleciona os elementos mais representativos do documento, dando velocidade a busca por documentos próximos (BRODER, 1997).

4.2.2 Minmaxwise Hashing

Como a etapa de permutação é a etapa mais lenta do *workflow* de permutação aleatória, JI *et al.* (2013) propuseram o método Minmaxwise (DUARTE *et al.*, 2017; JI *et al.*, 2013). O método Minmaxwise necessita de menos permutações para selecionar o mesmo número de assinaturas quando comparado ao Minwise, reduzindo o tempo de execução, sem perder a probabilidade de colisão entre pares (DUARTE *et al.*, 2017; JI *et al.*, 2013).

Para reduzir o tempo, além de selecionar o valor mínimo, de um conjunto de assinaturas em cada aplicação de uma função de permutação independente π_i, seleciona o valor máximo do mesmo conjunto. Isso ocorre com base nas propriedades 4.2 e 4.3, onde a probabilidade dos valores de mínimo e máximo de um conjunto A ser igual ao mínimo e máximo de um conjunto B é igual similaridade de Jaccard.

$$Pr[max(\pi_i(A)) = max(\pi_i(B))] = J(A, B) \tag{4.3}$$

Com base nisso, a função de seleção Minmaxwise mantém o maior limite inferior e o menor limite superior das permutações aleatórias $\frac{k}{2}$, em vez de selecionar apenas um limite de k-permutações, como acontece com o Minwise hashing. Assim, o método Minmaxwise reduz o número de permutações aleatórias necessárias e o tempo de permutação pela metade (JI *et al.*, 2013).

Um exemplo prático do uso do Minmaxwise, pode ser descrito como: a partir de um documento A, de conjunto de assinaturas $L_j(A) = \{1, 2, 3, 4, 5\}$, uma função de permutações π_3 é aplicada gerando o conjunto $\pi_3(A) = \{5, 3, 1, 2, 4\}$, onde é selecionado o valor de mínimo $min(\pi_3(A)) = 1$ e o valor de máximo $min(\pi_3(A)) = 5$, que representam o item 3 e 1, respectivamente, do conjunto de assinaturas de $L_j(A)$ nesta permutação.

4.2.3 CSA_L e CSA

Os métodos Minmax Circular Sector Arcs Lower Bound (CSA_L) e Minmax Circular Sector Arcs Full Bound (CSA) foram propostos com o mesmo objetivo do método Minmaxwise: Melhorar o tempo da etapa de permutação selecionando mais assinaturas por permutação (DUARTE *et al.*, 2017). Os métodos propostos por DUARTE *et al.* (2017) representam cada conjunto de valores resultantes S da aplicação de uma função de permutação π_i como um triângulo retângulo, onde o valor de mínimo do conjunto é aplicado a um cateto e o valor de máximo aplicado a hipotenusa do triângulo, assim, o cateto $K_{S_i,n}$, a partir da aplicação da equação 4.4, pode ser utilizado como assinaturas para representar o conjunto S.

$$K_{S_i,n} = \sqrt{max^2_{S_i,n} - max^2_{S_i,n}} \tag{4.4}$$

Contudo, esse método consegue contemplar apenas os casos em que os valores de mínimo e máximo de dois conjuntos são iguais. Para aumentar o alcance da propriedade triangular, Arcos de Setores Circulares (ASC) foram propostos. Os ASC aumentam a chance de representar conjuntos próximos, pois a seleção de valores de assinaturas passam a ser realizada em valores que estão contidos no mesmo círculo de centro (0,0) e raio=$max^2_{S_i,n}$, como na figura 4.1.

Um exemplo prático pode ser visto na figura 4.1, onde conjuntos $\pi_3(A) = \{5, 3, 1, 2, 4\}$ e $\pi_3(D) = \{5, 3, 2, 4\}$, onde os valores de mínimos e máximos são $[1, 5]$ e $[2, 5]$, respectivamente. A partir da aplicação do ASC, A e D estão no mesmo círculo, pois possuem o mesmo valor de $max_{S_i,n}$. Apesar de $K_{S_i,3}$ não conseguir representar D com o mesmo valor de A, por estarem no mesmo setor do círculo, A e D podem ser representados pelo valor do limite inferior do intervalo $\lfloor K_{S_i,n} \rfloor = K^{inf}_{S_i,3} = K^{inf}_{A,3} = K^{inf}_{D,3}$.

A diferença entre o CSA_L e o CSA é que em cada permutação π_n o CSA_L seleciona o conjunto de valores $\{min(\pi_n(S_i)), max(\pi_n(S_i)), K^{inf}_{S_i,n}\}$ e o CSA, além de selecionar o mesmo conjunto de valores, seleciona também o valor do limite superior do intervalo $\lceil K_{S_i,n} \rceil = K^{sup}_{S_i,n}$, recuperando o conjunto $\{min(\pi_n(S_i)), max(\pi_n(S_i)), K^{inf}_{S_i,n}, K^{sup}_{S_i,n}\}$. No exemplo da figura 4.1, CSA_L selecionaria os valores $A = \{1, 5, 4\}$ e $D = \{2, 5, 4\}$ e o CSA selecionaria os valores $A = \{1, 5, 4, 5\}$ e $D = \{2, 5, 4, 5\}$.

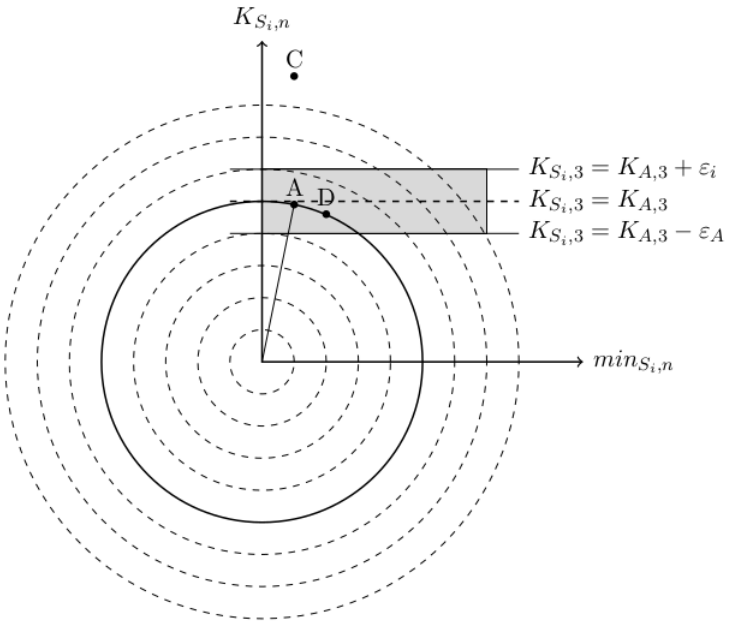


Figura 4.1 – Exemplo de aplicação de Arcos de Setores Circulares nos conjuntos A e D. Reprodução baseada em DUARTE *et al.* (2017)

4.2.3.1 Avaliação e resultados

Conforme afirmado em JI *et al.* (2013), aplicar funções de permutação é uma etapa muito custosa, assim, quanto menor for o número de permutações, mais rápido será o método. Além disso, DUARTE *et al.* (2017) definiu que a etapa de permutações é a etapa mais custosa do *workflow* proposto. Com base nisso, JI *et al.* (2013) e DUARTE *et al.* (2017) propuseram os métodos apresentados anteriormente, Minmax hashing, Minmax Circular Sector Arcs Lower Bound (CSA_L) e Minmax Circular Sector Arcs Full Bound (CSA), visando a diminuição da quantidade de funções de permutação independentes a serem aplicadas.

A tabela 4.1, adaptada de DUARTE (2017), apresenta a proposta de um cálculo para o custo computacional quando se representa um conjunto assinaturas S_i com P permutações, em cada método proposto. Os métodos CSA_L e CSA selecionam 3 e 4 valores por permutação, respectivamente, enquanto o Minwise Hashing seleciona apenas um valor, reduzindo a quantidade de tempo gasta em 66% e 75%. Minmaxwise hashing reduz apenas 50%, pois seleciona apenas 2 valores por permutação (DUARTE, 2017).

Método	custo por permutação	$min_{S_{i,n}}$	$max_{S_{i,n}}$	$K^{inf}_{S_{i,n}}$	$K^{sup}_{S_{i,n}}$
Minwise Hashing	$Px\|S_i\|$	Sim	-	-	-
Minmaxwise Hashing	$\frac{P}{2})x\|S_i\|$	Sim	Sim	-	-
MinMaxCSA$_L$	$(\frac{P}{3})x\|S_i\|$	Sim	Sim	Sim	-
MinMaxCSA	$(\frac{P}{4})x\|S_i\|$	Sim	Sim	Sim	Sim

Tabela 4.1 – Comparando custo computacional para seleção de valores. Adaptado de DUARTE (2017)

Com base nesse custo computacional apresentado, DUARTE *et al.* (2017) realizou dois conjuntos de experimentos com objetivo de validar a eficiência real dos métodos CSA$_L$ e CSA. Para isso, usou como base para comparação alguns métodos tradicionais, como: o índice invertido usando palavras, Minwise Hashing e Minmaxwise Hashing. O primeiro conjunto de experimentos avaliou se os métodos propostos estimam a similaridade de Jaccard, já o segundo experimento, avaliou se são aplicáveis a busca heurística no problema de plágio externo (DUARTE *et al.*, 2017).

Para o primeiro experimento, CSA$_L$ e CSA se mostraram 33% e 50% mais rápidos que o Minmaxwise hashing. Além disso, apesar de serem menos precisos que o MinMaxwise hashing, CSA$_L$ e CSA apresentaram resultados de erro na mesma ordem de magnitude (DUARTE *et al.*, 2017).

Para o segundo experimento, na tarefa de indexação, a Minmaxwise mostrou um rendimento 20% maior que o método Minwise, enquanto os métodos CSA$_L$ e CSA tiveram uma taxa de vazão de 31% e 36% maior do que o método Minwise. Na tarefa de recuperação de documentos candidatos a plágio, o método Minmaxwise levou metade do tempo, enquanto os métodos CSA$_L$ e CSA levaram um terço e um quarto do tempo gasto pelo Minwise para extrair consultas com o mesmo número de assinaturas. Além disso, o *recall* de todos os métodos avaliados foi mensurado e o método Minmaxwise obteve os maiores resultados de *recall*, seguido por valores menores de *recall* dos métodos CSA$_L$ e CSA (DUARTE *et al.*, 2017).

Capítulo 5

Propostas de Particionamento do LSH

Na execução da tarefa de busca heurística usando Locality-Sensitive Hashing, um documento é *tokenizado*, os *tokens* gerados são representados por assinaturas, essas sofrem uma etapa de permutação, e em sequência disso, a aplicação de uma função para seleção de assinaturas que representem o documento na avaliação de similaridade, como é definido no capítulo 3. Como apresentado na 3.2.3, a permutação é a etapa que leva mais tempo dentre as etapas do LSH. A partir disso, foram criadas algumas estratégias com o objetivo de diminuir o tempo de permutação.

As estratégias realizam o particionamento das assinaturas permutadas, gerando pequenos grupos, chamados células, onde são realizadas as seleções de valores de mínimos e máximos para utilização no cálculo de similaridade entre os documentos. Através dessas múltiplas seleções, são selecionadas mais assinaturas por permutação do que normalmente seriam selecionadas, levando a diminuição da quantidade de permutações necessárias. Em consequência disso, por diminuir o tempo de permutação, diminui o tempo utilizado para realização das etapas de indexação e de busca.

Nas seções seguintes são apresentadas as estratégias de particionamento das permutações, *Remainder at Cell* e *Remainder at End*, além de outras estratégias de particionamento, como *Distributed at Cell* e o particionamento do vocabulário.

5.1 Estratégias de particionamento

Particionamento das permutações é a estratégia que realiza, a partir de um conjunto de assinaturas oriundos do processo de permutação $\pi_i(L_j)$, onde $\pi_i(L_j) \in L'_{j,N}$, a divisão entre p células $\{c_l(\pi_i(L_j))\}_{l=1}^{l=p}$. A partir de cada célula c_l gerada é

aplicada uma função de seleção, onde são recuperados valores de mínimos e máximos que serão utilizados para o cálculo de similaridade posterior.

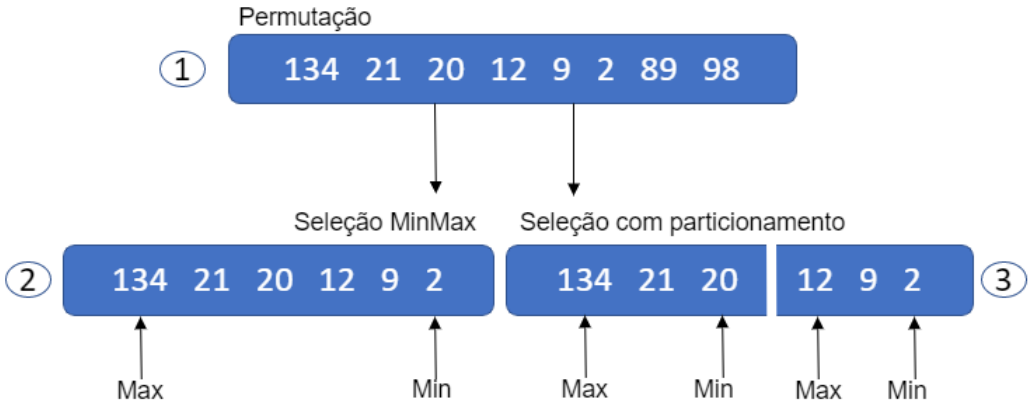


Figura 5.1 – Método de seleção por particionamento de permutações

No item 1 da figura 5.1, é apresentado o conjunto de valores $\{134, 21, 20, 12, 9, 2, 89, 98\}$ gerados no processo de permutação. Após tal conjunto gerado, são vistos exemplos de seleção de valores utilizando o MinMax sem estratégia de particionamento, item 2, e com a estratégia de particionamento, item 3. É notável que na seleção sem particionamento, são selecionados somente dois valores $\{134, 2\}$, um de mínimo e um de máximo, enquanto na abordagem com particionamento, com apenas uma permutação e duas células, são selecionados o dobro de valores $\{134, 20, 12, 2\}$.

A partir do exemplo acima citado, pode ser notada a relação entre o número de assinaturas selecionado e o número de células escolhido. O número de assinaturas $|S_{i,j}|$ é o resultado de $p * 2$, sendo p o número de células escolhido e 2, o número de assinaturas selecionadas ao utilizar o MinMax. No exemplo, $p = 2$, logo o $|S_{i,j}| = 4$.

5.1.1 Lidando com o resto do particionamento

No processo de particionamento, valores podem não ser contemplados com um lugar em alguma célula. Isso ocorre como em um exemplo de divisão simples, onde valores não divisíveis pelo seu divisor produzem resto. Isso pode ocorrer caso a quantidade de assinaturas criadas $|L_j|$ não é divisível pelo número de células p escolhida. Exemplo, de um documento são geradas 753 assinaturas e o número de células escolhidas é 2. Nesse caso, o valor de resto apresentado é 1. Para solucionar isso e possibilitar que o valor restante entre em alguma célula, foram criadas estratégias *Remainder at Cell* e *Remainder at End* apresentadas nas seções seguintes.

5.1.1.1 Remainder at Cell

Na estratégia Remainder at Cell (RaC), o conjunto de assinaturas geradas na etapa de permutação $\pi_i(L_j)$ são igualmente distribuídas entre cada célula gerada no processo de divisão. Com base nisso, $k = x \times p + r$ onde x é o tamanho inicial para todas as células e r $(0 \leq r < p)$ é o número de assinaturas no conjunto de resto. A distribuição das assinaturas restantes é realizada de forma sequêncial, para as primeiras r células $\{c_l(\pi_i(L_j))\}_{l=2}^{l=r}$, do inicio ao fim do conjunto de permutações $\pi_i(L_j)$. Vale a pena notar que $\{c_l(\pi_i(L_j))\}_{l=1}^{l=r}$ células possuirão $x + 1$ assinaturas enquanto $\{c_l(\pi_i(L_j))\}_{l=r}^{l=p}$ células possuirão x assinaturas.

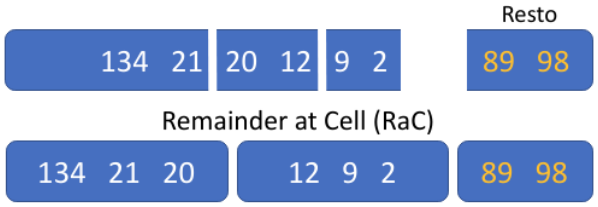


Figura 5.2 – Método de seleção RaC

Assim, a estratégia divide o conjunto de assinaturas geradas pela permutação $\pi_i(L_j)$ em p células, aloca $x + 1$ assinaturas para as primeiras r células da divisão e x assinaturas na última célula gerada na divisão. No exemplo da figura 5.2, o conjunto de assinaturas oriundas de uma permutação é $\{134, 21, 20, 12, 9, 2, 89, 98\}$ e o número de células $p = 3$. Caso fosse realizada uma divisão comum em células, o conjunto $\{89, 98\}$ de resto seria gerado. A fim de não gerar um conjunto de resto, a estratégia RaC é utilizada alocando assinaturas a mais que em cada célula desde a célula inicial. Como visto no exemplo, a célula inicial em um processos de divisão comum conteria os elementos $\{134, 24\}$ e passa a conter mais o elemento 20, se tornando $\{134, 24, 20\}$. Outro exemplo que pode ser visto é a célula que antes receberia $\{20, 12\}$, recebendo $\{12, 9, 2\}$. Esse processo é realizado em todas as células, do inicio ao fim.

5.1.1.2 Remainder at End

Na estratégia Remainder at End (RaE), o conjunto de assinaturas geradas da permutação $\pi_i(L_j)$ são divididas em p células, em sequência disso, $\lfloor \frac{k}{p} \rfloor$ assinaturas são alocadas para as primeiras $p - 1$ células e, as assinaturas restantes são alocadas na última célula.

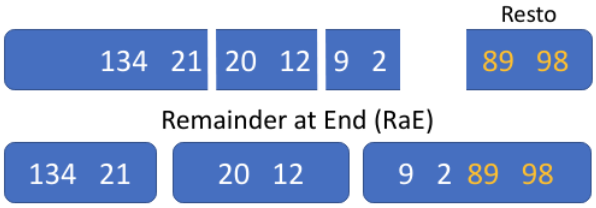


Figura 5.3 – Método de seleção RaE

Em outras palavras, as assinaturas restantes do processo de divisão são alocadas na última célula do conjunto de células. No exemplo da figura 5.3, o conjunto de assinaturas oriundas de uma permutação é $\{134, 21, 20, 12, 9, 2, 89, 98\}$ e o número de células $p = 3$. Ao ser realizada a divisão em células, o conjunto $\{89, 98\}$ de resto é gerado. A partir disso, a estratégia Remainder at End (RaE) é utilizada alocando o conjunto $\{89, 98\}$ na última célula da divisão. A célula que antes continha somente os valores $\{9, 2\}$, passa a conter $\{9, 2, 89, 98\}$.

5.1.1.3 Algoritmos

Esta seção apresenta e formaliza os algoritmos das estratégias RaC e RaE como a etapa de função de seleção do LSH. Ambos os algoritmos recebem como entrada o número de células p a serem criada e o $i-ésimo$ conjunto de assinaturas permutadas $\pi_i(L_j)$, e como saída, o conjunto de assinaturas selecionadas $S_{i,j}$ a partir de todas as p células, onde $\cup c_l = \pi_i(|L_j|)$ e $c_l \cap c_m = \emptyset, \forall j \neq m$.

A maior diferença entre RaC (Algoritmo 3) e RaE (Algoritmo 4) está no tamanho da célula, pois o RaC cria r células com $\lfloor \frac{|L_j|}{P} \rfloor + 1$ assinaturas e P - r células com $\lfloor \frac{|L_j|}{P} \rfloor$ assinaturas enquanto que RaE cria $P - 1$ células com $\lfloor \frac{|L_j|}{P} \rfloor$ assinaturas e uma célula com $\lfloor \frac{|L_j|}{P} \rfloor$ + r assinaturas.

Algoritmo 3: Função de seleção de assinaturas RaC

Entrada: $\pi_i(L_j) = \{f_1, f_2, f_3, \ldots, f_{|L_j|}\}$, $P =$ total de células

Saída: $S_{i,j}$

Inicialização: $S_{i,j} \leftarrow \{\}$,

$x \leftarrow \lfloor \frac{|L_j|}{P} \rfloor$,

$r \leftarrow |L_j| - x \times P$

início

`/* Preenchendo primeiras células com` $x+1$ `valores visando */`

`/* incluir valores sem criar restos */`

para $l \leftarrow 1$ ***até*** r **faça**

$minId \leftarrow \infty^+$

$maxId \leftarrow \infty^-$

para $a \leftarrow 1 + ((x+1) \times (l-1))$ ***até*** $(x+1) \times l$ **faça**

se $minId > f_a$ **então**

$minId \leftarrow f_a$

se $maxId < f_a$ **então**

$maxId \leftarrow f_a$

$S_{i,j} \leftarrow S_{i,j} \cup \{minId, maxId\}$

`/* Preenchendo células restantes` $P - r$ `com` x `valores */`

`/* visando incluir valores sem criar restos */`

para $l \leftarrow 1$ ***até*** $P - r$ **faça**

$minId \leftarrow \infty^+$

$maxId \leftarrow \infty^-$

$y \leftarrow (x+1) \times r$

para $a \leftarrow 1 + y + (x \times (l-1))$ ***até*** $y + (x \times l)$ **faça**

se $minId > f_a$ **então**

$minId \leftarrow f_a$

se $maxId < f_a$ **então**

$maxId \leftarrow f_a$

$S_{i,j} \leftarrow S_{i,j} \cup \{minId, maxId\}$

.

Algoritmo 4: Função de seleção de assinaturas RaE

Entrada: $\pi_i(L_j) = \{f_1, f_2, f_3, \ldots, f_{|L_j|}\}$, $P =$ total de células

Saída: $S_{i,j}$

Inicialização: $S_{i,j} \leftarrow \{\}$,
$x \leftarrow \lfloor \frac{|L_j|}{P} \rfloor$,
$r \leftarrow |L_j| - x \times P$
início
  para $l \leftarrow 1$ ***até*** P **faça**
    $minId \leftarrow \infty^+$
    $maxId \leftarrow \infty^-$
    `/* Preenchendo células` $l < P$ `com` x `valores */`
    se $l < P$ **então**
      para $a \leftarrow 1 + x \times (l-1)$ ***até*** $x \times l$ **faça**
        se $minId > f_a$ **então**
          $minId \leftarrow f_a$
        se $maxId < f_a$ **então**
          $maxId \leftarrow f_a$
    senão
      `/* Preenchendo célula final` P `com` $x + r$ `valores */`
      para $a \leftarrow 1 + x \times (l-1)$ ***até*** $|L_j|$ **faça**
        se $minId > f_a$ **então**
          $minId \leftarrow f_a$
        se $maxId < f_a$ **então**
          $maxId \leftarrow f_a$
    $S_{i,j} \leftarrow S_{i,j} \cup \{minId, maxId\}$

5.2 Análise da probabilidade de colisão

Na equação (5.1) e (5.2), para dois conjuntos selecionados aleatoriamente A e B, $c_l(\pi_i(A))$ e $c_l(\pi_i(B))$ são as l-ésimas células resultantes do particionamento da i-ésima permutação de A e B, respectivamente. Então, a equação (5.1) apresenta uma variável aleatória de Bernoulli X'_{ij}, onde $X'_{ij} = 1$ se as células correspondentes do conjunto A e B compartilham o mesmo valor de mínimo. Além disso, a equação (5.2) define Y'_{ij}, analogamente para equação (5.1), onde $Y'_{ij} = 1$ se as células correspondentes dos conjuntos A e B compartilham o mesmo valor de máximo.

$$X'_{ij} = \begin{cases} 1 & \text{se } min(c_l(\pi_i(A))) = min(c_l(\pi_i(B))) \\ 0 & \text{caso contrário} \end{cases} \tag{5.1}$$

$$Y'_{ij} = \begin{cases} 1 & \text{se } max(c_l(\pi_i(A))) = max(c_l(\pi_i(B))) \\ 0 & \text{caso contrário} \end{cases} \tag{5.2}$$

Os eventos de colisão na células c_l são relacionados com as duas variáveis mencionadas anteriormente, X'_{ij} e Y'_{ij} quando $X'_{ij} = 1$ e $Y'_{ij} = 1$. Com base nisso, na permutação π_i, os eventos de interesse $X'_{ij} = 1$ são os eventos em que a célula c_l, de partições em $\pi_i(A)$ e $\pi_i(B)$, tem o mesmo valor de mínimo enquanto que eventos de interesse $Y'_{ij} = 1$ são os eventos em que a célula c_l tem o mesmo valor de máximo. Contudo, X'_{ij} e Y'_{ij} não podem lidar com eventos em que c_l tem os mesmos valores de mínimos e máximos ou os eventos em que pelo menos uma célula deveria ter o mesmo valor de mínimo e máximo em partições de $\pi_i(A)$ e $\pi_i(B)$.

Uma variável aleatória Z', da equação 5.3, é relacionada com os eventos mencionados em que X'_{ij} e Y'_{ij} não conseguem lidar. Onde $Z' > 0$ se existe ao menos uma célula c_l e ao menos uma permutação π_i com $X'_{ij} = 1$ ou $Y'_{ij} = 1$. Além disso, os eventos de colisão das estratégia de partição RaC e RaE são relacionadas com a variável aleatória $Z' > 0$.

$$Z' = \frac{1}{k} \sum_{i=1}^{\frac{\lfloor k/p \rfloor}{2}} \sum_{j=1}^{p} X'_{ij} + Y'_{ij} \tag{5.3}$$

Dadas as definições anteriores, os lemas seguintes são provados. (As provas podem ser vistas no Apêndice A):

Lema 1. $\Pr(X'_{ij} = 1) = \mathbb{E}[X'_{ij}] = (\frac{1}{2} + \frac{1}{2p}) \times J(A, B)$

Lema 2. $\Pr(Y'_{ij} = 1) = \Pr(X'_{ij} = 1)$

Lema 3. $\Pr(Y'_{ij} = 0 | X'_{ij} = 0) = \Pr(Y'_{ij} = 0)$

O Teorema 1 mostra a probabilidade de dois conjuntos A e B não colidirem nas estratégias RaC e RaE. Além disso, os corolários 1.1 e 1.2 foram deduzidos a partir do Teorema 1 onde o corolário 1.2 apresenta o intervalo de probabilidade de colisão para as estratégias RaC e RaE. Além disso, a tabela 5.1 mostra o viés de similaridade de Jaccard no intervalo de probabilidade de colisão em RaC e RaE, e vale a pena notar que a probabilidade de colisão aumenta quando a similaridade de Jaccard aumenta.

O corolário 1.1 mostra que a probabilidade de colisão aumenta, desde que $\Pr(Z' = 0)$ diminua, quando o número de células aumenta e devido ao Teorema 1 e os corolários 1.1 e 1.2, a probabilidade de dois conjuntos A e B colidirem nas

estratégias RaC e RaE aumenta quando $J(A,B)$ aumenta e quando o número de células aumenta.

Teorema 1. $\Pr(Z'=0) = \frac{\left(1-\frac{1}{2}(1+\frac{1}{p})\times J(A,B)\right)^2}{(2^{p-1})^{\lfloor \frac{k}{p} \rfloor}}$

Corolário 1.1. *se* $p > k$ *então*

$$\Pr(Z'=0) = \left(1-\tfrac{1}{2}(1+\tfrac{1}{p})\times J(A,B)\right)^2$$

Corolário 1.2. Z' *probabilidade de colisão* $\Pr(Z'>0)$ *propriedade:*

$$1-\left(1-J(A,B)\right)^2 \geq \Pr(Z'>0) > 1-\left(1-\tfrac{J(A,B)}{2}\right)^2$$

Tabela 5.1 – Viés de Jaccard no intervalo de probabilidade de colisão em RaC and RaE

$J(A,B)$	intervalo de valores $\Pr(Z'>0)$
0	(0; 0]
0,2	(0,36; 0,19]
0,4	(0,64; 0,36]
0,6	(0,84; 0,51]
0,8	(0,96; 0,64]
1	(1,00; 0,75]

5.3 Outras estratégias de particionamento

5.3.1 Distributed at Cell

A estratégia Distributed at Cell (DaC) é uma estratégia de resolução de resto da divisão, como o RaC e o RaE, onde um conjunto de assinaturas geradas a partir da permutação $\pi_i(L_j)$ são divididas em p células a partir de um processo de *Round-robin* entre as p células. Esse processo de *Round-robin* é uma divisão igualitária entre as células, que se inicia na primeira célula, e que é realizada de forma circular até que o conjunto de assinaturas esteja totalmente compreendido entre células.

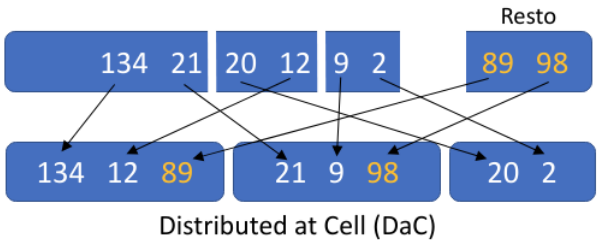


Figura 5.4 – Método de seleção DaC

No exemplo da figura 5.4 é apresentada uma divisão comum entre o conjunto de assinaturas $\{134, 21, 20, 12, 9, 2, 89, 98\}$ e o número de células $p = 3$. A partir dessa divisão, seria criado o conjunto de resto $\{89, 98\}$. A fim de evitar a criação de um conjunto de resto, o conjunto $\{134, 21, 20, 12, 9, 2, 89, 98\}$ é distribuído de forma igualitária e circular entre as p células, criando as células $\{134, 12, 89\}$, $\{21, 9, 98\}$ e $\{20, 2\}$.

5.3.1.1 Algoritmo

Esta seção apresenta e formaliza o algoritmo da estratégia DaC como a etapa de função de seleção do LSH. O algoritmo, como nas estratégias RaC e RaE apresentadas na seção 5.1.1.3, recebe como entrada o número de células p a serem criada e o $i - ésimo$ conjunto de assinaturas permutadas $\pi_i(L_j)$, e como saída, o conjunto de assinaturas selecionadas $S_{i,j}$ a partir de todas as p células, onde $\cup c_l = \pi_i(|L_j|)$ e $c_l \cap c_m = \emptyset, \forall j \neq m$.

Algoritmo 5: Função de seleção de assinaturas DaC

Entrada: $\pi_i(L_j) = \{f_1, f_2, f_3, \ldots, f_{|L_j|}\}$, $P =$ total de células
Saída: $S_{i,j}$

```
1  Inicialização:  S_{i,j} ← {}, x ← |L_j|
2  início
3      para l ← 1 até P faça
4          minId ← ∞⁺
5          maxId ← ∞⁻
6          a ← l
7          enquanto a <= x faça
8              se minId > f_a então
9                  minId ← f_a
10             se maxId < f_a então
11                 maxId ← f_a
12             a ← a + P
13         S_{i,j} ← S_{i,j} ∪ {minId, maxId}
```

5.3.2 Particionando a partir do vocabulário

Ao contrário do particionamento por permutação que divide as assinaturas geradas após a etapa de permutação $\pi_i(L_j)$ em p células, o particionamento a partir do vocabulário tem como objetivo particionar o vocabulário T, representado por assinaturas, em células, antes mesmo da etapa de permutação. Como pode ser visto

na figura 5.5, O vocabulário T é particionado em c_1 e c_2, após isso, em cada célula, é aplicada uma função de permutação π_i e selecionados, do conjunto resultante, valores de máximos e mínimos, usando Minmaxwise Hashing.

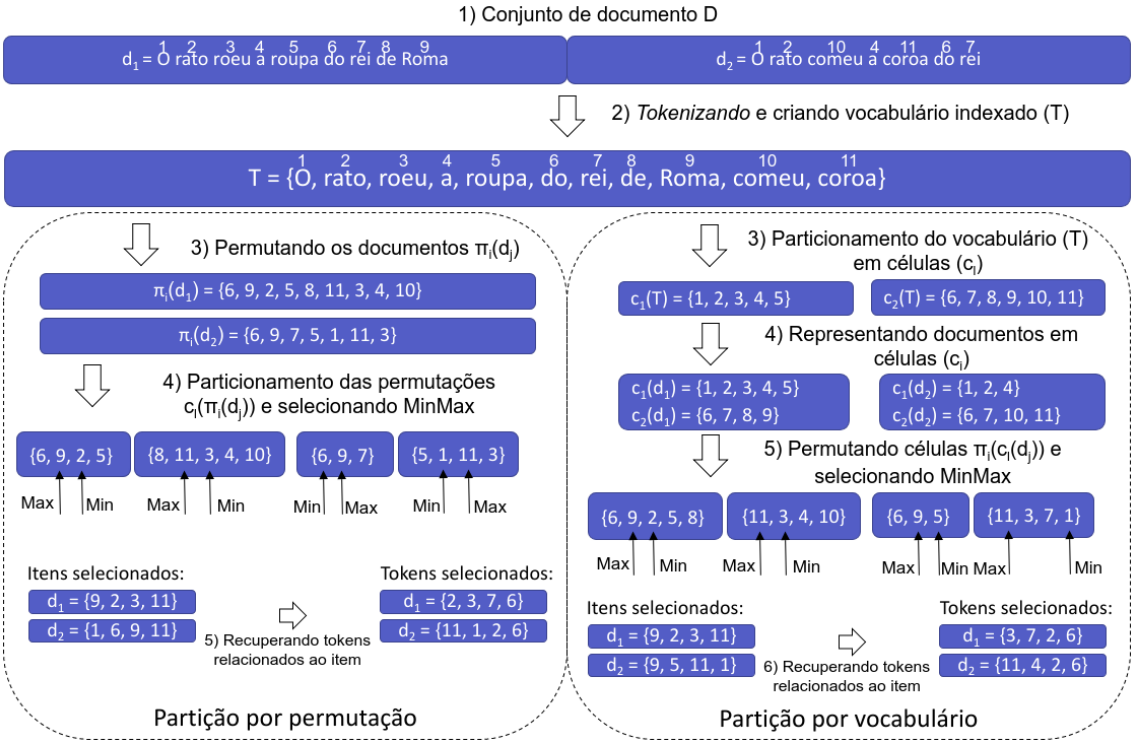


Figura 5.5 – Método de seleção por particionamento do vocabulário

Algo que não é possível no particionamento por permutações, mas que é possível no particionamento por vocabulário, é a criação de células vazias. Com base no particionamento proposto no exemplo 5.5, se um documento novo d_3 = "O rato roeu a roupa" passasse pelo mesmo processo de d_1 e d_2, as assinaturas geradas, a partir de $L(d_3) = \{1, 2, 3, 4, 5\}$, seriam alocadas em $c_1 = \{1, 2, 3, 4, 5\}$ e $c_2 = \{\}$. Nesta situação, os valores representativos para célula c_2 não são gerados, sendo utilizados somente os valores de c_1 para o cálculo de similaridade.

Como no particionamento por permutações, a estratégia de particionamento por vocabulário pode gerar resto, sendo aplicadas as mesmas soluções apresentadas anteriormente: Remainder at Cell (RaC), Remainder at End (RaE) e Distributed at Cell (DaC). Na seção a seguir são apresentados os algoritmos das estratégias quando aplicadas ao particionamento por vocabulário.

5.3.2.1 Algoritmos

Os algoritmos 6, 7 e 8 apresentados a seguir aplicam as estratégias RaC, RaE e DaC, respectivamente, ao particionamento por vocabulário. A diferença entre os

algoritmos apresentados anteriormente, nas seções 5.1.1.3 e 5.3.1.1, é basicamente os seguintes:

- Entrada passa a ter as assinaturas do vocabulário L_j, quando na divisão por permutações são recebidas o conjunto de assinaturas permutadas;
- Entrada passa a ter o documento d_j, que passa a ser necessário na verificação se o *token* é pertencente a d_j para aplicação da função de permutação π_i;
- Entrada passa a ter função de permutação π_i para a aplicação após o particionamento da célula.

Algoritmo 6: RaC aplicada no particionamento por vocabulário

Entrada: $L_j = \{L_1, L_2, L_3 \ldots, L_{|L_j|}\}$,
P = total de células,
d_j = documento,
π_i = função de permutação
Saída: $S_{i,j}$

Inicialização: $S_{i,j} \leftarrow \{\}$
$x \leftarrow \lfloor \frac{|L_j|}{P} \rfloor$
$r \leftarrow |L_j| - x \times P$
início
  /* Preenchendo primeiras células com $x+1$ valores visando */
  /* incluir valores sem criar restos */
  para $l \leftarrow 1$ ***até*** r **faça**
    $minId \leftarrow \infty^+$
    $maxId \leftarrow \infty^-$
    para $a \leftarrow 1 + ((x+1) \times (l-1))$ ***até*** $(x+1) \times l$ **faça**
      se $a \in d_j$ **então**
        se $minId > \pi_i(a)$ **então**
          $minId \leftarrow \pi_i(a)$
        se $maxId < \pi_i(a)$ **então**
          $maxId \leftarrow \pi_i(a)$
    $S_{i,j} \leftarrow S_{i,j} \cup \{minId, maxId\}$
  /* Preenchendo células restantes $P - r$ com valores x */
  /* incluir valores sem criar restos */
  para $l \leftarrow 1$ ***até*** $P - r$ **faça**
    $minId \leftarrow \infty^+$
    $maxId \leftarrow \infty^-$
    $y \leftarrow (x+1) \times r$
    para $a \leftarrow 1 + y + (x \times (l-1))$ ***até*** $y + (x \times l)$ **faça**
      se $a \in d_j$ **então**
        se $minId > \pi_i(a)$ **então**
          $minId \leftarrow \pi_i(a)$
        se $maxId < \pi_i(a)$ **então**
          $maxId \leftarrow \pi_i(a)$
    $S_{i,j} \leftarrow S_{i,j} \cup \{minId, maxId\}$

.

Algoritmo 7: RaE aplicada no particionamento por vocabulário

Entrada: $L_j = \{L_1, L_2, L_3 \ldots, L_{|L_j|}\}$,
P = total de células,
d_j = documento,
π_i = função de permutação
Saída: $S_{i,j}$

Inicialização: $S_{i,j} \leftarrow \{\}$, $x \leftarrow \lfloor \frac{|L_j|}{P} \rfloor$, $r \leftarrow |L_j| - x \times P$ **início**
  para $l \leftarrow 1$ ***até*** P **faça**
    $minId \leftarrow \infty^+$
    $maxId \leftarrow \infty^-$
    `/* Preenchendo células` $l < P$ `com` x `valores */`
    se $l < P$ **então**
      para $a \leftarrow 1 + x \times (l-1)$ ***até*** $x \times l$ **faça**
        se $a \in d_j$ **então**
          se $minId > \pi_i(a)$ **então**
            $minId \leftarrow \pi_i(a)$
          se $maxId < \pi_i(a)$ **então**
            $maxId \leftarrow \pi_i(a)$
    senão
      `/* Preenchendo célula final` P `com` $x + r$ `valores */`
      para $a \leftarrow 1 + x \times (l-1)$ ***até*** $|L_j|$ **faça**
        se $a \in d_j$ **então**
          se $minId > \pi_i(a)$ **então**
            $minId \leftarrow \pi_i(a)$
          se $maxId < \pi_i(a)$ **então**
            $maxId \leftarrow \pi_i(a)$
    $S_{i,j} \leftarrow S_{i,j} \cup \{minId, maxId\}$

Algoritmo 8: DaC aplicada no particionamento por vocabulário

Entrada: $L_j = \{L_1, L_2, L_3 \ldots, L_{|L_j|}\}$,
$P =$ total de células,
$d_j =$ documento,
$\pi_i =$ função de permutação
Saída: $S_{i,j}$

```
1  Inicialização:  S_{i,j} ← {}, x ← |L_j|
2  início
3      para l ← 1 até P faça
4          minId ← ∞^+
5          maxId ← ∞^−
6          enquanto a <= x faça
7              se a ∈ d_j então
8                  se minId > π_i(a) então
9                      minId ← π_i(a)
10                 se maxId < π_i(a) então
11                     maxId ← π_i(a)
12             a ← a + P
13         S_{i,j} ← S_{i,j} ∪ {minId, maxId}
```

Capítulo 6

Experimentos e Resultados

Este capítulo descreve a avaliação das estratégias de particionamento das permutação, RaE e RaC, visando a validação da eficiência real das estratégias propostas com relação a estimar a similaridade de Jaccard e a tarefa de busca heurística. Para isso, foram realizados os experimentos: Estimativa de Similaridade de Jaccard par a par (PJSE) e Busca Heurística de Plágio Externo (EPHR). Os experimentos de PJSE e EPHR foram executados em um computador Intel i7 (3.4 Ghz) de 16 GB de RAM usando como base para comparação os métodos Minmaxwise Hashing e o BM25.

Minmaxwise Hashing foi o *baseline* dos experimentos em DUARTE *et al.* (2017) para validação dos métodos CSA_L e CSA, que apesar de ter apresentado resultados inferiores, em vazão de indexação e tempo de busca, apresentou valores de *recall* superiores com relação a CSA_L e CSA, sendo escolhido como método base para comparação nos experimentos PJSE e EPHR. Além disso, por buscar melhorar os tempos usando o Minmaxwise em suas estratégias, CSA_L e CSA foram utilizados como base de comparação em EPHR.

Apesar de ter sido utilizado como base de comparação em JI *et al.* (2013) e em DUARTE *et al.* (2017), Minwise Hashing apresenta resultados inferiores em tempos de indexação e busca, além de menores valores de *recall*, quando comparado ao método Minmaxwise Hashing, assim, não sendo utilizado como base de comparação nos experimentos deste capítulo.

Outro método usado como base de comparação para os experimentos PJSE e EPHR, BM25 é um dos mais populares e efetivos algoritmos de ranqueamento em busca e recuperação de informação (BAEZA-YATES & RIBEIRO-NETO, 2011) e foi escolhido para comparação nos experimentos por não ter sido comparado em DUARTE *et al.* (2017) e não sendo utilizado na tarefa de indexação em nenhum dos trabalhos apresentados no PAN para a tarefa de busca heurística no plágio externo.

A Seção 6.1 mostra a descrição e estatísticas do conjunto de dados, enquanto as métricas de avaliação são descritas na Seção 6.2. A Seção 6.3 e a Seção 6.4

mostram os resultados dos experimentos PJSE e EPHR. Nas seções 6.5 e **??** são apresentados resultados comparativos de RaE e RaC com relação a outras estratégias de particionamento e resto criadas, como DaC e particionamento por vocabulário.

6.1 Conjunto de dados

Nos experimentos foi utilizado o conjunto de dados PAN-11 POTTHAST *et al.* (2011), criado especificamente para a tarefa de identificação de plágio externo, possuindo 26.939 documentos, 61.064 fragmentos de texto classificados como plágio e um total de 686.668.842 palavras, sendo 1.207.741 palavras distintas. Além disso, os documentos no PAN11 são classificados como: fonte (50%) - documentos originais, dos quais o texto é plagiado; falsos positivos (25%) - documentos suspeitos sem nenhum caso de plágio; e verdadeiros positivos (25%) - documentos plagiados que devem ser recuperados.

Os experimentos de PJSE foram conduzidos em 1000 pares de documentos selecionados aleatoriamente, escritos em inglês, e a etapa de *tokenização*, descrito na seção 3.2.1, gerou um vocabulário com 333.443 palavras distintas. No experimento EPHR, foram selecionados dois conjuntos de documentos, os documentos de origem e os documentos suspeitos. Além disso, apenas documentos em inglês com pelo menos um fragmento de plágio foram selecionados, obtendo assim, 15.966 documentos de origem e 4.992 documentos suspeitos, com um vocabulário de 457.092 palavras distintas.

6.2 Métricas de Avaliação

Esta seção mostra as métricas de avaliação para PJSE e EPHR. Antes de explicar cada métrica, vale a pena notar que todos os documentos são conjuntos de *tokens* ou conjuntos de assinaturas. Além disso, as métricas de avaliação dos experimentos do PJSE se baseiam na similaridade de Jaccard entre um determinado par de documentos d_i e d_j, como mostra a equação 6.1.

$$\mathrm{J}(d_i, d_j) = \frac{|d_i \cap d_j|}{|d_i \cup d_j|} \tag{6.1}$$

Quatro matrizes de similaridade Jaccard par a par J, J^{MM}, J^{RaE} e J^{RaC} foram criadas onde $J_{i,j}$ é a similaridade Jaccard entre conjuntos d_i e d_j de *tokens*, enquanto $J^{MM}_{i,j}$, $J^{RaE}_{i,j}$ e $J^{RaC}_{i,j}$ são a similaridade Jaccard entre conjuntos d_i e d_i de assinaturas geradas pelos métodos Minmax, RaE e RaC. Assim, o erro $\mathrm{Error}^{MM}_{i,j}$, $\mathrm{Error}^{RaE}_{i,j}$ e $\mathrm{Error}^{RaC}_{i,j}$ são os erros de similaridade de Jaccard em pares da equação 6.2 para representar, respectivamente, d_i e d_j com os métodos Minmax, RaE e RaC.

$$\text{Error}_{i,j}^{método} = \text{J}_{i,j} - \text{J}_{\text{i,j}}^{\text{método}} \tag{6.2}$$

Nos experimentos do PJSE, como mostra a Equação 6.3, o Erro Médio Absoluto (MAE) é avaliado com relação aos erros de similaridade de Jaccard para todos os pares de documentos d_i e d_j em $(i, j) \in D$. Além disso, a equação 6.4 mostra o erro médio quadrático, conhecido como Mean Squared Error (MSE), uma vez que o valor do MSE é mais suscetível a alterações da variação dos erros de similaridade do Jaccard do que o MAE.

$$\text{MAE}^{método} = \frac{1}{|D|} \sum_{(i,j)\in D} \left| \text{Error}_{i,j}^{método} \right| \tag{6.3}$$

$$\text{MSE}^{método} = \frac{1}{|D|} \sum_{(i,j)\in D} \left(\text{Error}_{i,j}^{método} \right)^2 \tag{6.4}$$

Os experimentos do EPHR foram conduzidos e avaliados de forma similar a DUARTE *et al.* (2017). Na tarefa de indexação, os métodos Minmax (tr^{MM}), RaE (tr^{RaE}) e RaC (tr^{RaC}) foram avaliados com relação a vazão para a indexação dos documentos fontes D_{src}. A equação 6.5 apresenta o cálculo realizado para a avaliação, onde é calculada a razão entre o número de documentos indexados e o somatório do tempo de indexação do conjunto de documentos D_{src}.

$$tr^{método} = \frac{|D_{src}|}{\sum\limits_{d_i \in D_{src}} \Delta_t \text{ para indexar } d_i} \tag{6.5}$$

Para a tarefa de busca do EPHR, a partir de um documento suspeito $d_q \in D_{susp}$ suspeito, deve-se encontrar um conjunto de documentos F_q que foram copiados por d_q e estão disponíveis em D, sendo $F_q \subset D$. Para medir a qualidade desse retorno, o *recall* médio é utilizado, onde é calculada a média, de documentos relevantes retornados com sucesso na busca, entre as consultas D_{susp}. Na equação 6.6, é apresentada a fórmula do *recall* médio, onde F_q são os documentos retornados na busca, e R, os documentos relevantes para aquele documento d_q, listados no conjunto de dados de avaliação.

$$rec = \frac{1}{|D_{susp}|} \sum_{d_q \in D_{susp}} \frac{|F_q \cap R|}{|F_q|} \tag{6.6}$$

O *recall* médio ($rec_{d_q,p}^{MM}$, $rec_{d_q,p}^{RaE}$ e $rec_{d_q,p}^{RaC}$) é avaliado para cada método. Em seguida, o erro absoluto médio do recall, foi avaliado para (MAE_R^{MM}), RaE (MAE_R^{RaE}) e RaC (MAE_R^{RaC}) com relação ao recall do BM25 $rec_{d_q,p}^{BM25}$, como apresentado na equação 6.7. Finalmente, o Speedup, apresentado na equação 6.8, dos métodos RaE e RaC foram mensurados para os experimentos PJSE e EPHR.

$$MAE_R^{método} = \frac{1}{|D_{susp}|} \sum_{d_q \in D_{susp}} (rec_{d_q,p}^{BM25} - rec_{d_q,p}^{método}) \tag{6.7}$$

$$SP^{método} = \text{mean} \left(\frac{\Delta_t \text{ para } \textit{Minmax} \text{ executar a tarefa}}{\Delta_t \text{ para } \textit{método} \text{ executar a tarefa}} \right) \tag{6.8}$$

6.3 Estimativa de similaridade de Jaccard par a par (PJSE)

O primeiro experimento PJSE tem como objetivo principal avaliar como os métodos propostos estimam a similaridade de Jaccard. Os experimentos em PJSE foram avaliados de maneira semelhante à de JI *et al.* (2013) e DUARTE *et al.* (2017), onde o erro do Jaccard entre dois documentos é avaliado medindo o erro médio absoluto (MAE) e o erro médio quadrático (RMSE). A diferença principal entre MAE e RMSE é que o MAE não sofre a influência dos resultados fora da média, onde erros muito grandes não influenciam muito no valor calculado, e o RMSE é influenciado pela variância dos erros, tendo peso maior no calculo. Além da avaliação do erro, é avaliado o tempo necessário para cada método representar e comparar em pares todos os 1000 pares de documentos.

Os métodos avaliados em PJSE foram Minmax, RaE e RaC. Cada método foi avaliado utilizando o conjunto de assinaturas 100, 200, 400 e 800, por documento. Além disso, todas as comparações entre os pares foram repetidas 30 vezes e, em seguida, a média e o desvio padrão de cada métrica foram avaliados e apresentados nas tabelas 6.1, 6.2 e 6.3.

A tabela 6.1 mostra a avaliação do método usado como *baseline*, Minmax. Como esperado, com o aumento na quantidade de assinaturas, a quantidade de tempo aumenta linearmente e, como cada documento é representado com uma quantidade maior de informações, os erros MSE^{MM} e MAE^{MM} diminuem. Além disso, os resultados para RaE e RaC nos experimentos PJSE são apresentados nas tabelas 6.2 e 6.3. De forma semelhante aos resultados do Minmax, os erros MSE^{RaE}, MAE^{RaE}, MSE^{RaC} e MAE^{RaC} diminuem quando a quantidade de assinaturas aumenta.

Tabela 6.1 – Resultados do PJSE para o Minmax. Os valores de erro são multiplicados por 10^{-2}.

k	MSE^{MM}	MAE^{MM}	Δt (seconds)
100	$0,157 \pm 0,244$	$3,07 \pm 2,51$	$225,48 \pm 1,4$
200	$0,077 \pm 0,121$	$2,14 \pm 1,76$	$451,58 \pm 3,43$
400	$0,038 \pm 0,059$	$1,52 \pm 1,24$	$903,88 \pm 7,65$
800	$0,018 \pm 0,028$	$1,04 \pm 0,86$	$1820,42 \pm 15,47$

Nas tabelas 6.2 e 6.3, os resultados de RaC e RaE mostram que à medida que aumentamos a quantidade de assinaturas, ambos os métodos reduzem os erros MAE e MSE, com 2, 4 e 8 células. Apesar de reduzir em quantidade menor, os erros continuam reduzindo, quando particionados com 16 e 32 células. Por outro lado, para a mesma quantidade de assinaturas, o tempo de processamento RaE e RaC diminui drasticamente à medida que aumentamos a quantidade de células.

Tabela 6.2 – MSE, MAE e tempo de comparação relacionado a J x J^{RaE}. Os valores de erro são multiplicados por 10^{-2}.

100 assinaturas

	2 Cells	4 Cells	8 Cells	16 Cells	32 Cells
MSE	$0,17^{+}_{-}$ 0,25	$0,18^{+}_{-}$ 0,27	$0,25^{+}_{-}$ 0,37	$0,58^{+}_{-}$ 0,79	$1,36^{+}_{-}$ 1,52
MAE	$3,16^{+}_{-}$ 2,57	$3,28^{+}_{-}$ 2,66	$3,88^{+}_{-}$ 3,09	$6,11^{+}_{-}$ 4,59	$9,76^{+}_{-}$ 6,37
Δt	$125,35^{+}_{-}$ 0,84	$57,69^{+}_{-}$ 0,33	$26,84^{+}_{-}$ 0,14	$12,28^{+}_{-}$ 0,14	$4,02^{+}_{-}$ 0,01

200 assinaturas

	2 Cells	4 Cells	8 Cells	16 Cells	32 Cells
MSE	$0,08^{+}_{-}$ 0,12	$0,10^{+}_{-}$ 0,15	$0,20^{+}_{-}$ 0,31	$0,50^{+}_{-}$ 0 0,69	$1,25^{+}_{-}$ 1,39
MAE	$2,18^{+}_{-}$ 1,78	$2,48^{+}_{-}$ 2,00	$3,54^{+}_{-}$ 2,81	$5,64^{+}_{-}$ 4,24	$9,44^{+}_{-}$ 5,99
Δt	$255,36^{+}_{-}$ 1,94	$126,03^{+}_{-}$ 1,21	$58,15^{+}_{-}$ 0,50	$27,11^{+}_{-}$ 0,29	$12,42^{+}_{-}$ 0,15

400 assinaturas

	2 Cells	4 Cells	8 Cells	16 Cells	32 Cells
MSE	$0,04^{+}_{-}$ 0,06	$0,06^{+}_{-}$ 0,10	$0,16^{+}_{-}$ 0,26	$0,50^{+}_{-}$ 0,68	$1,23^{+}_{-}$ 1,35
MAE	$1,55^{+}_{-}$ 1,26	$1,92^{+}_{-}$ 1,57	$3,15^{+}_{-}$ 2,51	$5,71^{+}_{-}$ 4,11	$9,41^{+}_{-}$ 5,83
Δt	$512,25^{+}_{-}$ 3,63	$255,54^{+}_{-}$ 1,86	$126,6^{+}_{-}$ 1,54	$58,40^{+}_{-}$ 0,77	$27,17^{+}_{-}$ 0,14

800 assinaturas

	2 Cells	4 Cells	8 Cells	16 Cells	32 Cells
MSE	$0,02^{+}_{-}$ 0,03	$0,05^{+}_{-}$ 0,07	$0,15^{+}_{-}$ 0,24	$0,48^{+}_{-}$ 0,66	$1,24^{+}_{-}$ 1,35
MAE	$1,15^{+}_{-}$ 0,94	$1,66^{+}_{-}$ 1,36	$3,03^{+}_{-}$ 2,39	$5,62^{+}_{-}$ 4,02	$9,53^{+}_{-}$ 5,78
Δt	$1029,16^{+}_{-}$ 7,16	$514,66^{+}_{-}$ 3,29	$257,30^{+}_{-}$ 2,28	$127,12^{+}_{-}$ 1,19	$58,97^{+}_{-}$ 0,56

Tabela 6.3 – MSE, MAE e tempo de comparação relacionado a J x J^{RaC}. Os valores de erro são multiplicados por 10^{-2}.

100 assinaturas

	2 Cells	4 Cells	8 Cells	16 Cells	32 Cells
MSE	$0{,}17^{+}_{-}\ 0{,}25$	$0{,}18^{+}_{-}\ 0{,}27$	$0{,}25^{+}_{-}\ 0{,}37$	$0{,}58^{+}_{-}\ 0{,}79$	$1{,}36^{+}_{-}\ 1{,}51$
MAE	$3{,}16^{+}_{-}\ 2{,}57$	$3{,}28^{+}_{-}\ 2{,}66$	$3{,}88^{+}_{-}\ 3{,}09$	$6{,}11^{+}_{-}\ 4{,}58$	$9{,}74^{+}_{-}\ 6{,}37$
Δt	$125{,}34^{+}_{-}\ 0{,}85$	$57{,}79^{+}_{-}\ 0{,}48$	$26{,}86^{+}_{-}\ 0{,}13$	$12{,}29^{+}_{-}\ 0{,}14$	$4{,}03^{+}_{-}\ 0{,}01$

200 assinaturas

	2 Cells	4 Cells	8 Cells	16 Cells	32 Cells
MSE	$0{,}08^{+}_{-}\ 0{,}12$	$0{,}10^{+}_{-}\ 0{,}15$	$0{,}20^{+}_{-}\ 0{,}31$	$0{,}50^{+}_{-}\ 0{,}69$	$1{,}25^{+}_{-}\ 1{,}39$
MAE	$2{,}18^{+}_{-}\ 1{,}78$	$2{,}48^{+}_{-}\ 2{,}00$	$3{,}54^{+}_{-}\ 2{,}81$	$5{,}63^{+}_{-}\ 4{,}24$	$9{,}42^{+}_{-}\ 5{,}99$
Δt	$255{,}28^{+}_{-}\ 2{,}27$	$126{,}11^{+}_{-}\ 1{,}33$	$58{,}17^{+}_{-}\ 0{,}49$	$27{,}14^{+}_{-}\ 0{,}29$	$12{,}46^{+}_{-}\ 0{,}15$

400 assinaturas

	2 Cells	4 Cells	8 Cells	16 Cells	32 Cells
MSE	$0{,}04^{+}_{-}\ 0{,}06$	$0{,}06^{+}_{-}\ 0{,}10$	$0{,}16^{+}_{-}\ 0{,}26$	$0{,}49^{+}_{-}\ 0{,}67$	$1{,}22^{+}_{-}\ 1{,}35$
MAE	$1{,}55^{+}_{-}\ 1{,}26$	$1{,}92^{+}_{-}\ 1{,}57$	$3{,}15^{+}_{-}\ 2{,}51$	$5{,}70^{+}_{-}\ 4{,}11$	$9{,}40^{+}_{-}\ 5{,}83$
Δt	$511{,}55^{+}_{-}\ 3{,}58$	$255{,}55^{+}_{-}\ 2{,}50$	$126{,}58^{+}_{-}\ 1{,}51$	$58{,}50^{+}_{-}\ 0{,}85$	$27{,}24^{+}_{-}\ 0{,}16$

800 assinaturas

	2 Cells	4 Cells	8 Cells	16 Cells	32 Cells
MSE	$0{,}02^{+}_{-}\ 0{,}03$	$0{,}05^{+}_{-}\ 0{,}07$	$0{,}15^{+}_{-}\ 0{,}24$	$0{,}48^{+}_{-}\ 0{,}66$	$1{,}24^{+}_{-}\ 1{,}35$
MAE	$1{,}15^{+}_{-}\ 0{,}94$	$1{,}66^{+}_{-}\ 1{,}36$	$3{,}03^{+}_{-}\ 2{,}39$	$5{,}61^{+}_{-}\ 4{,}02$	$9{,}51^{+}_{-}\ 5{,}78$
Δt	$1028{,}73^{+}_{-}\ 9{,}40$	$515{,}78^{+}_{-}\ 6{,}18$	$257{,}84^{+}_{-}\ 3{,}00$	$127{,}19^{+}_{-}\ 1{,}16$	$59{,}10^{+}_{-}\ 0{,}58$

O Speedup de ambos os métodos, RaC e RaE, com relação ao Minmax são mostrados na tabela 6.4. O Speedup de menor valor ocorre com 2 células e 200, 400 ou 800 assinaturas, para ambos métodos, sendo mais rápido cerca de 75% que o Minmax. O maior valor apresentado foi executado com 32 células e 100 assinaturas, no RaE, onde é mais rápido cerca de 5508%.

Tabela 6.4 – Speedup de RaE (SP^{RaE}) e RaC (SP^{RaC})

	SP^{RaE}					SP^{RaC}				
k	2	4	8	16	32	2	4	8	16	32
100	01,79	03,90	08,40	18,36	56,08	01,79	03,90	08,39	18,34	55,95
200	01,76	03,58	07,76	16,65	36,35	01,76	03,58	07,76	16,63	36,24
400	01,76	03,53	07,13	15,47	33,26	01,76	03,53	07,14	15,45	33,18
800	01,76	03,53	07,07	14,32	30,87	01,76	03,52	07,06	14,31	30,80

6.4 Busca heurística em plágio externo (EPHR)

Esta subseção apresenta os resultados do experimento de Busca Heurística da tarefa de Plágio Externo (EPHR). Esta etapa tem como objetivo reduzir o número de documentos candidatos para a etapa de Comparação Detalhada do *workflow* de detecção de plágio externo.

Os experimentos EPHR foram realizados contemplando as etapas do *workflow* proposto por DUARTE *et al.* (2017) para o LSH e definido na seção 3.2. Na etapa de *tokenização*, os documentos foram segmentados e representados no modelo *bag-of-words*. Em seguida, as palavras que não ocorrem em mais de um documento foram descartadas, durante as etapas de extração de consulta para a tarefa de Busca e na etapa de extração de *features* na tarefa de Indexação. Isso ocorre porque essas palavras nunca ocorrem em dois documentos e, portanto, não são úteis para a tarefa de busca.

Como apresentado na seção 6.1, a etapa de *tokenização* resulta em um vocabulário com 457.092 palavras. Essas palavras foram representadas por assinaturas, e a partir delas, através do processo de permutação e seleção, selecionados conjuntos de assinaturas para representar cada documento. Cada documento foi representado por 48, 96, 192, 384 e 768 assinaturas e essas armazenadas em um índice invertido orientado a assinatura. Além disso, os métodos BM25 e Minmax foram usados como *baseline* para os experimentos EPHR.

O impacto da partição nos resultados de RaE e RaC foi avaliado usando 2, 4 e 8 células ao gerar 48, 96, 192, 384 e 768 assinaturas (k). Além disso, na tarefa Busca, o tempo de recuperação e consulta foi medido para recuperar 10%, 25%, 50% e 75% do conjunto de documentos de origem, ou seja, recuperação e tempo para recuperar até 1597, 3991, 7983 e 11975 documentos. Os resultados da tarefa de Indexação pode ser vista na seção 6.4.1, e da tarefa de Busca pelo documento fonte de plágio, na seção 6.4.2.

6.4.1 Tarefa de indexação

A figura 6.1 apresenta os valores de vazão para os métodos Minmax, RaE e RaC na etapa de indexação dos documentos fontes. Como pode ser visto, quando comparado ao Minmax, RaE e RaC apresentam valores superiores com relação a quantidade de documentos por hora indexados. Um exemplo disso é a indexação utilizando duas células, que compreende, aproximadamente, valores de vazão 101%, 96%, 96%, 97% e 99% maiores, ao gerar e indexar 48, 96, 192, 384 e 768 assinaturas, respectivamente. Além disso, a vazão de RaC e RaE aumenta quando o número de células aumenta, assim ambos os métodos podem indexar documentos mais rápidos

com mais células, podendo atingir valores 693% maiores que o Minmax quando utilizado o particionamento por oito células.

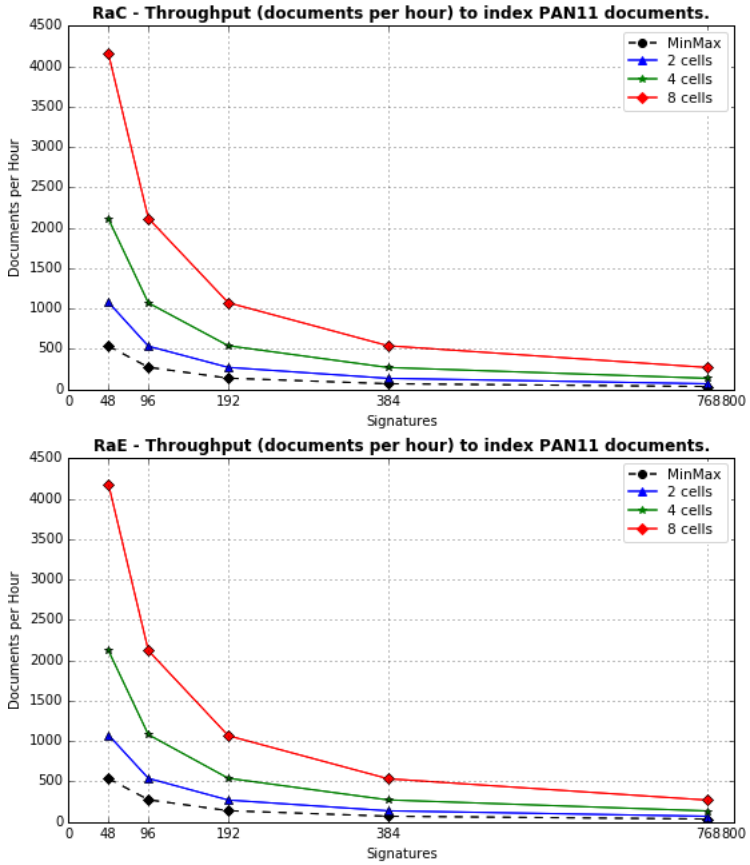


Figura 6.1 – Vazão (documentos por hora) de RaC e RaE para indexar documentos do PAN11.

A tabela 6.5 é uma comparação entre os valores de vazão resultantes da execução dos métodos RaE, RaC, CSA e CSA_L. Para tal, RaE e RaC são representados pelos valores resultantes do uso de 2 e 8 células. Essa representação tem com objetivo comparar os extremos do número de células experimentados, levando em conta que, como apresentado na figura 6.1, particionar com 2 células retorna valores inferiores e com 8 células, superiores. A versão completa da tabela, incluindo valores com 4 células, está disponível no apêndice B.1.

Com base na tabela 6.5 pode ser visto que RaE e RaC com 2 células tem vazão pelo menos 30% maior que CSA_L. Quando utilizadas as 8 células, RaE e RaC tem vazão pelo menos 293% e 411% maior que CSA e CSA_L. No entanto, nos resultados da tarefa de indexação, o BM25 teve o maior rendimento com relação a RaE, RaC, CSA e CSA_L, indexando aproximadamente 101.110 documentos por hora.

Tabela 6.5 – Vazão (documentos por hora) de RaC e RaE para indexar documentos do PAN-11. Resultados de CSA e CSA_L foram extraídos de (DUARTE *et al.*, 2017)

k	RAC$^{2\text{ cells}}$	RAE$^{2\text{ cells}}$	RAC$^{8\text{ cells}}$	RAE$^{8\text{ cells}}$	CSA_L	CSA
48	1077.52	1075.35	4154.46	4183.40	811.44	1055.95
96	533.67	537.70	2117.17	2119.61	409.93	533.62
192	270.90	268.99	1071.70	1066.02	204.66	269.55
384	134.96	135.58	538.57	532.64	102.59	135.21
768	68.08	67.69	269.32	267.78	51.16	67.77

6.4.2 Tarefa de busca por documentos fonte de plágio

A tarefa de busca por documentos fonte de plágio pode responder as três perguntas a seguir: ($Q1$) quão rápido o RaE e o RaC pré-processam e extraem uma consulta de um documento suspeito? ($Q2$) RaE e RaC são mais rápidos que os *baselines* na tarefa de Busca? ($Q3$) RaE e RaC são melhores que os *baselines* para recuperar os documentos originais de um documento plagiado? Nas seções seguintes são apresentados os resultados para $Q1$, $Q2$ e $Q3$.

6.4.2.1 ($Q1$) Extração de consultas

O tempo de extração da consulta foi avaliado ao medir o tempo de pré-processamento e de se extrair uma consulta para a tarefa Busca. A tabela 6.6 apresenta o tempo de extração de consulta do Minmax e o Speedup dos métodos RaC, RaE, CSA_L e CSA com relação ao próprio Minmax. RaE e RaC, quando avaliados com 2 células, são aproximadamente 2 vezes mais rápidos que o Minmax e pelo menos 30% mais rápido que o CSA_L, entretanto, apresentam mesmo Speedup do CSA. No entanto, quando avaliados com 4 e 8 células, RaE e RaC são 2 e 4 vezes mais rápidos que o CSA. Além disso, RaE e RaC apresentam aumento de seus Speedup proporcionais ao número de células, onde ao se dobrar o número de células, dobra ou se aproxima de se dobrar o Speedup.

Tabela 6.6 – Tempo médio de extração de consulta do Minmax (em segundo) e o Speedup de SP^{RaE}, SP^{RaC}, SP^{CSA_L} e SP^{CSA}

k	Δt MinMax	SP^{RaC}			SP^{RaE}			SP^{CSA_L}	SP^{CSA}
		2	4	8	2	4	8		
48	6.64	1.99	3.92	7.74	1.98	3.95	7.74	1.50	1.96
96	13.26	1.99	3.93	7.82	1.98	3.92	7.80	1.50	1.96
192	26.46	1.99	3.95	7.86	1.98	3.94	7.84	1.50	1.97
384	53.35	2.00	3.99	7.97	1.99	4.00	7.95	1.52	1.99
768	105.68	1.99	3.97	7.87	1.98	3.98	7.91	1.50	1.98

6.4.2.2 ($Q2$) Busca

As figuras 6.2 e 6.3 mostram a comparação do tempo total de busca para Minmax, RaC e RaE ao selecionar 1597 e 11975 documentos, respectivamente. Os resultados para 3991 e 7983 não são apresentados, pois possuem comportamento semelhante, podendo serem vistos nas figuras B.1 e B.2 do apêndice. Como pode ser notado, RaC e RaE com 2 células são duas vezes mais rápidos que o Minmax, quatro vezes mais rápidos com 4 células e oito vezes mais rápidos com 8 células. Esse comportamento se mostra consistente, mesmo com alteração das quantidades de documentos selecionados ou de assinaturas selecionadas.

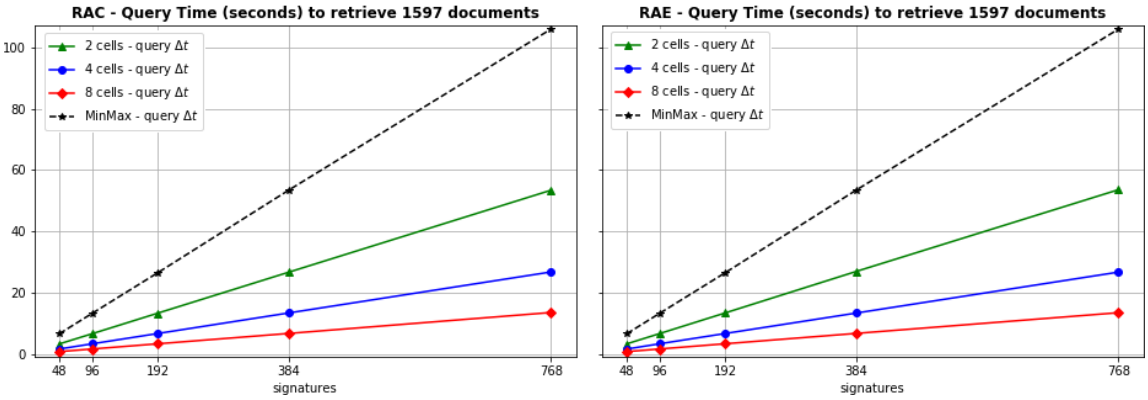


Figura 6.2 – Tempo total de busca para RaC, RaE e Minmax para selecionar 1597 documentos.

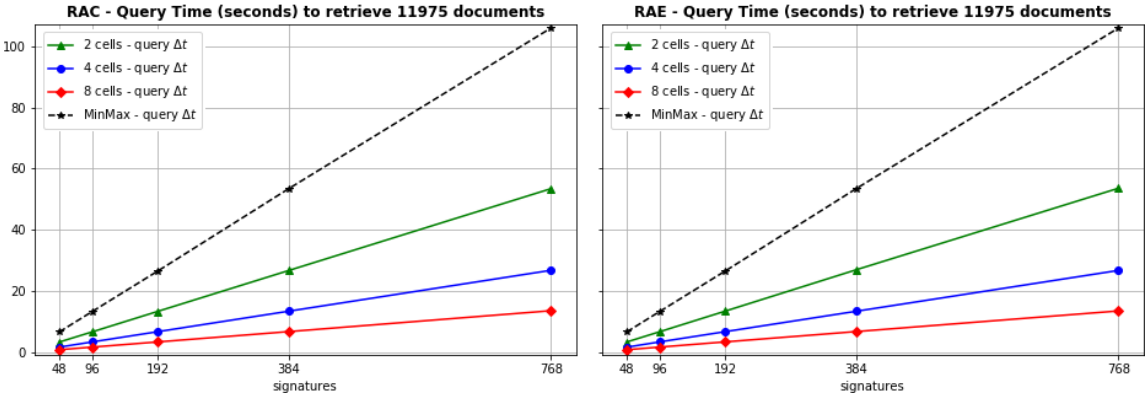


Figura 6.3 – Tempo total de busca para RaC, RaE e Minmax para selecionar 11975 documentos.

As figuras 6.4 e 6.5 mostram o Speedup do Minmax, RaC e RaE na tarefa de busca com relação ao tempo usado pelo BM25, ao selecionar 1597 e 11975 documentos, respectivamente. Os resultados para 3991 e 7983 não são apresentados, pois possuem comportamento semelhante, podendo serem vistos nas figuras B.7 e B.8 do apêndice. Como esperado, independentemente do número de documentos recuperados, o Speedup de RaE, RaC e Minmax diminuem com relação ao aumento do número de assinaturas por documento. Contudo, RaE e RaC, com 4 e 8 células, foram pelo menos 12% mais rápidos que o Minmax.

A figura 6.4 mostrou que RaE e RaC tiveram um Speedup melhor com 8 células para gerar 48 e 96 assinaturas e com 4 células para gerar 48 assinaturas. A partir disso, pode ser visto que RaE e RaC são 75% mais rápidos com relação ao BM25 ao utilizar 8 células para gerar 48 assinaturas.

A figura 6.5 mostrou que os resultados de Speedup para RaE e RaC melhoraram quando houve um aumento no número de documentos recuperados. De fato, tanto o Speedup com 8 células para gerar 96 assinaturas, quanto com 4 células para gerar 48 assinaturas, tiveram uma pequena melhora de Speedup, enquanto ambos os métodos com 8 células para gerar 48 assinaturas foram 100% mais rápidos que o BM25.

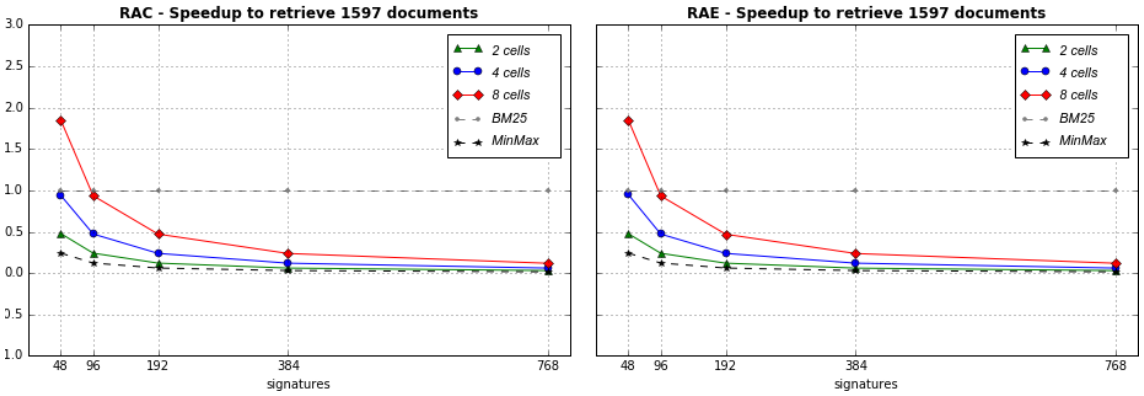


Figura 6.4 – Speedup de RaC, RaE, Minmax com relação ao tempo de busca do BM25 para 1597 documentos.

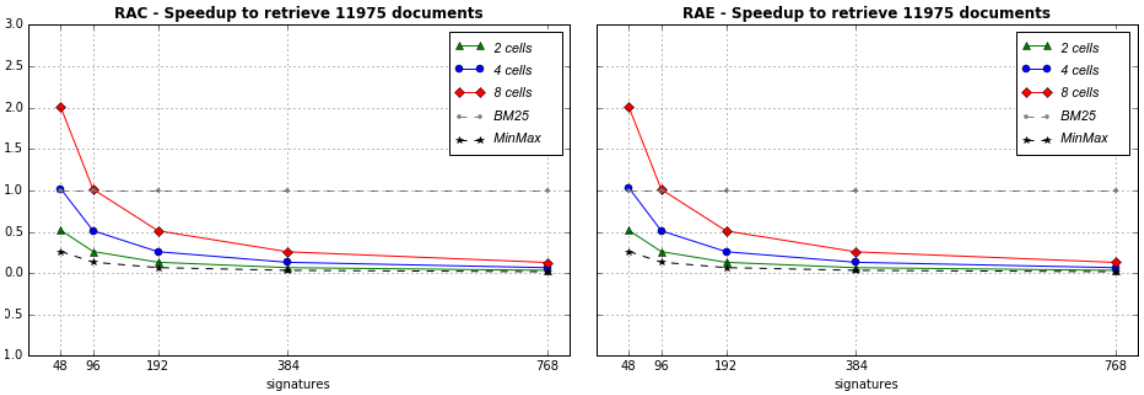


Figura 6.5 – Speedup de RaC, RaE, Minmax com relação ao tempo de busca do BM25 para 11975 documentos.

6.4.2.3 ($Q3$) Recall

O Speedup para recuperar 1597, 3391, 7983 e 11975 documentos foi avaliado para o Minmax, RaE e RaC, sendo apresentado na seção 6.4.2.2. A partir destes resultados, foram selecionadas as configurações de RaE e RaC que obtiveram melhor desempenho de Speedup com relação ao BM25, para uma melhor comparação, levando em consideração o erro MAE do *recall* e o próprio Speedup. Esses valores para comparação são apresentados na tabela 6.7, que além de dispor dos valores de RaE e RaC, exibe valores de Minmax, CSA e CSA_L.

Como esperado, com o aumento do número de assinaturas, o *recall* tem uma melhora em todas as abordagens $MAE_R^{CSA_L}$, MAE_R^{CSA}, MAE_R^{MM}, MAE_R^{RaE} e MAE_R^{RaC}, como pode ser visto entre as tabelas para 48 assinaturas e 96 assinaturas. Outro fato que pode ser notado é que, na maioria dos casos da tarefa de busca por fontes de plágio, RaE e RaC tiveram um *recall* melhor do que CSA_L e CSA. Contudo, apresentou *recall* pior que o BM25 e ligeiramente pior que o método Minmax.

Nas tabelas B.3 e B.3 do apêndice podem ser vistos todos os valores de MAE das técnicas RaE e RaC com relação a Minmax e BM25. É notável que, RaC e RaE apresentaram valores de *recall* piores que o BM25, entretanto, o erro ligeiramente maior é compensado pelos bons ganhos com relação a tempo de busca. Além disso, para ambos, Minmax e BM25, o MAE de RaC e RaE diminui quando o número de assinaturas é aumentado, levando RaC e RaE a terem *recall* superior ao do Minmax em alguns casos.

Tabela 6.7 – MAE do Recall e Speedup para Minmax, CSA, CSA_L, RaC e RaE com 48 e 96 assinaturas em relação ao BM25.

Resultados para recuperar p documentos com 48 assinaturas														
p	Minmax		CSA_L		CSA		RAC$^{4\ cells}$		RAE$^{4\ cells}$		RAC$^{8\ cells}$		RAE$^{8\ cells}$	
	MAE_R^{MM}	Sp[1]	$MAE_R^{CSA_L}$	Sp^a	MAE_R^{CSA}	Sp^a	MAE_R^{RaC}	Sp^a	MAE_R^{RaE}	Sp^a	MAE_R^{RaC}	Sp^a	MAE_R^{RaE}	Sp^a
1597	**0.11**	0.24	0.14	0.36	0.15	0.47	0.12	0.94	0.12	0.95	0.12	**1.85**	0.12	**1.85**
3991	**0.09**	0.24	0.12	0.36	0.13	0.46	0.11	0.92	0.11	0.93	0.10	**1.81**	0.10	**1.81**
7983	**0.17**	0.23	0.19	0.35	0.21	0.45	0.19	0.90	0.19	0.91	0.18	**1.77**	0.17	**1.77**
11975	**0.10**	0.26	0.12	0.39	0.13	0.51	0.11	**1.02**	0.11	**1.03**	0.11	**2.01**	0.11	**2.01**
Resultados para recuperar p documentos com 96 assinaturas														
p	Minmax		CSA_L		CSA		RAC$^{4\ cells}$		RAE$^{4\ cells}$		RAC$^{8\ cells}$		RAE$^{8\ cells}$	
	MAE_R^{MM}	Sp^a	$MAE_R^{CSA_L}$	Sp^a	MAE_R^{CSA}	Sp^a	MAE_R^{RaC}	Sp^a	MAE_R^{RaE}	Sp^a	MAE_R^{RaC}	Sp^a	MAE_R^{RaE}	Sp^a
1597	**0.08**	0.12	0.11	0.18	0.13	0.24	0.09	0.47	0.09	0.47	0.10	0.93	0.10	0.93
3991	**0.06**	0.12	0.08	0.18	0.11	0.23	0.08	0.46	0.08	0.46	0.08	0.91	0.08	0.91
7983	**0.15**	0.12	0.15	0.17	0.18	0.23	0.17	0.45	0.17	0.45	0.16	0.89	0.16	0.89
11975	**0.09**	0.13	0.09	0.20	0.10	0.26	0.09	0.51	0.10	0.51	0.10	**1.01**	0.10	**1.01**

6.5 RaC e RaE x DaC

Esta seção tem como objetivo apresentar os resultados dos experimentos PJSE e EPHR para a estratégia de tratamento de restos DaC aplicada ao particionamento das permutação. Além disso, é vista uma comparação entre o DaC e as estratégias RaE e RaC, analisadas na seção 6.3 e 6.4.

6.5.1 PJSE

Os experimentos PJSE, que tem como objetivo principal avaliar como os métodos propostos estimam a similaridade de Jaccard, foram executados para DaC de forma similar ao apresentado na seção 6.3 para Minmax, CSA e CSA_L. Os resultados com relação a erro e a tempo foram apresentados na tabela 6.8, onde é visto que os erros MSE e MAE se alteram muito pouco ao se aumentar o número de assinaturas, além de apresentar aumento dos erros quando se aumenta o número de células. Além disso, se comparado com valores de RaE e RaC, apresentados nas tabelas 6.2 e 6.2, o método DaC apresenta desempenho inferior com relação a MSE, MAE e tempo.

A tabela 6.9 apresenta valores de Speedup do DaC com relação ao Minmax. DaC apresenta bons valores de Speedup, onde em seu menor valor, com 2 células e 200, 400 ou 800 assinaturas, é 74% maior que o Minmax. Além disso, quando analisado seu maior valor apresentado, DaC é superior em 5445%. Contudo, se comparado com valores de RaE e RaC, apresentados na tabela 6.4, a estratégia DaC possui desempenho ligeiramente inferior.

Tabela 6.8 – MSE, MAE e tempo de comparação relacionado a J x J^{DaC}. Os valores de erro são multiplicados por 10^{-2}.

100 assinaturas

	2 Cells	4 Cells	8 Cells	16 Cells	32 Cells
MSE	1.50^{+}_{-} 1.27	2.88^{+}_{-} 2.10	3.70^{+}_{-} 2.59	4.12^{+}_{-} 2.81	4.35^{+}_{-} 2.94
MAE	$10{,}91^{+}_{-}$ 5.63	15.42^{+}_{-} 7.11	17.57^{+}_{-} 7.86	18.58^{+}_{-} 8.17	19.10^{+}_{-} 8.36
Δt	128.52^{+}_{-} 0.86	58.58^{+}_{-} 0.33	27.05^{+}_{-} 0.14	12.35^{+}_{-} 0.13	4.07^{+}_{-} 0.042

200 assinaturas

	2 Cells	4 Cells	8 Cells	16 Cells	32 Cells
MSE	1.45^{+}_{-} 1.13	2.88^{+}_{-} 2.04	3.70^{+}_{-} 2.55	4.12^{+}_{-} 2.80	4.34^{+}_{-} 2.92
MAE	10.83^{+}_{-} 5.27	15.46^{+}_{-} 7.00	17.58^{+}_{-} 7.79	18.60^{+}_{-} 8.15	19.09^{+}_{-} 8.32
Δt	259.98^{+}_{-} 1.94	128.85^{+}_{-} 1.18	58.70^{+}_{-} 0.51	27.18^{+}_{-} 0.30	12.44^{+}_{-} 0.18

400 assinaturas

	2 Cells	4 Cells	8 Cells	16 Cells	32 Cells

MSE	1.43^{+}_{-} 1.06	2.86^{+}_{-} 1.99	3.69^{+}_{-} 2.52	4.13^{+}_{-} 2.79	4.34^{+}_{-} 2.92
MAE	10.80^{+}_{-} 5.10	15.42^{+}_{-} 6.92	17.58^{+}_{-} 7.74	18.61^{+}_{-} 8.14	19.10^{+}_{-} 8.32
Δt	519.71^{+}_{-} 3.78	261.89^{+}_{-} 1.79	129.25^{+}_{-} 1.86	58.94^{+}_{-} 1.09	27.21^{+}_{-} 0.28
800 assinaturas					
	2 Cells	4 Cells	8 Cells	16 Cells	32 Cells
MSE	1.41^{+}_{-} 1.03	2.86^{+}_{-} 1.98	3.69^{+}_{-} 2.51	4.12^{+}_{-} 2.78	4.34^{+}_{-} 2.91
MAE	10.79^{+}_{-} 5.01	15.44^{+}_{-} 6.90	17.58^{+}_{-} 7.74	18.60^{+}_{-} 8.13	19.10^{+}_{-} 8.32
Δt	1044.46^{+}_{-} 10.13	527.79^{+}_{-} 5.47	264.75^{+}_{-} 1.96	130.17^{+}_{-} 1.14	59.60^{+}_{-} 0.55

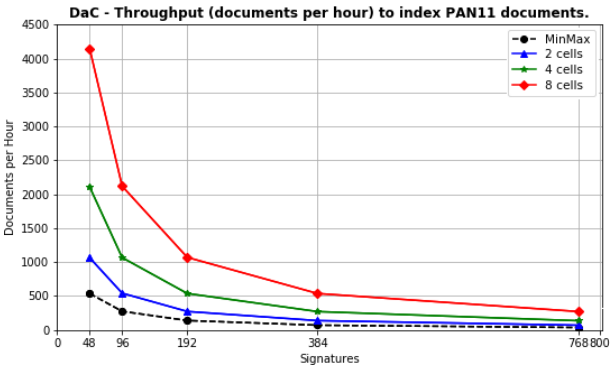
Tabela 6.9 – DaC (SP^{DaC}) Speedup

k	SP^{DaC}				
	2	4	8	16	32
100	1.75	3.85	8.33	18.26	55.45
200	1.74	3.50	7.69	16.62	36.31
400	1.74	3.45	6.99	15.33	33.22
800	1.74	3.45	6.88	13.98	30.54

6.5.2 EPHR

Para o DaC, foi realizado o experimento EPHR nos mesmos moldes apresentados na seção 6.4, onde o experimento é realizado em duas etapas: indexação e busca. A etapa de indexação apresentou valores de vazão, que podem ser vistos na tabela e figura 6.6, onde apresenta valores superiores ao Minmax. Ao indexar com 2 células, DaC apresenta valores 98%, 98%, 97%, 98% e 99% maiores que o Minmax, ao gerar e indexar 48, 96, 192, 384, 768 assinaturas, respectivamente. Além disso, com o aumento das células pode chegar a um valor 693% maior que o Minmax com 8 células. A partir dos resultados obtidos pode ser visto que o DaC tem valores de vazão semelhantes aos de RaE e RaC, apresentados na seção 6.4.1.

Na segunda etapa, após a indexação dos documentos fonte, é realizada a tarefa de busca pela fonte de plágio. A busca é dividida em três questões: tempo de extração de consulta ($Q1$), tempo de busca ($Q2$) e avaliação do *recall* ($Q3$). Para a etapa de extração de consulta ($Q1$) foi criada a tabela 6.10, onde são apresentados os valores de tempo para extração de consulta do Minmax e o Speedup do DaC com relação ao próprio Minmax. DaC possui desempenho ligeiramente inferior quando comparado a valores de Speedup de RaE e RaC com relação ao Minmax, apresentados na seção 6.4.2.1.



k	DAC$^{2\ \text{cells}}$	DAC$^{4\ \text{cells}}$	DAC$^{8\ \text{cells}}$
48	1064.01	2109.71	4150.61
96	540.26	1065.14	2118.70
192	270.70	536.11	1068.79
384	135.76	269.14	535.64
768	67.94	134.17	269.86

Figura 6.6 – Vazão (documentos por hora) de DaC para indexar documentos do PAN-11.

Tabela 6.10 – Tempo médio de extração de consulta do Minmax (em segundo) e o Speedup de SP^{DaC}

k	Δt MinMax	SP^{DaC}		
		2	4	8
48	6.64	1.97	3.91	7.71
96	13.26	1.97	3.91	7.76
192	26.46	1.97	3.93	7.81
384	53.35	2.00	3.98	7.96
768	105.68	1.99	3.97	7.93

Para a questão de tempo de busca ($Q2$), a figura 6.7 mostra a comparação do tempo total de busca para Minmax e DaC ao selecionar 1597 e 11975 documentos. Os resultados para 3991 e 7983 não são apresentados, pois possuem comportamento semelhante, podendo serem vistos na figura B.3 do apêndice. Como pode ser notado, DaC apresenta comportamento semelhante a RaC e RaE, apresentado na seção 6.4.2.2, onde com 2 células apresenta busca duas vezes mais rápidas que o Minmax, quatro vezes mais rápidas com 4 células e oito vezes mais rápidas com 8 células. Esse comportamento se mostra consistente, mesmo com alteração das quantidades de documentos selecionados ou de assinaturas selecionadas.

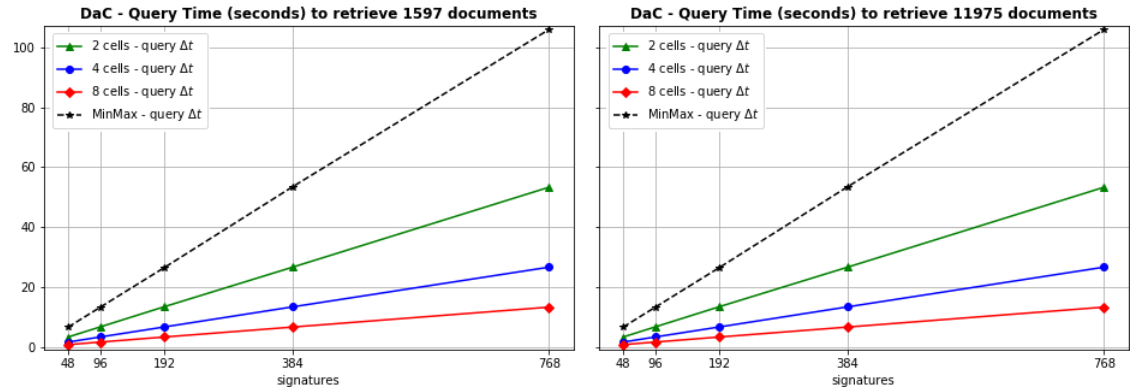


Figura 6.7 – Tempo total de busca para DaC selecionar 1597 e 11975 documentos.

A figura 6.8 mostra o Speedup do Minmax e DaC na tarefa de busca com relação ao tempo usado pelo BM25, ao selecionar 1597 e 11975 documentos. Os resultados para 3991 e 7983 não são apresentados, pois possuem comportamento semelhante, podendo serem vistos na figura B.9 do apêndice. Como pode ser notado, DaC apresenta comportamento semelhante a RaC e RaE, apresentado na seção 6.4.2.2, onde teve um Speedup melhor que o BM25 com 8 células para gerar 48 e 96 assinaturas e com 4 células para gerar 48 assinaturas.

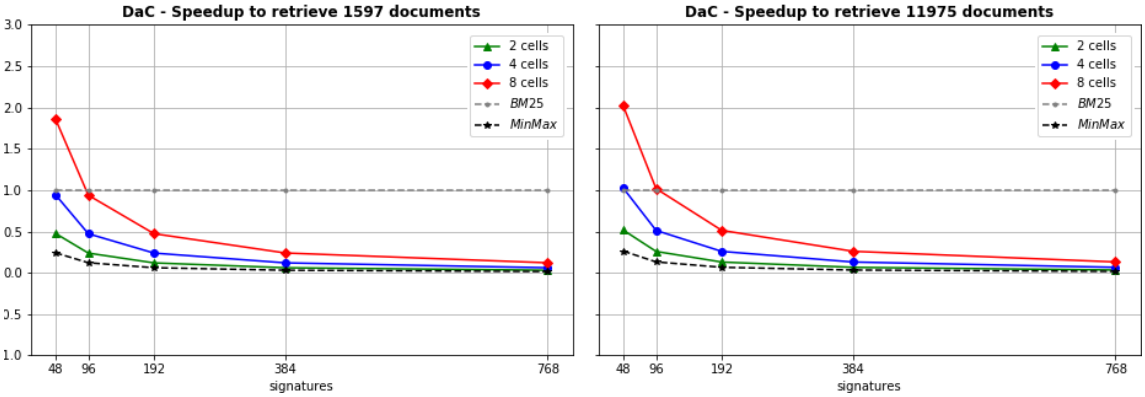


Figura 6.8 – Speedup de DaC com relação ao tempo de busca do BM25 para 1597 e 11975 documentos.

Para a questão de análise de *recall* ($Q3$) foram selecionadas as configurações de DaC que obtiveram melhor desempenho de Speedup com relação ao BM25. Como pode ser notado, as configurações selecionadas foram as mesmas de RaE e RaC. Assim sendo, para uma melhor comparação, a tabela 6.11 foi criada, apresentando valores de MAE e Speedup para Minmax, RaC, RaE e DaC.

Como esperado, com o aumento do número de assinaturas, o *recall* tem uma melhora no MAE_R^{DaC} para DaC em comparação ao BM25, levando a valores semelhantes ao Minmax, quando utilizada 8 células. Além disso, com 8 células, DaC tem valores de MAE_R^{DaC} inferiores aos de RaE e RaC, tendo esse comportamento acontecendo com todos valores de assinaturas experimentados (48, 96, 192, 384, 768) e

todos valores de seleção de documentos (1597, 3991, 7983, 11975). Em contrapartida, para execuções com 2 e 4 células, DaC tem valores de MAE_R^{DaC} superiores a RaE e RaC. Nas tabelas B.5 e B.5 do apêndice podem ser vistos todos os valores de MAE da técnica DaC com relação a Minmax e BM25.

Tabela 6.11 – MAE do Recall e Speedup para Minmax, RaC, RaE e DaC com 48 e 96 assinaturas em relação ao BM25.

results to retrieve p documents with 48 fingerprints														
p	Minmax		RAC$^{4\ cells}$		RAE$^{4\ cells}$		RAC$^{8\ cells}$		RAE$^{8\ cells}$		DAC$^{4\ cells}$		DAC$^{8\ cells}$	
	MAE_R^{MM}	Sp^a	MAE_R^{RaC}	Sp^a	MAE_R^{RaE}	Sp^a	MAE_R^{RaC}	Sp^a	MAE_R^{RaE}	Sp^a	MAE_R^{DaC}	Sp^a	MAE_R^{DaC}	Sp^a
1597	**0.11**	0.24	0.12	0.94	0.12	0.95	0.12	1.85	0.12	1.85	0.23	0.94	**0.11**	**1.86**
3991	**0.09**	0.24	0.11	0.92	0.11	0.93	0.10	1.81	0.10	1.81	0.21	0.87	**0.09**	**1.82**
7983	**0.17**	0.19	0.19	0.90	0.19	0.91	0.18	1.77	0.17	1.77	0.34	0.90	**0.17**	**1.78**
11975	**0.10**	0.26	0.11	1.02	0.11	1.03	0.11	2.01	0.11	2.01	0.27	1.02	0.11	**2.02**
results to retrieve p documents with 96 fingerprints														
p	Minmax		RAC$^{4\ cells}$		RAE$^{4\ cells}$		RAC$^{8\ cells}$		RAE$^{8\ cells}$		DAC$^{4\ cells}$		DAC$^{8\ cells}$	
	MAE_R^{MM}	Sp^a	MAE_R^{RaC}	Sp^a	MAE_R^{RaE}	Sp^a	MAE_R^{RaC}	Sp^a	MAE_R^{RaE}	Sp^a	MAE_R^{DaC}	Sp^a	MAE_R^{DaC}	Sp^a
1597	**0.08**	0.12	0.09	0.47	0.09	0.47	0.10	0.93	0.10	0.93	0.21	0.47	**0.08**	**0.94**
3991	**0.06**	0.12	0.08	0.46	0.08	0.46	0.08	0.91	0.08	0.91	0.18	0.46	**0.06**	**0.92**
7983	**0.15**	0.12	0.17	0.45	0.17	0.45	0.16	0.89	0.16	0.89	0.28	0.45	**0.15**	**0.90**
11975	**0.09**	0.13	0.09	0.51	0.10	0.51	0.10	1.01	0.10	1.01	0.20	0.51	**0.09**	**1.02**

6.6 Particionando as permutações x Particionando o vocabulário

Esta seção tem como objetivo apresentar os resultados dos experimentos PJSE e EPHR para a estratégia de particionamento do vocabulário (VP). As estratégias de tratamento de restos RaE, RaC e DaC foram aplicadas da mesma maneira que em estratégias de particionamento por permutação. Além disso, é vista uma comparação entre ambas estratégias de particionamento, separadas por estratégia de tratamento de resto. Nesta seção, as estratégias de resto aplicadas ao particionamento do vocabulário serão nomeadas com o prefixo VP (VP^{metodo}) e as de particionamento das permutações permanecerão com os nomes dados nas seções anteriores. As estratégias de resto aplicadas ao particionamento das permutações foram apresentadas nas seções 6.3, 6.4 e 6.5.

6.6.1 PJSE

Os experimentos PJSE foram executados para o particionamento do vocabulário (VP) da mesma maneira que para o particionamento das permutações. Foram avaliados os erros MSE e MAE com relação a estimativa de Jaccard e seu tempo de execução. Nas subseções seguintes são apresentados os valores resultantes do experimento e uma comparação entre a estratégia de resto quando aplicada ao particionamento do vocabulário e das permutações.

6.6.1.1 RaC x VP^{RaC}

A tabela 6.12 apresenta os resultados de MSE, MAE e do tempo de execução para a estratégia de particionamento do vocabulário VP^{RaC}. Como esperado, os erros MAE e MSE aumentam ao se aumentar o número de células e diminui ao se aumentar o número de assinaturas. Além disso, um outro comportamento esperado que ocorre é o de se aumentar o tempo quando o número de assinaturas aumenta e de diminuir o tempo quando o número de células aumenta. Quando VP^{RaC} é comparado ao RaC, apresenta valores de MAE e MSE, na maioria das vezes, piores e valores de tempo pelo menos duas vezes maiores. Os valores de MAE, MSE e tempo de RaC podem ser vistos na tabela 6.3.

Tabela 6.12 – MSE, MAE e tempo de comparação relacionado a J x $J^{\mathrm{VP^{RaC}}}$. Os valores de erro são multiplicados por 10^{-2}.

100 assinaturas

	2 Cells	4 Cells	8 Cells	16 Cells	32 Cells

MSE	$0{,}14^{+}_{-}$ 0,21	$0{,}32^{+}_{-}$ 0,49	$0{,}59^{+}_{-}$ 0,89	$1{,}18^{+}_{-}$ 1,80	$3{,}37^{+}_{-}$ 4,83
MAE	$2{,}91^{+}_{-}$ 2,37	$4{,}40^{+}_{-}$ 3,57	$6{,}00^{+}_{-}$ 4,82	$8{,}49^{+}_{-}$ 6,76	$14{,}5^{+}_{-}$ 11,23
Δt	$482{,}00^{+}_{-}$ 5,00	$166{,}26^{+}_{-}$ 0,95	$66{,}63^{+}_{-}$ 0,26	$28{,}84^{+}_{-}$ 0,18	$8{,}92^{+}_{-}$ 0,04

200 assinaturas

	2 Cells	4 Cells	8 Cells	16 Cells	32 Cells
MSE	$0{,}07^{+}_{-}$ 0,118	$0{,}14^{+}_{-}$ 0,21	$0{,}30^{+}_{-}$ 0,45	$0{,}63^{+}_{-}$ 0,94	$1{,}33^{+}_{-}$ 1,75
MAE	$2{,}07^{+}_{-}$ 1,68	$2{,}94^{+}_{-}$ 2,35	$4{,}26^{+}_{-}$ 3,42	$6{,}23^{+}_{-}$ 4,96	$9{,}26^{+}_{-}$ 6,88
Δt	$947{,}07^{+}_{-}$ 13,79	$346{,}58^{+}_{-}$ 4,33	$136{,}07^{+}_{-}$ 1,60	$59{,}62^{+}_{-}$ 1,16	$26{,}99^{+}_{-}$ 0,47

400 assinaturas

	2 Cells	4 Cells	8 Cells	16 Cells	32 Cells
MSE	$0{,}04^{+}_{-}$ 0,06	$0{,}08^{+}_{-}$ 0,12	$0{,}15^{+}_{-}$ 0,23	$0{,}34^{+}_{-}$ 0,50	$0{,}86^{+}_{-}$ 1,18
MAE	$1{,}53^{+}_{-}$ 1,27	$2{,}17^{+}_{-}$ 1,76	$3{,}03^{+}_{-}$ 2,47	$4{,}59^{+}_{-}$ 3,66	$7{,}38^{+}_{-}$ 5,62
Δt	$1986{,}73^{+}_{-}$ 14,57	$723{,}50^{+}_{-}$ 3,98	$297{,}62^{+}_{-}$ 1,45	$126{,}37^{+}_{-}$ 0,67	$57{,}47^{+}_{-}$ 0,29

800 assinaturas

	2 Cells	4 Cells	8 Cells	16 Cells	32 Cells
MSE	$0{,}02^{+}_{-}$ 0,03	$0{,}05^{+}_{-}$ 0,07	$0{,}10^{+}_{-}$ 0,15	$0{,}19^{+}_{-}$ 0,29	$0{,}50^{+}_{-}$ 0,69
MAE	$1{,}07^{+}_{-}$ 0,88	$1{,}68^{+}_{-}$ 1,35	$2{,}41^{+}_{-}$ 1,96	$3{,}43^{+}_{-}$ 2,78	$5{,}63^{+}_{-}$ 4,31
Δt	$3840{,}57^{+}_{-}$ 33,52	$1415{,}49^{+}_{-}$ 7,98	$591{,}75^{+}_{-}$ 2,89	$267{,}41^{+}_{-}$ 1,55	$119{,}10^{+}_{-}$ 0,42

A tabela 6.13 apresenta valores de Speedup para VP^{RaC} e RaC com relação ao Minmax. Como pode ser visto, VP^{RaC} apresenta valores superiores ao Minmax, podendo ser até 25 vezes mais rápido quando executado com 32 células e 100 assinaturas. Em contrapartida, quando comparado com RaC, apresenta valores muito inferiores, podendo chegar a ser 3,8 vezes mais lento, com 2 células e 100 assinaturas.

Tabela 6.13 – VP^{RaC} ($SP^{VP^{RaC}}$) e RaC (SP^{RaC}) Speedup

k	$SP^{VP^{RaC}}$					RaC				
	2	4	8	16	32	2	4	8	16	32
100	0,47	1,36	3,38	7,82	25,27	1,79	3,90	8,39	18,34	55,95
200	0,48	1,30	3,32	7,57	16,73	1,76	3,58	7,76	16,63	36,24
400	0,45	1,25	3,04	7,15	15,73	1,76	3,53	7,14	15,45	33,18
800	0,47	1,29	3,08	6,81	15,28	1,76	3,52	7,06	14,31	30,80

6.6.1.2 RaE x VP^{RaE}

Na tabela 6.14 são apresentados os valores de MSE, MAE e do tempo de execução para VP^{RaE}, onde pode ser visto que MSE e MAE se alteram muito pouco quando aumentado o número de células. Além disso, algumas características esperadas acontecem, como: os valores de MAE e MSE diminuem quando o número de assinaturas aumenta, o tempo aumenta quando o número de assinaturas aumenta e

o tempo diminui quando o número de células aumenta. Quando VP^{RaE} é comparado ao RaE, apresenta valores de MAE e MSE, na maioria das vezes, melhores e valores de tempo pelo menos três vezes maiores. Os valores de MAE, MSE e tempo de RaE são apresentados na tabela 6.2.

Tabela 6.14 – MSE, MAE e tempo de comparação relacionado a J x $J^{\mathrm{VP^{RaE}}}$. Os valores de erro são multiplicados por 10^{-2}.

100 assinaturas

	2 Cells	4 Cells	8 Cells	16 Cells	32 Cells
MSE	$0{,}14^{+}_{-}\ 0{,}21$	$0{,}16^{+}_{-}\ 0{,}24$	$0{,}15^{+}_{-}\ 0{,}24$	$0{,}16^{+}_{-}\ 0{,}24$	$0{,}22^{+}_{-}\ 0{,}34$
MAE	$2{,}91^{+}_{-}\ 2{,}37$	$3{,}07^{+}_{-}\ 2{,}50$	$3{,}03^{+}_{-}\ 2{,}48$	$3{,}05^{+}_{-}\ 2{,}50$	$3{,}67^{+}_{-}\ 2{,}99$
Δt	$467{,}06^{+}_{-}\ 3{,}54$	$276{,}87^{+}_{-}\ 2{,}22$	$154{,}71^{+}_{-}\ 1{,}17$	$80{,}10^{+}_{-}\ 0{,}73$	$27{,}26^{+}_{-}\ 0{,}38$

200 assinaturas

	2 Cells	4 Cells	8 Cells	16 Cells	32 Cells
MSE	$0{,}07^{+}_{-}\ 0{,}11$	$0{,}07^{+}_{-}\ 0{,}11$	$0{,}07^{+}_{-}\ 0{,}12$	$0{,}08^{+}_{-}\ 0{,}13$	$0{,}08^{+}_{-}\ 0{,}12$
MAE	$2{,}07^{+}_{-}\ 1{,}68$	$2{,}07^{+}_{-}\ 1{,}70$	$2{,}12^{+}_{-}\ 1{,}74$	$2{,}21^{+}_{-}\ 1{,}82$	$2{,}16^{+}_{-}\ 1{,}75$
Δt	$909{,}53^{+}_{-}\ 13{,}61$	$561{,}23^{+}_{-}\ 8{,}95$	$301{,}31^{+}_{-}\ 4{,}69$	$156{,}23^{+}_{-}\ 3{,}21$	$78{,}81^{+}_{-}\ 1{,}29$

400 assinaturas

	2 Cells	4 Cells	8 Cells	16 Cells	32 Cells
MSE	$0{,}071^{+}_{-}\ 0{,}11$	$0{,}07^{+}_{-}\ 0{,}11$	$0{,}07^{+}_{-}\ 0{,}12$	$0{,}08^{+}_{-}\ 0{,}13$	$0{,}08^{+}_{-}\ 0{,}12$
MAE	$1{,}53^{+}_{-}\ 1{,}27$	$1{,}53^{+}_{-}\ 1{,}26$	$1{,}46^{+}_{-}\ 1{,}19$	$1{,}49^{+}_{-}\ 1{,}21$	$1{,}49^{+}_{-}\ 1{,}22$
Δt	$1904{,}32^{+}_{-}\ 16{,}05$	$1186{,}97^{+}_{-}\ 11{,}02$	$669{,}76^{+}_{-}\ 6{,}15$	$335{,}20^{+}_{-}\ 3{,}43$	$169{,}61^{+}_{-}\ 1{,}71$

800 assinaturas

	2 Cells	4 Cells	8 Cells	16 Cells	32 Cells
MSE	$0{,}02^{+}_{-}\ 0{,}03$	$0{,}02^{+}_{-}\ 0{,}03$	$0{,}02^{+}_{-}\ 0{,}03$	$0{,}02^{+}_{-}\ 0{,}03$	$0{,}02^{+}_{-}\ 0{,}03$
MAE	$1{,}07^{+}_{-}\ 0{,}88$	$1{,}09^{+}_{-}\ 0{,}90$	$1{,}07^{+}_{-}\ 0{,}88$	$1{,}06^{+}_{-}\ 0{,}87$	$1{,}07^{+}_{-}\ 0{,}87$
Δt	$3656{,}45^{+}_{-}\ 18{,}04$	$2266{,}02^{+}_{-}\ 16{,}91$	$1277{,}91^{+}_{-}\ 10{,}57$	$666{,}70^{+}_{-}\ 5{,}36$	$325{,}52^{+}_{-}\ 2{,}86$

Valores de Speedup de VP^{RaE} e RaE com relação ao Minmax são apresentados na tabelas 6.15, onde VP^{RaE} apresenta valores superiores ao Minmax, chegando a ser 8 vezes mais rápido quando executado com 32 células. Contudo, se comparado com RaC, apresenta valores muito inferiores, chegando a ser 3,7 vezes mais lento, utilizando 2 células e 100 assinaturas.

Tabela 6.15 – VP^{RaE} (SP^{RaE}) e VP^{RaE} (SP^{RaE}) Speedup

k	$SP^{VP^{RaE}}$					SP^{RaE}				
	2	4	8	16	32	2	4	8	16	32
100	0,48	0,81	1,46	2,81	8,27	1,79	3,90	8,40	18,36	56,08
200	0,50	0,80	1,50	2,89	5,73	1,76	3,58	7,76	16,65	36,35
400	0,47	0,76	1,35	2,70	5,33	1,76	3,53	7,13	15,47	33,26
800	0,50	0,80	1,42	2,73	5,59	1,76	3,53	7,07	14,32	30,87

6.6.1.3 DaC x VP^{DaC}

A tabela 6.16 apresenta os resultados de MSE, MAE e do tempo de execução para a estratégia de VP^{DaC}. Os valores de MSE e MAE são bem baixos e se alteram muito pouco quando aumentado o número de células. Como esperado, os erros diminuem ao se aumentar o número de assinaturas. Além disso, um outro comportamento esperado que ocorre é o de se aumentar o tempo quando o número de assinaturas aumenta e o de diminuir o tempo quando o número de células aumenta. Quando VP^{DaC} é comparado ao DaC, apresenta valores de MAE e MSE muito melhores, entretanto, os valores de tempo são pelo menos nove vezes maiores. Os valores de MAE, MSE e tempo de DaC podem ser vistos na tabela 6.8.

Tabela 6.16 – MSE, MAE e tempo de comparação relacionado a J x $J^{\mathrm{VP^{DaC}}}$. Os valores de erro são multiplicados por 10^{-2}.

100 assinaturas

	2 Cells	4 Cells	8 Cells	16 Cells	32 Cells
MSE	$0{,}14^{+}_{-}\ 0{,}22$	$0{,}16^{+}_{-}\ 0{,}24$	$0{,}15^{+}_{-}\ 0{,}23$	$0{,}15^{+}_{-}\ 0{,}23$	$0{,}23^{+}_{-}\ 0{,}35$
MAE	$2{,}90^{+}_{-}\ 2{,}38$	$3{,}06^{+}_{-}\ 2{,}51$	$3{,}03^{+}_{-}\ 2{,}47$	$2{,}98^{+}_{-}\ 2{,}43$	$3{,}71^{+}_{-}\ 3{,}04$
Δt	$1220{,}95^{+}_{-}\ 3{,}53$	$582{,}65^{+}_{-}\ 1{,}46$	$288{,}92^{+}_{-}\ 1{,}22$	$143{,}60^{+}_{-}\ 0{,}50$	$47{,}97^{+}_{-}\ 0{,}27$

200 assinaturas

	2 Cells	4 Cells	8 Cells	16 Cells	32 Cells
MSE	$0{,}07^{+}_{-}\ 0{,}11$	$0{,}07^{+}_{-}\ 0{,}11$	$0{,}08^{+}_{-}\ 0{,}12$	$0{,}08^{+}_{-}\ 0{,}12$	$0{,}08^{+}_{-}\ 0{,}12$
MAE	$2{,}10^{+}_{-}\ 1{,}70$	$2{,}06^{+}_{-}\ 1{,}68$	$2{,}14^{+}_{-}\ 1{,}75$	$2{,}17^{+}_{-}\ 1{,}79$	$2{,}20^{+}_{-}\ 1{,}79$
Δt	$2463{,}93^{+}_{-}\ 40{,}21$	$1230{,}92^{+}_{-}\ 17{,}00$	$588{,}91^{+}_{-}\ 8{,}84$	$292{,}19^{+}_{-}\ 4{,}43$	$145{,}33^{+}_{-}\ 2{,}17$

400 assinaturas

	2 Cells	4 Cells	8 Cells	16 Cells	32 Cells
MSE	$0{,}04^{+}_{-}\ 0{,}05$	$0{,}04^{+}_{-}\ 0{,}06$	$0{,}04^{+}_{-}\ 0{,}06$	$0{,}04^{+}_{-}\ 0{,}06$	$0{,}04^{+}_{-}\ 0{,}06$
MAE	$1{,}47^{+}_{-}\ 1{,}20$	$1{,}51^{+}_{-}\ 1{,}24$	$1{,}49^{+}_{-}\ 1{,}24$	$1{,}55^{+}_{-}\ 1{,}27$	$1{,}51^{+}_{-}\ 1{,}23$
Δt	$5014{,}00^{+}_{-}\ 34{,}52$	$2506{,}80^{+}_{-}\ 9{,}81$	$1254{,}26^{+}_{-}\ 7{,}10$	$600{,}84^{+}_{-}\ 4{,}54$	$298{,}22^{+}_{-}\ 2{,}10$

800 assinaturas

	2 Cells	4 Cells	8 Cells	16 Cells	32 Cells
MSE	$0{,}02^{+}_{-}\ 0{,}03$	$0{,}02^{+}_{-}\ 0{,}03$	$0{,}02^{+}_{-}\ 0{,}03$	$0{,}02^{+}_{-}\ 0{,}03$	$0{,}02^{+}_{-}\ 0{,}03$
MAE	$1{,}09^{+}_{-}\ 0{,}90$	$1{,}09^{+}_{-}\ 0{,}89$	$1{,}07^{+}_{-}\ 0{,}88$	$1{,}01^{+}_{-}\ 0{,}83$	$1{,}04^{+}_{-}\ 0{,}85$
Δt	$9861{,}60^{+}_{-}\ 70{,}17$	$4924{,}60^{+}_{-}\ 20{,}79$	$2462{,}13^{+}_{-}\ 10{,}36$	$1230{,}79^{+}_{-}\ 4{,}93$	$588{,}33^{+}_{-}\ 2{,}64$

A tabela 6.17 exibe valores de Speedup de VP^{DaC} e DaC com relação ao Minmax, onde VP^{DaC} apresenta valores superiores ao Minmax, chegando a ser 4,7 vezes mais rápido, quando executado com 32 células. Em contrapartida, se comparado com DaC, apresenta valores muito inferiores, sendo ao menos 9,7 vezes mais lento, utilizando 2 células e 100 assinaturas.

Tabela 6.17 – VP^{DaC} ($SP^{VP^{DaC}}$) e DaC (SP^{DaC}) Speedup

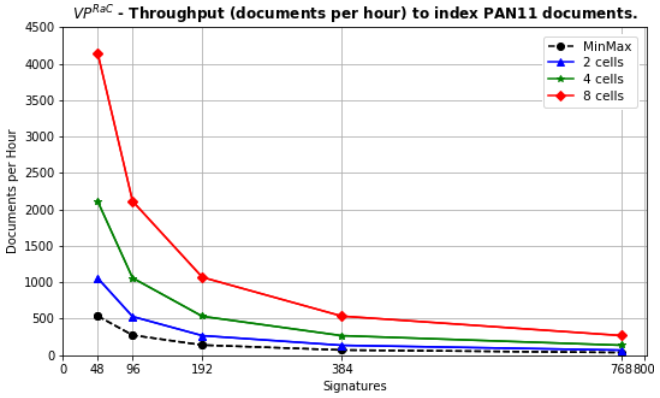
k	$SP^{VP^{DaC}}$					SP^{DaC}				
	2	4	8	16	32	2	4	8	16	32
100	0,18	0,39	0,78	1,57	4,70	1.75	3.85	8.33	18.26	55.45
200	0,18	0,37	0,77	1,54	3,11	1.74	3.50	7.69	16.62	36.31
400	0,18	0,36	0,72	1,50	3,03	1.74	3.45	6.99	15.33	33.22
800	0,18	0,37	0,74	1,47	3,09	1.74	3.45	6.88	13.98	30.54

6.6.2 EPHR

Para as estratégias de particionamento do vocabulário foi realizado o experimento EPHR nos mesmos moldes apresentados na seção 6.4, onde o experimento é realizado em duas etapas: indexação e busca. Cada estratégia de tratamento de resto (RaE, RaC e DaC) é analisada e comparada com as estratégias de particionamento das permutações. Estas comparações podem ser vistas nas seções 6.6.2.1, 6.6.2.2 e 6.6.2.3.

6.6.2.1 RaC x VP^{RaC}

A figura 6.9 exibe valores de vazão para VP^{RaC} na etapa de indexação do experimento EPHR. VP^{RaC} possui valores superiores ao Minmax, onde ao indexar utilizando 2 células, apresenta valores 97%, 94%, 94%, 95% e 96% maiores que o Minmax, ao gerar e indexar 48, 96, 192, 384, 768 assinaturas, respectivamente. Além disso, com o aumento das células, pode chegar a um valor 688% maior que o Minmax com 8 células. Contudo, quando comparado ao RaC apresenta desempenho levemente inferior, podendo chegar a 2,1% de perda. Os valores de vazão de RaC são apresentados na seção 6.4.1.



k	$VP^{RaC}_{2\text{ cells}}$	$VP^{RaC}_{4\text{ cells}}$	$VP^{RaC}_{8\text{ cells}}$
48	1055,89	2112,48	4145,50
96	528,25	1059,12	2111,99
192	266,00	531,76	1067,65
384	133,14	265,97	533,91
768	66,65	134,20	267,70

Figura 6.9 – Vazão (documentos por hora) de VP^{RaC} para indexar documentos do PAN-11.

Na tarefa de busca pela fonte de plágio, a primeira questão a ser avaliada é o tempo de extração de consulta ($Q1$), apresentado na tabela 6.18. A extração é calculada através do Speedup do método com relação ao tempo de consulta do Minmax, onde VP^{RaC} demonstra bom desempenho, comparável ao do RaC com relação ao Minmax, mas com pequena perda.

Tabela 6.18 – Tempo médio de extração de consulta do Minmax (em segundo) e o Speedup de $SP^{VP^{RaC}}$ e SP^{RaC}

k	Δt MinMax	$SP^{VP^{RaC}}$			SP^{RaC}		
		2	4	8			
48	6.64	1.96	3.87	7.73	1.99	3.92	7.74
96	13.26	1.96	3.88	7.75	1.99	3.93	7.82
192	26.46	1.95	3.88	7.80	1.99	3.95	7.86
384	53.35	1.98	3.94	7.95	2.00	3.99	7.97
768	105.68	1.95	3.93	7.89	1.99	3.97	7.87

Para a questão de tempo de busca ($Q2$), a figura 6.10 mostra a comparação do tempo total de busca para Minmax e VP^{RaC} ao selecionar 1597 e 11975 documentos. Os resultados para 3991 e 7983 não são apresentados, pois possuem comportamento semelhante, podendo serem vistos na figura B.4 do apêndice. Como pode ser notado, VP^{RaC} apresenta comportamento semelhante a RaC, apresentado na seção 6.4.2.2, onde o número de células é, aproximadamente, o número de vezes que o método é mais rápido que o Minmax, como exemplo, com 2 células VP^{RaC} busca duas vezes mais rápido que o Minmax.

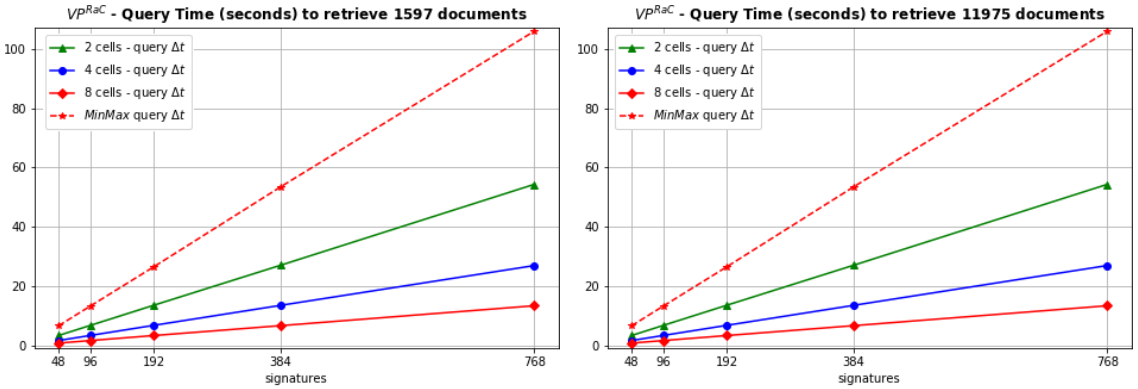


Figura 6.10 – Tempo total de busca de VP^{RaC} para selecionar 1597 e 11975 documentos.

A figura 6.11 apresenta valores de Speedup do Minmax e VP^{RaC} na tarefa de busca com relação ao tempo usado pelo BM25, ao selecionar 1597 e 11975 documentos. Os resultados para 3991 e 7983 não são apresentados, pois possuem comportamento semelhante, podendo serem vistos na figura B.10 do apêndice. Como pode

ser notado, VP^{RaC} apresenta comportamento semelhante a RaC, apresentado na seção 6.4.2.2, onde teve um Speedup melhor que o BM25 com 8 células para gerar 48 e 96 assinaturas e com 4 células para gerar 48 assinaturas.

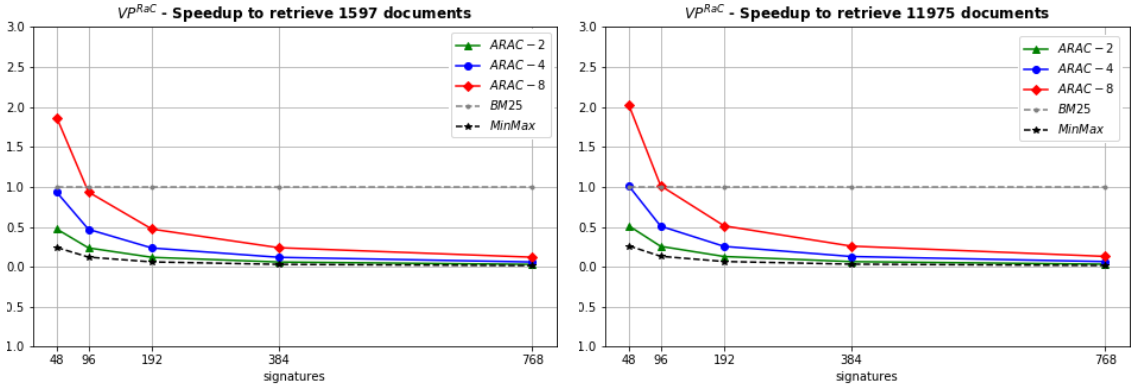


Figura 6.11 – Speedup de VP^{RaC} e Minmax com relação ao tempo de busca do BM25 para 1597 e 11975 documentos.

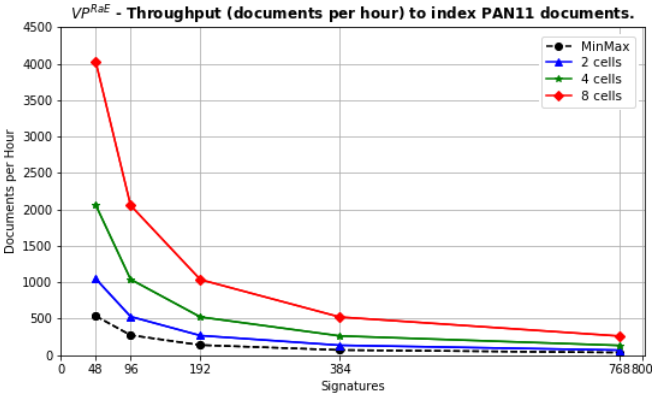
Com o objetivo de analisar o *recall* ($Q3$) foram selecionadas as configurações de RaC e VP^{RaC} que obtiveram melhor desempenho de Speedup com relação ao BM25. Assim sendo, a tabela 6.19 foi criada, apresentando valores de MAE e Speedup para Minmax, RaC e VP^{RaC}. Como esperado, com o aumento do número de assinaturas, o *recall* tem uma melhora no $MAE_R^{VP^{RaC}}$ em comparação ao BM25. Entretanto, se comparado ao Minmax e ao RaC, apresenta erro superior. Nas tabelas B.6 e B.7 do apêndice podem ser vistos todos os valores de MAE da estratégia VP^{RaC} com relação a Minmax e BM25.

Tabela 6.19 – MAE do Recall e Speedup para Minmax, RaC e VP^{RaC} com 48 e 96 assinaturas em relação ao BM25.

results to retrieve p documents with 48 fingerprints										
p	Minmax		RAC$^{4\ cells}$		$VP^{RaC}_{4\ cells}$		RAC$^{8\ cells}$		$VP^{RaC}_{8\ cells}$	
	MAE	Sp^a	MAE	Sp^a	MAE	Sp^a	MAE	Sp^a	MAE	Sp^a
1597	**0.11**	0.24	0.12	0.94	0.15	0.93	0.12	1.85	0.16	**1.87**
3991	**0.09**	0.24	0.11	0.92	0.12	0.91	0.10	1.81	0.13	**1.82**
7983	**0.17**	0.23	0.19	0.90	0.21	0.89	0.18	1.77	0.26	**1.79**
11975	**0.10**	0.26	0.11	1.02	0.13	1.01	0.11	2.01	0.18	**2.02**
results to retrieve p documents with 96 fingerprints										
p	Minmax		RAC$^{4\ cells}$		$VP^{RaC}_{4\ cells}$		RAC$^{8\ cells}$		$VP^{RaC}_{8\ cells}$	
	MAE^{MM}_{R}	Sp^a	MAE	Sp^a	MAE	Sp^a	MAE	Sp^a	MAE	Sp^a
1597	**0.08**	0.12	0.09	0.47	0.11	0.47	0.10	**0.93**	0.14	**0.93**
3991	**0.06**	0.12	0.08	0.46	0.09	0.46	0.08	**0.91**	0.11	**0.91**
7983	**0.15**	0.12	0.17	0.45	0.18	0.45	0.16	0.89	0.19	**0.90**
11975	**0.09**	0.13	**0.09**	0.51	0.11	0.51	0.10	**1.01**	0.12	**1.01**

6.6.2.2 RaE x VP^{RaE}

A figura 6.9 apresenta valores de vazão para VP^{RaE}, onde apresenta valores 95%, 94%, 95%, 96% e 97% maiores que o Minmax, ao gerar e indexar 48, 96, 192, 384, 768 assinaturas, respectivamente e usar 2 célula. Além disso, com o aumento das células, pode chegar a um valor 668% maior que o Minmax com 8 células. Entretanto, quando comparado ao RaE, apresenta desempenho levemente inferior, podendo chegar a 2,6% de perda. Os valores de vazão de RaE são apresentados na seção 6.4.1.



k	$VP^{RaE}_{2\ cells}$	$VP^{RaE}_{4\ cells}$	$VP^{RaE}_{8\ cells}$
48	1047,02	2063,75	4028,72
96	527,90	1041,08	2054,87
192	267,06	523,21	1035,69
384	133,77	262,72	521,34
768	66,87	131,81	261,01

Figura 6.12 – Vazão (documentos por hora) de VP^{RaE} para indexar documentos do PAN-11.

A tabela 6.20 apresenta valores de Speedup oriundos da avaliação quanto a extração de consulta ($Q1$) da tarefa de busca pela fonte de plágio. Os valores de Speedup exibidos são calculados entre o método (VP^{RaC} ou RaC) e o Minmax. Como pode ser visto, VP^{RaC} demonstra bom desempenho com relação ao tempo de extração do Minmax, comparável aos valores apresentados por RaC, mas com certa perda.

Tabela 6.20 – Tempo médio de extração de consulta do Minmax (em segundo) e o Speedup de $SP^{VP^{RaE}}$ e SP^{RaE}

k	Δt MinMax	$SP^{VP^{RaE}}$			SP^{RaE}		
		2	4	8	2	4	8
48	6.64	1.96	3.82	7.54	1.98	3.95	7.74
96	13.26	1.95	3.84	7.59	1.98	3.92	7.80
192	26.46	1.96	3.85	7.64	1.98	3.94	7.84
384	53.35	1.98	3.91	7.79	1.99	4.00	7.95
768	105.68	1.96	3.88	7.73	1.98	3.98	7.91

A figura 6.13 é apresentada com valores para avaliação do tempo de busca ($Q2$), onde é apresentado o comportamento de Minmax e VP^{RaE} com relação ao tempo de busca e a variação do número de assinaturas, células e documentos selecionados. Os resultados para 3991 e 7983 não são apresentados, pois possuem comportamento semelhante, podendo serem vistos na figura B.5 do apêndice. É notável que VP^{RaE} apresenta comportamento semelhante a RaE, apresentado na seção 6.4.2.2. VP^{RaE} e RaE apresentam valores onde o número de células é, aproximadamente, o número de vezes que o método é mais rápido que o Minmax. Por exemplo, quando utilizadas 2 células VP^{RaE} e RaE buscam duas vezes mais rápido que o Minmax.

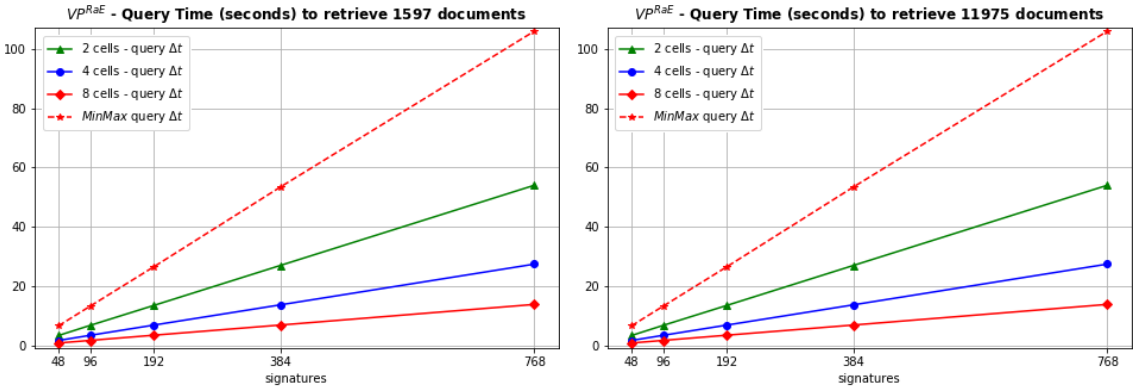


Figura 6.13 – Tempo total de busca de VP^{RaE} para selecionar 1597 e 11975 documentos.

Ao avaliar o Speedup do Minmax e VP^{RaE} com relação ao BM25 na tarefa de busca, foi criada a figura 6.14, onde são apresentados valores de Speedup ao selecionar 1597 e 11975 documentos. Os resultados para 3991 e 7983 não são apresentados,

pois possuem comportamento semelhante, podendo serem vistos na figura B.11 do apêndice. É notável que VP^{RaE} apresenta comportamento semelhante a RaE, apresentado na seção 6.4.2.2, onde teve um Speedup melhor que o BM25 com 8 células para gerar 48 e 96 assinaturas e com 4 células para gerar 48 assinaturas.

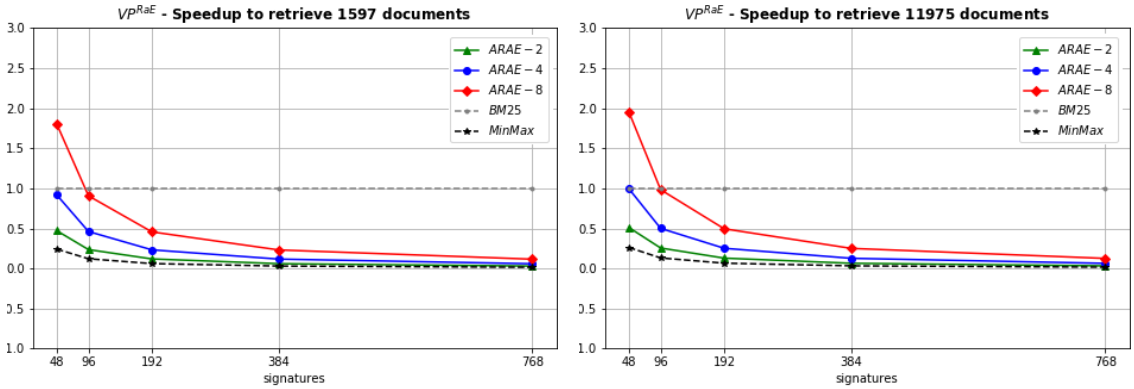


Figura 6.14 – Speedup de VP^{RaE} e Minmax com relação ao tempo de busca do BM25 para 1597 e 11975 documentos.

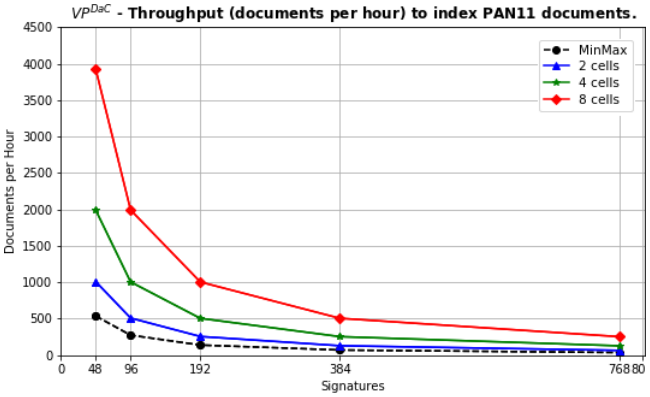
Visando responder a terceira questão com relação a análise do *recall* ($Q3$) foram selecionadas as configurações de RaE e VP^{RaE} que obtiveram melhor desempenho de Speedup com relação ao BM25. A tabela 6.21 apresenta valores de MAE e Speedup para Minmax, RaE e VP^{RaE}. Quando comparado ao Minmax, VP^{RaE} apresenta valores de erro próximo, chegando a ter melhor desempenho com 8 células e selecionando 1597 documentos. Quando comparado com o RaE, apresenta valores de erro mais baixo, na maioria dos casos. Nas tabelas B.6 e B.7 do apêndice podem ser vistos todos os valores de MAE da estratégia VP^{RaE} com relação a Minmax e BM25.

Tabela 6.21 – Minmax, RaE e VP^{RaE} recall MAE and Speedup from BM25 results with 48 and 96 fingerprints.

results to retrieve p documents with 48 fingerprints										
p	Minmax		RAE$^{4\ cells}$		$VP^{RaE}_{4\ cells}$		RAE$^{8\ cells}$		$VP^{RaE}_{8\ cells}$	
	MAE	Sp^2	MAE	Sp^a	MAE	Sp^a	MAE	Sp^a	MAE	Sp^a
1597	0.11	0.24	0.12	0.95	0.12	0.92	0.12	**1.85**	**0.10**	1.80
3991	**0.09**	0.24	0.11	0.93	0.10	0.90	0.10	**1.81**	**0.09**	1.76
7983	**0.17**	0.23	0.19	0.91	0.18	0.88	**0.17**	**1.77**	0.18	1.72
11975	**0.10**	0.26	0.11	1.03	0.11	1.00	0.11	**2.01**	0.11	1.96
results to retrieve p documents with 96 fingerprints										
p	Minmax		RaE$^{4\ cells}$		$VP^{RaE}_{4\ cells}$		RaE$^{8\ cells}$		$VP^{RaE}_{8\ cells}$	
	MAE^{MM}_{R}	Sp^3	MAE	Sp^a	MAE	Sp^a	MAE	Sp^a	MAE	Sp^a
1597	**0.08**	0.12	0.09	0.47	0.09	0.46	0.10	**0.93**	**0.08**	0.91
3991	**0.06**	0.12	0.08	0.46	0.08	0.45	0.08	**0.91**	0.07	0.89
7983	**0.15**	0.12	0.17	0.45	0.16	0.44	0.16	**0.89**	0.16	0.87
11975	**0.09**	0.13	0.10	0.51	**0.09**	0.50	0.10	**1.01**	0.10	0.98

6.6.2.3 DaC x VP^{DaC}

A figura 6.15 exibe valores de vazão para VP^{DaC}. Quando comparado com Minmax, VP^{DaC} possui valores superiores, atingindo valores 88%, 88%, 86%, 87% e 87% maiores que o Minmax, ao gerar e indexar 48, 96, 192, 384, 768 assinaturas, respectivamente, utilizando 2 células. Além disso, com o aumento das células, pode chegar a um valor 642% maior que o Minmax, utilizando 8 células. Em contrapartida, quando comparado ao DaC apresenta desempenho inferior, podendo chegar a 6,5% de perda. Os valores de vazão de DaC são apresentados na seção 6.5.2.



k	$VP^{DaC}_{2\ cells}$	$VP^{DaC}_{4\ cells}$	$VP^{DaC}_{8\ cells}$
48	1008,02	1994,88	3927,11
96	507,70	1005,70	1992,00
192	254,20	504,48	1002,45
384	127,56	252,93	504,06
768	63,50	126,75	252,10

Figura 6.15 – Vazão (documentos por hora) de VP^{DaC} para indexar documentos do PAN-11.

A primeira questão a ser avaliada na tarefa de busca pela fonte de plágio é o tempo de extração de consulta ($Q1$). Os valores de Speedup, para VP^{DaC} e DaC, com relação ao tempo de extração de Minmax são apresentados na tabela 6.22. VP^{DaC} demonstra com desempenho com relação ao Minmax, mas apresenta perda significativa com relação ao DaC, podendo chegar a 6%.

Tabela 6.22 – Tempo médio de extração de consulta do Minmax (em segundo) e o Speedup de $SP^{VP^{DaC}}$ e SP^{DaC}

k	Δt MinMax	$SP^{VP^{DaC}}$			SP^{DaC}		
		2	4	8	2	4	8
48	6.64	1.87	3.69	7.29	1.97	3.91	7.71
96	13.26	1.88	3.70	7.36	1.97	3.91	7.76
192	26.46	1.87	3.70	7.40	1.97	3.93	7.81
384	53.35	1.90	3.77	7.54	2.00	3.98	7.96
768	105.68	1.87	3.74	7.48	1.99	3.97	7.93

A segunda questão a ser avaliada é o tempo de busca ($Q2$), para tal, a figura 6.16 é apresentada com valores de tempo para Minmax e VP^{DaC} ao selecionar 1597 e 11975 documentos. Os resultados para 3991 e 7983 não são apresentados, pois possuem comportamento semelhante, podendo serem vistos na figura B.6 do apêndice. Como pode ser notado, VP^{DaC} e DaC apresentam comportamento semelhante, apresentam valores onde o número de células é, aproximadamente, o número de vezes que o método é mais rápido que o Minmax. Por exemplo, quando utilizadas 2 células, ambos os métodos buscam duas vezes mais rápido que o Minmax. Valores de busca para DaC são apresentados na seção 6.5.2.

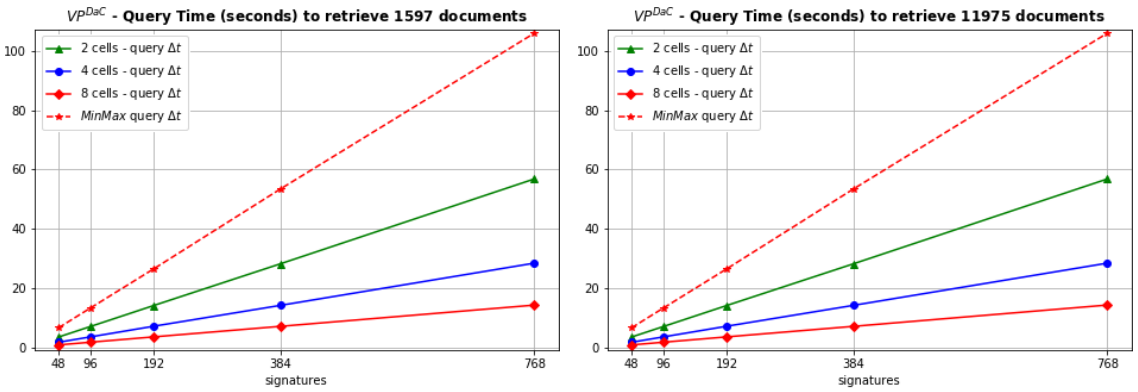


Figura 6.16 – Tempo total de busca de VP^{DaC} para selecionar 1597 e 11975 documentos.

A figura 6.17 apresenta valores de avaliação do Speedup do Minmax e VP^{DaC} com relação ao BM25 na tarefa de busca ao selecionar 1597 e 11975 documentos.

Os resultados para 3991 e 7983 não são apresentados, pois possuem comportamento semelhante, podendo serem vistos na figura B.12 do apêndice. É notável que VP^{DaC} e DaC apresentam comportamento semelhante, onde tiveram um Speedup melhor que o BM25 com 8 células para gerar 48 e 96 assinaturas e com 4 células para gerar 48 assinaturas. Valores de DaC são exibidos na seção 6.5.2

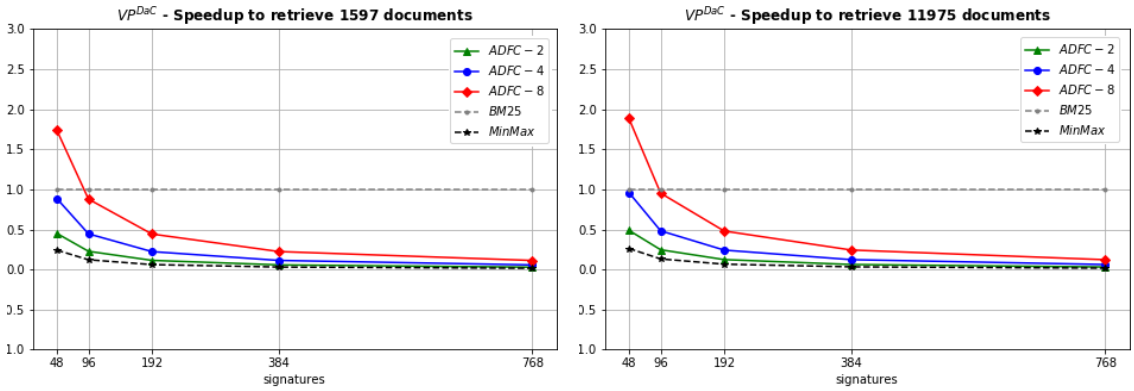


Figura 6.17 – Speedup de VP^{DaC} e Minmax com relação ao tempo de busca do BM25 para 1597 e 11975 documentos.

Ao analisar o *recall* ($Q3$) foram selecionadas as configurações de DaC e VP^{DaC} que obtiveram melhor desempenho de Speedup com relação ao BM25. Os valores de MAE e Speedup para Minmax, DaC e VP^{DaC} são apresentados na tabela 6.23. Quando comparado ao Minmax, o método VP^{DaC} apresenta valores de MAE bem próximo, chegando a ter desempenho igual ao Minmax em duas ocasiões. Já quando comparado ao DaC, apresenta desempenho bem superior. Nas tabelas B.6 e B.7 do apêndice podem ser vistos todos os valores de MAE da estratégia VP^{DaC} com relação a Minmax e BM25.

Tabela 6.23 – Minmax, DaC e VP^{DaC} recall MAE and Speedup from BM25 results with 48 and 96 fingerprints.

results to retrieve p documents with 48 fingerprints										
p	Minmax		DaC[4 cells]		$VP^{DaC}_{4\text{ cells}}$		DaC[8 cells]		$VP^{DaC}_{8\text{ cells}}$	
	MAE	Sp^4	MAE	Sp^a	MAE	Sp^a	MAE	Sp^a	MAE	Sp^a
1597	**0.11**	0.24	0.20	0.94	0.12	0.89	0.23	**1.86**	0.12	1.74
3991	**0.09**	0.24	0.17	0.92	0.11	0.87	0.21	**1.82**	**0.09**	1.70
7983	**0.17**	0.23	0.27	0.90	0.18	0.85	0.34	**1.78**	0.18	1.67
11975	**0.10**	0.26	0.18	1.02	0.11	0.97	0.27	**2.01**	0.11	1.89
results to retrieve p documents with 96 fingerprints										
p	Minmax		DaC[4 cells]		$VP^{DaC}_{4\text{ cells}}$		DaC[8 cells]		$VP^{DaC}_{8\text{ cells}}$	
	MAE^{MM}_R	Sp^5	MAE	Sp^a	MAE	Sp^a	MAE	Sp^a	MAE	Sp^a
1597	**0.08**	0.12	0.17	0.47	0.09	0.44	0.21	**0.94**	0.09	0.88
3991	**0.06**	0.12	0.14	0.46	0.07	0.43	0.18	**0.92**	0.08	0.86
7983	**0.15**	0.12	0.22	0.45	0.16	0.43	0.28	**0.90**	0.16	0.84
11975	**0.09**	0.13	0.14	0.51	**0.09**	0.48	0.20	**1.02**	**0.09**	0.95

Capítulo 7

Conclusões

Neste trabalho foi abordado o problema do plágio e suas definições, tipos e estratégias de identificação. Dentre essas estratégias de identificação, teve como foco principal a detecção de plágio externo. Sendo essa definida como uma estratégia onde a região de busca é uma coleção de documentos conhecidos, tornando essa estratégia solucionável como uma tarefa de Busca e Recuperação de Informação, onde um conjunto de documentos (ou trecho deles) mais similares é recuperado a partir de um documento consultado. No caso do plágio externo, um conjunto de documentos candidatos a plágio é recuperado a partir de um documento suspeito consultado.

Além disso, foi apresentado um *workflow* para detecção de plágio externo, que compreende algumas etapas, dentre elas a Busca Heurística, que tem como objetivo reduzir a carga de trabalho das etapas posteriores selecionando um conjunto de documentos candidatos, reduzindo a quantidade de comparações a serem realizadas. A Busca Heurística, por ter a necessidade de ser uma etapa executada de forma rápida, pode ser implementada como uma busca por vizinhos mais próximos (NNS). Foram apresentadas algumas formas de NNS, sendo a busca aproximada (ANN) a escolhida por este trabalho.

Dentre os métodos de ANN existentes, o Locality-Sensitive Hashing (LSH) é um dos métodos mais utilizados por ser eficiente na presença de uma grande quantidade de dados, mapeando cada documento em um conjunto de assinaturas, melhorando o tempo de indexação e busca. Foi visto também que, o LSH pode ser dividido em duas categorias: projeções aleatórias e permutações aleatórias. Para ANN usando LSH por permutações randômicas, WANG *et al.* (2014) propuseram um *workflow*, posteriormente detalhado por DUARTE *et al.* (2017). Este *workflow* é utilizado nos trabalhos relacionados CSA e CSA_L, propostos por DUARTE *et al.* (2017), para a execução da Busca Heurística na detecção do plágio externo e também é utilizado pelas propostas deste trabalho.

Foram propostas, primeiramente, duas estratégias para particionamento das permutações, nomeadas como *Remainder at Cell* e *Remainder at End*, que diferem pela forma com que tratam os valores restantes do particionamento. Posteriormente, outras estratégias foram criadas, como *Distributed at Cell*, para tratamento de resto da divisão, e o particionamento do vocabulário (VP). O que diferencia o particionamento das permutações do particionamento do vocabulário é a etapa do *workflow* na qual o particionamento acontece, podendo ambos utilizarem das estratégias de tratamento de resto RaC, RaE e DaC.

Ambas as propostas de particionamento tem como objetivo a diminuição do tempo de indexação e busca. Essa diminuição acontece pois ao particionar são selecionadas mais assinaturas por permutação, diminuindo o número de vezes que a etapa de permutações é executada. A fim de avaliar as estratégias propostas, foram realizados dois experimentos computacionais, denominados PJSE e EPHR. Os resultados foram apresentados para o particionamento das permutações, RaE e RaC, e depois estes, comparados a DaC. Após isso, cada estratégia de tratamento de resto foi aplicada ao particionamento por vocabulário, nomeados por VP^{RAE}, VP^{RAC} e VP^{DaC}, e depois comparados a RaE, RaC e DaC, respectivamente.

No experimento PJSE, as estratégias RaE e RaC apresentaram um comportamento semelhante em relação ao erro produzido (conforme medido pelo MAE e MSE). Além disso, a medida que a quantidade de células é aumentada, o tempo de processamento é bastante reduzido e os erros MSE e MAE têm um pequeno aumento. Quando comparados ao DaC, apresentaram desempenho superior com relação ao MAE, MSE e tempo. Quando RaE, RaC e DaC são comparados ao particionamento do vocabulário, apresentam os seguintes resultados:

(i) RaC x VP^{RAC}: RaC apresenta valores de MAE e MSE, na maioria das vezes, melhores e valores de tempo pelo menos duas vezes menores.

(ii) RaE x VP^{RAE}: RaE apresenta valores de MAE e MSE, na maioria das vezes, piores e valores de tempo pelo menos três vezes menores.

(iii) DaC x VP^{DaC}: DaC apresenta valores de MAE e MSE muito piores, entretanto, os valores de tempo são pelo menos nove vezes menores.

No experimento EPHR, as tarefas de indexação e busca por origem de plágio foram avaliadas. Além disso, a tarefa de busca por origem de plágio teve como objetivo responder: (Q1) com que rapidez a estratégia pré-processa e extrai uma consulta de um documento suspeito? (Q2) A estratégia é mais rápida que os *baselines* de busca por origem de plágio? (Q3) A estratégia é melhor que os *baselines* ao buscar e recuperar os documentos de origem de um documento plagiado?

Os resultados da tarefa de indexação mostraram que o RaE e o RaC tinham uma taxa de vazão de pelo menos 101%, 293%, 411% superior a $MinMax$, CSA e CSA_L. Quando comparados ao DaC, apresentaram valores de vazão semelhantes. Quando comparados as estratégias de particionamento dos vocabulários, RaC, RaE e DaC apresentam desempenho 2,1%, 2,6%, 6,5% superior a VP^{RAE}, VP^{RAC} e VP^{DaC}, respectivamente.

Para responder ao Q1, o tempo de extração da consulta foi avaliado. Ambos os métodos são pelo menos duas vezes mais rápidos que o Minmax, pelo menos 30% mais rápidos que o CSA_L. Além disso, o Speedup de RaE e o RaC aumentam proporcionalmente ao número de células. Quando comparado ao DaC, as estratégia RaE e RaC possuem desempenho ligeiramente superior. Quando comparados as estratégias de particionamento dos vocabulários, RaC, RaE e DaC apresentam desempenho superior.

O impacto da partição nos resultados da tarefa de busca das estratégias foi avaliado para 2, 4 e 8 células. Além disso, o tempo para gerar 48, 96, 192, 384 e 768 assinaturas e executar a tarefa de busca foi medido e a velocidade das estratégias foram avaliadas para responder a segunda questão Q2. RaC e RaE superaram o estado da arte no tempo da tarefa de busca, onde com 8 células para gerar 48 assinaturas foram entre 77% e 101% mais rápido que o BM25, pelo menos 354% mais rápido que CSA_L e pelo menos 247% mais rápido que CSA. Além disso, os resultados mostraram que RaE e RaC com 2, 4 e 8 células são aproximadamente 2, 4 e 8 vezes mais rápidas que $MinMax$. Quando comparado com DaC, VP^{RAE}, VP^{RAC} e VP^{DaC}, possui desempenho semelhante.

Para responder à Q3, o *recall* para recuperar até 1597, 3991, 7983 e 11975 documentos foi avaliada na tarefa busca. Em relação ao *recall*, RaE e RaC com 8 células para gerar 48 assinaturas apresentaram um MAE entre 0,10 e 0,17 quando comparado ao *recall* BM25. Além disso, RaE e RaC tiveram um MAE aproximadamente 14,8% superior ao Minmax. Em contrapartida, ambos tiveram desempenho superior a CSA e CSA_L, com diferença de aproximadamente 13,3% no MAE. Quando RaE e RaC são comparados ao DaC, para execuções com 2 e 4 células, DaC tem valores de MAE superiores, já com 8 células para gerar 48 assinaturas, DaC apresenta valores inferiores, semelhantes ao Minmax.

Quando comparado ao particionamento dos vocabulários, RaC, RaE e DaC apresentam:

(i) RaC x VP^{RAC}: Quando comparado ao RaC, VP^{RAC} apresenta valores piores de MAE.

(ii) RaE x VP^{RAE}: Quando comparado ao RaE, apresenta valores de MAE semelhantes.

(iii) DaC x VP^{DaC}: Quando comparado ao DaC, apresenta valores melhores de MAE.

A partir dos resultados dos experimentos PSJE e EPHR pode ser concluído que as estratégias de particionamento das permutações, RaE e RaC, tiveram desempenho levemente superior com relação a DaC, VP^{RAE}, VP^{RAC} e VP^{DaC}. Além disso, as RaE e RaC foram aproximadamente 101% mais rápidos que BM25, 8 vezes mais rápidos que $MinMax$ e pelo menos 354% e 247% mais rápidos que CSA_L e CSA na tarefa de busca. As estratégias tiveram uma taxa de transferência 101%, 293% e 411% superior a $MinMax$, CSA_L e CSA na tarefa de indexação. No entanto, as RaE e RaC apresentam dois pontos fracos. Primeiro, ambos foram superados pelo BM25 na tarefa de indexação. Além disso, eles eram mais imprecisos que o BM25 para recuperar os documentos de origem de um documento plagiado.

Apesar dos pontos fracos, as RaE e RaC superaram o estado da arte na tarefa de busca, pois: (i) apresentaram maior velocidade do que o estado da arte; (ii) a perda de *recall* é pequena quando comparada à redução do tempo de consulta; (iii) em grandes coleções de texto, os requisitos para responder a consultas em um período de tempo razoável são críticos. Nesse cenário, usar as estratégias de particionamento é uma vantagem, já que o tempo de geração de assinatura é reduzido ao custo de uma pequena perda de *recall*.

Como trabalhos futuros, com o objetivo de melhorar o *recall* das estratégias na etapa de busca, será realizada uma avaliação de diferentes abordagens para pré-processamento de documentos, extração e expansão de consultas. Outro trabalho futuro é implementar as estratégias de particionamento e tratamento de restos em um ambiente distribuído a fim de reduzir o tempo de indexação e busca. Outro trabalho a ser explorado é a aplicação das estratégias de particionamento e tratamento de restos usando outras funções de seleção como Minwise Hashing, CSA e CSA_L. Além disso, como as estratégias podem ser aplicadas a qualquer problema do NNS, existe a intenção de aplicá-las como recuperação de informações multimídia. Outra ideia a ser explorada é a variação de maneiras de distribuição para o DaC, visando distribuir elementos mais significativos pelas células com o objetivo de aumentar a probabilidade deles serem selecionados. Finalmente, uma avaliação em coleções maiores de texto deve ser realizada.

Referências Bibliográficas

ALZAHRANI, S. M., SALIM, N., ABRAHAM, A., 2012, "Understanding plagiarism linguistic patterns, textual features, and detection methods", *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, v. 42, n. 2, pp. 133–149.

ANDONI, A., INDYK, P., RAZENSHTEYN, I., 2018, "Approximate nearest neighbor search in high dimensions", *arXiv preprint arXiv:1806.09823*.

BAEZA-YATES, R., RIBEIRO-NETO, B., 2011, *Modern Information Retrieval: The Concepts and Technology Behind Search*. Pearson Addison Wesley. ISBN: 978-0-321-41691-9.

BARRÓN-CEDEÑO, A., VILA, M., MARTÍ, M., et al., 2013, "Plagiarism Meets Paraphrasing: Insights for the Next Generation in Automatic Plagiarism Detection", *Comput. Linguist.*, v. 39, n. 4 (dec.), pp. 917–947. ISSN: 0891-2017. doi: 10.1162/COLI_a_00153. http://dx.doi.org/10.1162/COLI_a_00153 .

BARRÓN-CEDEÑO, A., 2010, *On the mono-and cross-language detection of text reuse and plagiarism*. Ph.D. Thesis, Universidade Politécnica de Valência.

BELLMAN, R., CORPORATION, R., COLLECTION, K. M. R., 1957, *Dynamic Programming*. Rand Corporation research study. Princeton University Press. ISBN: 9780691079516. https://books.google.it/books?id=wdtoPwAACAAJ .

BRADLEY, J., NI, Y., CHU, K., 2017. "Detecting Abuse at Scale: Locality Sensitive Hashing at Uber Engineering". databricks.com/blog/2017/05/09/detecting-abuse-scale-locality-sensitive-hashing-uber-engineering.html. [Online; accessed 19-July-2018].

Braschler, M., Harman, D., Pianta, E. (Eds.), 2010, *Notebook Papers of CLEF 2010 Labs and Workshops, 22-23 September, Padua, Italy*. ISBN: 978-88-904810-2-4. http://ceur-ws.org/Vol-1176 .

BRODER, A. Z., 1997, "On the resemblance and containment of documents". In: *Compression and Complexity of Sequences 1997. Proceedings*, pp. 21–29. IEEE.

BRODER, A. Z., GLASSMAN, S. C., MANASSE, M. S., et al., 1997, "Syntactic clustering of the web", *Computer Networks and ISDN Systems*, v. 29, n. 8, pp. 1157–1166.

BRODER, A. Z., CHARIKAR, M., FRIEZE, A. M., et al., 2000, "Min-wise independent permutations", *Journal of Computer and System Sciences*, v. 60, n. 3, pp. 630–659.

Cappellato, L., Ferro, N., Jones, G., et al. (Eds.), 2015, *CLEF 2015 Evaluation Labs and Workshop – Working Notes Papers, 8-11 September, Toulouse, France*, CEUR Workshop Proceedings. CEUR-WS.org. `http://ceur-ws.org/Vol-1391` .

CASANOVA, M. A., CÂMARA, G., DAVIS, C., et al., 2005, *Banco de dados geográficos*. MundoGEO Curitiba.

CLOUGH, P., 2000, *Plagiarism in natural and programming languages: an overview of current tools and technologies*. Relatório técnico, University of Sheffield.

CLOUGH, P., STEVENSON, M., 2011, "Developing a corpus of plagiarised short answers", *Language Resources and Evaluation*, v. 45, n. 1, pp. 5–24.

DAHLGAARD, S., KNUDSEN, M. B. T., ROTENBERG, E., et al., 2015, "Hashing for statistics over k-partitions". In: *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pp. 1292–1310. IEEE.

DITTMAR, C., HILDEBRAND, K. F., GAERTNER, D., et al., 2012, "Audio forensics meets music information retrieval—a toolbox for inspection of music plagiarism". In: *Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European*, pp. 1249–1253. IEEE.

DUARTE, F., CALED, D., XEXÉO, G., 2017, "Minmax Circular Sector Arc for External Plagiarism's Heuristic Retrieval stage", *Knowledge-Based Systems*, v. 137, n. Supplement C, pp. 1 – 18. ISSN: 0950-7051. doi: https://doi.org/10.1016/j.knosys.2017.08.013. `http://www.sciencedirect.com/science/article/pii/S0950705117303696` .

DUARTE, F. R., 2017, *IDENTIFICANDO PLAGIO EXTERNO COM LOCALITY-SENSITIVE HASHING*. Ph.D. Thesis, Universidade Federal do Rio de Janeiro.

EHSAN, N., SHAKERY, A., 2016, "Candidate document retrieval for cross-lingual plagiarism detection using two-level proximity information", *Information Processing & Management*, v. 52, n. 6, pp. 1004 – 1017. ISSN: 0306-4573. doi: http://dx.doi.org/10.1016/j.ipm.2016.04.006. http://www.sciencedirect.com/science/article/pii/S0306457316300784 .

EINSTEIN, A., 1905, "Zur Elektrodynamik bewegter Körper. (German) [On the electrodynamics of moving bodies]", *Annalen der Physik*, v. 322, n. 10, pp. 891–921. doi: http://dx.doi.org/10.1002/andp.19053221004.

ETHICS, C. F. Y., 2012. "2012 Report Card on the Ethics of American Youth". https://charactercounts.org/wp-content/uploads/2014/02/ReportCard-2012-DataTables.pdf .

FAWZY, F., 2016. "From speeches to Ph.D.'s: Politicians called out for copying". http://edition.cnn.com/2016/07/19/politics/politicians-plagiarism/index.html. [Online; accessed 01-march-2018].

FETTERLY, D., MANASSE, M., NAJORK, M., et al., 2003, "A large-scale study of the evolution of web pages". In: *Proceedings of the 12th international conference on World Wide Web*, pp. 669–678. ACM.

Forner, P., Navigli, R., Tufis, D. (Eds.), 2013, *CLEF 2013 Evaluation Labs and Workshop – Working Notes Papers, 23-26 September, Valencia, Spain*. CEUR-WS.org. ISBN: 978-88-904810-3-1. http://ceur-ws.org/Vol-1179 .

GAIZAUSKAS, R., FOSTER, J., WILKS, Y., et al., 2001, "The METER corpus: a corpus for analysing journalistic text reuse". In: *The Corpus Linguistics 2001 Conference*, pp. 214–223.

GIONIS, A., INDYK, P., MOTWANI, R., et al., 1999, "Similarity search in high dimensions via hashing". In: *Vldb*, v. 99, pp. 518–529.

GUARDIAN, 2011. "German defence minister resigns in PhD plagiarism row". www.theguardian.com/world/2011/mar/01/german-defence-minister-resigns-plagiarism. [Online; accessed 01-march-2018].

HUANG, A., 2008, "Similarity measures for text document clustering". In: *Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand*, pp. 49–56.

INDYK, P., MOTWANI, R., 1998, "Approximate nearest neighbors: towards removing the curse of dimensionality". In: *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pp. 604–613. ACM.

JI, J., LI, J., YAN, S., et al., 2013, "Min-max hash for jaccard similarity". In: *2013 IEEE 13th International Conference on Data Mining*, pp. 301–309. IEEE.

LEE, D. C., KE, Q., ISARD, M., 2010, "Partition min-hash for partial duplicate image discovery". In: *European Conference on Computer Vision*, pp. 648–662. Springer.

LESKOVEC, J., RAJARAMAN, A., ULLMAN, J. D., 2014, *Mining of massive datasets*. Cambridge University Press.

LI, J., ZHENG, R., CHEN, H., 2006, "From Fingerprint to Writeprint", *Commun. ACM*, v. 49, n. 4 (apr.), pp. 76–82. ISSN: 0001-0782. doi: 10.1145/1121949.1121951. http://doi.acm.org/10.1145/1121949.1121951 .

LI, P., KÖNIG, A. C., 2011, "Theory and applications of b-bit minwise hashing", *Communications of the ACM*, v. 54, n. 8, pp. 101–109.

LI, P., KÖNIG, C., 2010, "b-Bit minwise hashing". In: *Proceedings of the 19th international conference on World wide web*, pp. 671–680. ACM.

LI, P., OWEN, A., ZHANG, C.-H., 2012, "One permutation hashing". In: *Advances in Neural Information Processing Systems*, pp. 3113–3121.

LOWE, D. G., 2004, "Distinctive Image Features from Scale-Invariant Keypoints", *Int. J. Comput. Vision*, v. 60, n. 2 (nov.), pp. 91–110. ISSN: 0920-5691. doi: 10.1023/B:VISI.0000029664.99615.94. https://doi.org/10.1023/B:VISI.0000029664.99615.94 .

MARIMONT, R., SHAPIRO, M., 1979, "Nearest neighbour searches and the curse of dimensionality", *IMA Journal of Applied Mathematics*, v. 24, n. 1, pp. 59–70.

MARTIN, B., 1994, "Plagiarism: a misplaced emphasis", *Journal of Information Ethics*, v. 3, n. 2, pp. 36–47.

MAURER, H., KAPPE, F., ZAKA, B., 2006, "Plagiarism - A Survey", *Journal of Universal Computer Science*, v. 12, n. 8 (aug), pp. 1050–1084. http://www.jucs.org/jucs_12_8/plagiarism_a_survey .

MCCABE, D., 2009, "Academic Dishonesty in Nursing Schools: An Empirical Investigation", *The Journal of nursing education*, v. 48 (08), pp. 614–23. doi: 10.3928/01484834-20090716-07.

MEUSCHKE, N., GIPP, B., 2014, "Reducing computational effort for plagiarism detection by using citation characteristics to limit retrieval space". In: *IEEE/ACM Joint Conference on Digital Libraries*, pp. 197–200, Sept. doi: 10.1109/JCDL.2014.6970168.

MICHAELIS, D., 2019. "Plágio". https://michaelis.uol.com.br/moderno-portugues/busca/portugues-brasileiro/plagio/ .

MOHAMED, H., MARCHAND-MAILLET, S., 2015, "Quantized Ranking for Permutation-based Indexing", *Inf. Syst.*, v. 52, n. C (aug.), pp. 163–175. ISSN: 0306-4379. doi: 10.1016/j.is.2015.01.009. http://dx.doi.org/10.1016/j.is.2015.01.009 .

NIWATTANAKUL, S., SINGTHONGCHAI, J., NAENUDORN, E., et al., 2013, "Using of Jaccard coefficient for keywords similarity". In: *Proceedings of the International MultiConference of Engineers and Computer Scientists*, v. 1, pp. 13–15.

OXFORD, D., 2019. "Plágio". https://dictionary.cambridge.org/pt/dicionario/ingles/plagiarism .

PAN, J., MANOCHA, D., 2012, "Bi-level locality sensitive hashing for k-nearest neighbor computation". In: *2012 IEEE 28th International Conference on Data Engineering*, pp. 378–389. IEEE.

PEMMARAJU, S., SKIENA, S., 2003, *Computational Discrete Mathematics: Combinatorics and Graph Theory with Mathematica®*. Cambridge university press.

PLAGIARISM.ORG, 2017. "What is Plagiarism?" https://www.plagiarism.org/article/what-is-plagiarism .

POSNER, R. A., 2007, *The little book of plagiarism*. Pantheon.

POTTHAST, M., EISELT, A., BARRÓN-CEDEÑO, A., et al., 2011, "Overview of the 3rd International Competition on Plagiarism Detection". In: Petras,

V., Forner, P., Clough, P. (Eds.), *Working Notes Papers of the CLEF 2011 Evaluation Labs*, sep. ISBN: 978-88-904810-1-7. http://www.clef-initiative.eu/publication/working-notes .

POTTHAST, M., HAGEN, M., BEYER, A., et al., 2014, "Overview of the 6th International Competition on Plagiarism Detection". In: Cappellato, L., Ferro, N., Halvey, M., et al. (Eds.), *Working Notes Papers of the CLEF 2014 Evaluation Labs*, CEUR Workshop Proceedings. CLEF and CEUR-WS.org, sep. http://www.clef-initiative.eu/publication/working-notes .

PRIBERAM, D., 2019. "Plágio". https://dicionario.priberam.org/plagio .

SAMUELSON, P., 1994, "Self-plagiarism or Fair Use", *Commun. ACM*, v. 37, n. 8 (aug.), pp. 21–25. ISSN: 0001-0782. doi: 10.1145/179606.179731. http://doi.acm.org/10.1145/179606.179731 .

SHAKHNAROVICH, G., INDYK, P., DARRELL, T., 2006, *Nearest-neighbor methods in learning and vision: theory and practice*.

SLANEY, M., CASEY, M., 2008, "Locality-sensitive hashing for finding nearest neighbors [lecture notes]", *IEEE Signal processing magazine*, v. 25, n. 2, pp. 128–131.

STAMATATOS, E., 2009, "Intrinsic plagiarism detection using character n-gram profiles", *threshold*, v. 2, n. 1,500.

STEIN, B., ZU EISSEN, S. M., POTTHAST, M., 2007, "Strategies for Retrieving Plagiarized Documents". In: *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, pp. 825–826, New York, NY, USA. ACM. ISBN: 978-1-59593-597-7. doi: 10.1145/1277741.1277928. http://doi.acm.org/10.1145/1277741.1277928 .

THOMPSON, V. U., PANCHEV, C., OAKES, M., 2015, "Performance evaluation of similarity measures on similar and dissimilar text retrieval". In: *2015 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K)*, v. 01, pp. 577–584, Nov.

WANG, J., SHEN, H. T., SONG, J., et al., 2014, "Hashing for similarity search: A survey", *arXiv preprint arXiv:1408.2927*.

WANG, J., LIU, W., KUMAR, S., et al., 2016, "Learning to hash for indexing big data—A survey", *Proceedings of the IEEE*, v. 104, n. 1, pp. 34–57.

WANG, X., DING, X., TUNG, A. K. H., et al., 2013, "Efficient and Effective KNN Sequence Search with Approximate N-grams", *Proc. VLDB Endow.*, v. 7, n. 1 (sep.), pp. 1–12. ISSN: 2150-8097. doi: 10.14778/2732219.2732220. http://dx.doi.org/10.14778/2732219.2732220 .

WIKIMEDIA, 2019. "Wikipedia statistics All languages". https://stats.wikimedia.org/EN/TablesWikipediaZZ.htm .

WIKIPEDIA, 2019. "Wikipedia:Size comparisons". https://en.wikipedia.org/wiki/Wikipedia:Size_comparisons .

ZHANG, H., CHOW, T. W., 2011, "A coarse-to-fine framework to efficiently thwart plagiarism", *Pattern Recognition*, v. 44, n. 2, pp. 471 – 487. ISSN: 0031-3203. doi: http://dx.doi.org/10.1016/j.patcog.2010.08.023. http://www.sciencedirect.com/science/article/pii/S0031320310004097 .

Apêndice A

Lema, Teorema e Corolários

As equações (A.1) e (A.2) apresentam duas variáveis aleatórias de Bernoulli X_{ij} e Y_{ij}. X_{ij} é igual a 1 se, e somente se, o mínimo da i-ésima permutação de A está na interseção das células indexadas por j em A e B enquanto que Y_{ij} é igual a 1 se, e somente se, o máximo da i-ésima permutação de A está na interseção de células indexadas por j em A e B.

$$X_{ij} = \begin{cases} 1 & \text{se } min(\pi_i(A)) \in c_j(\pi_i(A)) \cap c_j(\pi_i(B)) \\ 0 & \text{caso contrário} \end{cases} \tag{A.1}$$

$$Y_{ij} = \begin{cases} 1 & \text{se } max(\pi_i(A)) \in c_j(\pi_i(A)) \cap c_j(\pi_i(B)) \\ 0 & \text{caso contrário} \end{cases} \tag{A.2}$$

Demonstração. **Prova do Lema 1**.

A Esperança da variável aleatória X'_{ij} é definida como:

$$\mathbb{E}[X'_{ij}] = \Pr(X'_{ij} = 0) \times 0 + \Pr(X'_{ij} = 1) \times 1$$
$$\implies \mathbb{E}[X'_{ij}] = \Pr(X'_{ij} = 1).$$

Além disso, da regra de Bayes, $\Pr(X'_{ij} = 1)$ é estimada como:

$$\frac{\Pr(X'_{ij} = 1 \,|\, X_i = 1)}{\Pr(X'_{ij} = 1)} = \frac{\Pr(X_i = 1 \,|\, X'_{ij} = 1)}{\Pr(X_i = 1)}$$
$$\implies \Pr(X'_{ij} = 1) = \frac{\Pr(X'_{ij} = 1 \,|\, X_i = 1)}{\Pr(X_i = 1 \,|\, X'_{ij} = 1)} \times \Pr(X_i = 1)$$
$$\implies \Pr(X'_{ij} = 1) = \Pr(X'_{ij} = 1 \,|\, X_i = 1) \times \Pr(X_i = 1)$$
$$\overset{eq.(4.2)}{\implies} \Pr(X'_{ij} = 1) = \Pr(X'_{ij} = 1 \,|\, X_i = 1) \times J(A, B)$$

Da Lei da probabilidade total (LPT):

$$\begin{aligned}
&\Pr(X'_{ij} = 1 \,|\, X_i = 1) = \Pr(X'_{ij} = 1 \,|\, X_i = 1 \cap M_{ij} = 1)\times \\
&\Pr(M_{ij} = 1 \,|\, X_i = 1) + \Pr(X'_{ij} = 1 \,|\, X_i = 1 \cap M_{ij} = 0)\times \\
&\Pr(M_{ij} = 0 \,|\, X_i = 1) \\
&= \Pr(X'_{ij} = 1 \,|\, X_i = 1 \cap M_{ij} = 1) \times \frac{1}{p} + \\
&\quad \Pr(X'_{ij} = 1 \,|\, X_i = 1 \cap M_{ij} = 0) \times (1 - \frac{1}{p}) \\
&\overset{eq.(A.1)}{\implies} \Pr(X'_{ij} = 1 \,|\, X_i = 1) = 1 \times \frac{1}{p} + \\
&\quad \Pr(X'_{ij} = 1 \,|\, X_i = 1 \cap M_{ij} = 0) \times (1 - \frac{1}{p})
\end{aligned}$$

Dado que $\{X'_{ij}, X'_{2j}, \ldots, X'_{pj}\}$ são as variáveis aleatórias de Bernoulli p, existem 2^p combinações para X'_{ij} quando $X_i = 1$ e $M_{ij} = 0$, e $2^{(p-1)}$ quando $X'_{ij} = 1$. Assim:

$$\Pr(X'_{ij} = 1 \,|\, X_i = 1 \cap M_{ij} = 0) = \frac{2^{(p-1)}}{2^p} = \frac{1}{2}$$
$$\implies \Pr(X'_{ij} = 1 \,|\, X_i = 1) = \frac{1}{p} + \frac{1}{2} \times (1 - \frac{1}{p}) = \frac{1}{2} \times (1 + \frac{1}{p})$$
$$\implies \Pr(X'_{ij} = 1) = \mathbb{E}[X'_{ij}] = \frac{1}{2} \times (1 + \frac{1}{p}) \times J(A, B)$$
$$\implies \Pr(X'_{ij} = 1) = \mathbb{E}[X'_{ij}] = (\frac{1}{2} + \frac{1}{2p}) \times J(A, B)$$

□ □

Demonstração. **Prova do Lema 2** é similar a prova do lema 1 ao substituir X'_{ij}, X_i e M_{ij} por Y'_{ij}, Y_i e N_{ij}, respectivamente. □

Demonstração. **Prova do Lema 3**. Da regra de Bayes:

$$\Pr(Y'_{ij} = 0 \cap X'_{ij} = 0) = \Pr(Y'_{ij} = 0 \,|\, X'_{ij} = 0) \times \Pr(X'_{ij} = 0)$$
$$\implies \Pr(Y'_{ij} = 0 \,|\, X'_{ij} = 0) = \frac{\Pr(Y'_{ij} = 0 \cap X'_{ij} = 0)}{\Pr(X'_{ij} = 0)}$$

X'_{ij} e X'_{ij} são independentes e, portanto:

$$\Pr(Y'_{ij} = 0 \,|\, X'_{ij} = 0) = \frac{\Pr(Y'_{ij} = 0) \times \Pr(X'_{ij} = 0)}{\Pr(X'_{ij} = 0)}$$
$$\implies \Pr(Y'_{ij} = 0 \,|\, X'_{ij} = 0) = \Pr(Y'_{ij} = 0)$$

□ □

Demonstração. **Prova do Teorema 1**. Da regra de Bayes:

$$\frac{\Pr(Z' = 0 \,|\, X'_{ij} = 0)}{\Pr(Z' = 0)} = \frac{\Pr(X'_{ij} = 0 \,|\, Z' = 0)}{\Pr(X'_{ij} = 0)}$$
$$\implies \Pr(Z' = 0) = \frac{\Pr(Z' = 0 \,|\, X'_{ij} = 0) \times \Pr(X'_{ij} = 0)}{\Pr(X'_{ij} = 0 \,|\, Z' = 0)}$$

$Z' = 0$ se, e somente se, $X'_{ij} = 0$ e $Y'_{ij} = 0$ para todas permutações e células e, assim sendo, $\Pr(X'_{ij} = 0 \,|\, Z' = 0) = 1$ para eles. Portanto:

$$\Pr(Z' = 0) = \Pr(Z' = 0 \,|\, X'_{ij} = 0) \times \Pr(X'_{ij} = 0)$$

Da Lei da Probabilidade Total (LPT):

$$\begin{aligned}
&\Pr(Z' = 0 \,|\, X'_{ij} = 0) = \Pr(Z' = 0 \,|\, X'_{ij} = 0 \cap Y'_{ij} = 0) \times \\
&\Pr(Y'_{ij} = 0 \,|\, X'_{ij} = 0) + \Pr(Z' = 0 \,|\, X'_{ij} = 0 \cap Y'_{ij} = 1) \times \\
&\Pr(Y'_{ij} = 1 \,|\, X'_{ij} = 0)
\end{aligned}$$

$Y'_{ij} = 1 \iff Z' \neq 0 \implies \Pr(Z' = 0 \,|\, X'_{ij} = 0 \cap Y'_{ij} = 1) = 0$ e, assim sendo:

$$\begin{aligned}
&\Pr(Z' = 0 \,|\, X'_{ij} = 0) = \Pr(Z' = 0 \,|\, X'_{ij} = 0 \cap Y'_{ij} = 0) \times \\
&\Pr(Y'_{ij} = 0 \,|\, X'_{ij} = 0) \\
&\quad \implies \Pr(Z' = 0) = \Pr(Z' = 0 \,|\, X'_{ij} = 0 \cap Y'_{ij} = 0) \times \\
&\Pr(Y'_{ij} = 0 \,|\, X'_{ij} = 0) \times \Pr(X'_{ij} = 0) \\
&\overset{lemma3}{\implies} \Pr(Z' = 0) = \Pr(Z' = 0 \,|\, X'_{ij} = 0 \cap Y'_{ij} = 0) \times \\
&\qquad \Pr(Y'_{ij} = 0) \times \Pr(X'_{ij} = 0)
\end{aligned}$$

A i-ésima permutação, com p células, tem p variáveis aleatórias de Bernoulli X'_{ij} e p variáveis aleatórias de Bernoulli Y'_{ij} tal como $1 \leq j \leq p$. Além disso, o tamanho do espaço amostral é $|\Omega| = 2^{p-1}$ quando a j-ésima célula tem $X'_{ij} = 0$ e o tamanho do espaço amostral é $|\Omega| = 2^{p-1} \times 2^{p-1}$ quando a j-ésima célula tem $X'_{ij} = 0$ e $Y'_{ij} = 0$. Assim sendo, para $\frac{\lfloor \frac{k}{p} \rfloor}{2}$ permutações:

$$\Pr(Z' = 0 \,|\, X'_{ij} = 0 \cap Y'_{ij} = 0) = \frac{1}{\left(2^{p-1} \times 2^{p-1}\right)^{\frac{\lfloor \frac{k}{p} \rfloor}{2}}} = \left(\frac{1}{2^{p-1}}\right)^{\lfloor \frac{k}{p} \rfloor}$$

$$\implies \Pr(Z' = 0) = \left(\frac{1}{2^{p-1}}\right)^{\lfloor \frac{k}{p} \rfloor} \times \Pr(Y'_{ij} = 0) \times \Pr(X'_{ij} = 0)$$

$$\Pr(Z' = 0) = \left(\frac{1}{2^{p-1}}\right)^{\lfloor \frac{k}{p} \rfloor} \times (1 - \Pr(Y'_{ij} = 1) \times (1 - \Pr(X'_{ij} = 1))$$

$$\overset{lemma2}{\implies} \Pr(Z' = 0) = \left(\frac{1}{2^{p-1}}\right)^{\lfloor \frac{k}{p} \rfloor} \times (1 - \Pr(X'_{ij} = 1))^2$$

$$\overset{lemma1}{\implies} \Pr(Z' = 0) = \left(\frac{1}{2^{p-1}}\right)^{\lfloor \frac{k}{p} \rfloor} \times \left(1 - \frac{1}{2}(1 + \frac{1}{p}) \times J(A, B)\right)^2$$

□ □

Demonstração. **Prova do Corolário 1.1**.

$$p > k \implies \left\lfloor \frac{k}{p} \right\rfloor = 0 \overset{Theo.1}{\implies} \Pr(Z' = 0) = \frac{\left(1 - \frac{1}{2}(1 + \frac{1}{p}) \times J(A, B)\right)^2}{(2^{p-1})^0}$$

$$\implies \Pr(Z' = 0) = \left(1 - \frac{1}{2}(1 + \frac{1}{p}) \times J(A, B)\right)^2$$

□ □

Demonstração. **Prova do Corolário 1.2**.

$$\Pr(Z' > 0) = 1 - \Pr(Z' = 0)$$

$$\overset{theo.1}{\Longrightarrow} \Pr(Z' > 0) = 1 - \frac{\left(1 - \frac{1}{2}(1 + \frac{1}{p}) \times J(A,B)\right)^2}{(2^{p-1})^{\lfloor \frac{k}{p} \rfloor}}$$

Para obter o intervalo de valores de $\Pr(Z' > 0)$ uma análise de p valores extremos $[1, \infty)$ é necessária, isto é,

$$\lim_{p \mapsto 1} \Pr(Z' > 0) \geq \Pr(Z' > 0) > \lim_{p \mapsto \infty} \Pr(Z' > 0)$$

.

$$\overset{p=1}{\Longrightarrow} \Pr(Z' > 0) = 1 - \frac{\left(1 - \frac{1}{2}(1 + \frac{1}{1}) \times J(A,B)\right)^2}{(2^0)^{\lfloor \frac{k}{1} \rfloor}}$$

$$\Pr(Z' > 0) = 1 - (1 - J(A,B))^2$$

$$\Pr(Z' > 0) = (1 + 1 - J(A,B)) \times (1 - 1 + J(A,B))$$

$$\Pr(Z' > 0) = (2 - J(A,B)) \times J(A,B)$$

$$= 1 - (1 - J(A,B))^2$$

$$\lim_{p \mapsto \infty} \Pr(Z' > 0) = \lim_{p \mapsto \infty} \left[1 - \Pr(Z' = 0)\right]$$

$$= 1 - \lim_{p \mapsto \infty} \Pr(Z' = 0)$$

$$\overset{cor.1.1}{\Longrightarrow} \lim_{p \mapsto \infty} \Pr(Z' > 0) = 1 - \lim_{p \mapsto \infty} \left(1 - \frac{1}{2}(1 + \frac{1}{p}) \times J(A,B)\right)^2$$

$$= 1 - \left(1 - \frac{J(A,B)}{2}\right)^2 = J(A,B) - \left(\frac{J(A,B)}{2}\right)^2$$

$$= 1 - \left(1 - \frac{J(A,B)}{2}\right)^2$$

e, assim:

$$\lim_{p \mapsto 1} \Pr(Z' > 0) \geq \Pr(Z' > 0) > \lim_{p \mapsto \infty} \Pr(Z' > 0)$$
$$= 1 - (1 - J(A, B))^2 \geq \Pr(Z' > 0) > 1 - \left(1 - \frac{J(A, B)}{2}\right)^2$$

□ □

Apêndice B

Resultados Complementares

Tabela B.1 – Vazão (documentos por hora) de RaC e RaE para indexar documentos do PAN-11

k	RAC$^{2\ \text{cells}}$	RAE$^{2\ \text{cells}}$	RAC$^{4\ \text{cells}}$	RAE$^{4\ \text{cells}}$	RAC$^{8\ \text{cells}}$	RAE$^{8\ \text{cells}}$
48	1077.52	1075.35	2109.55	2119.97	4154.46	4183.40
96	533.67	537.70	1069.39	1079.22	2117.17	2119.61
192	270.90	268.99	538.87	538.66	1071.70	1066.02
384	134.96	135.58	269.37	270.51	538.57	532.64
768	68.08	67.69	135.34	133.71	269.32	267.78

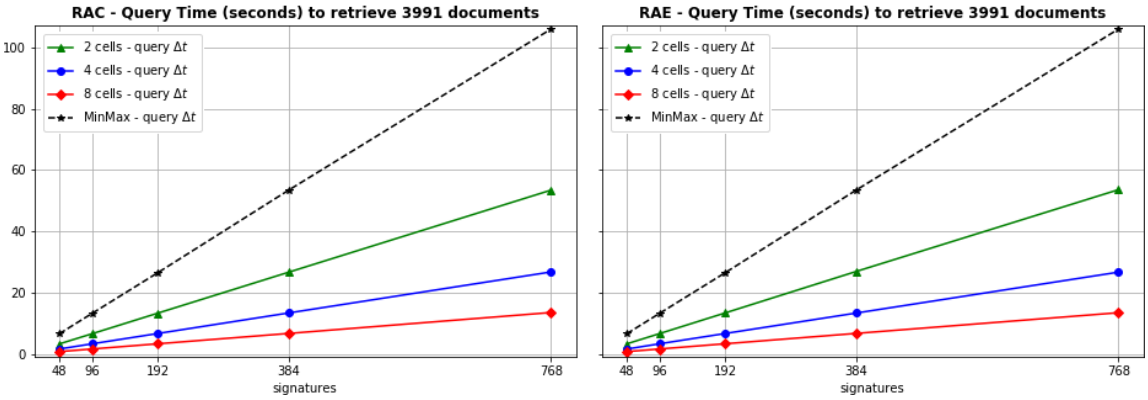


Figura B.1 – Tempo total de busca para RaC, RaE e Minmax para selecionar 3991 documentos.

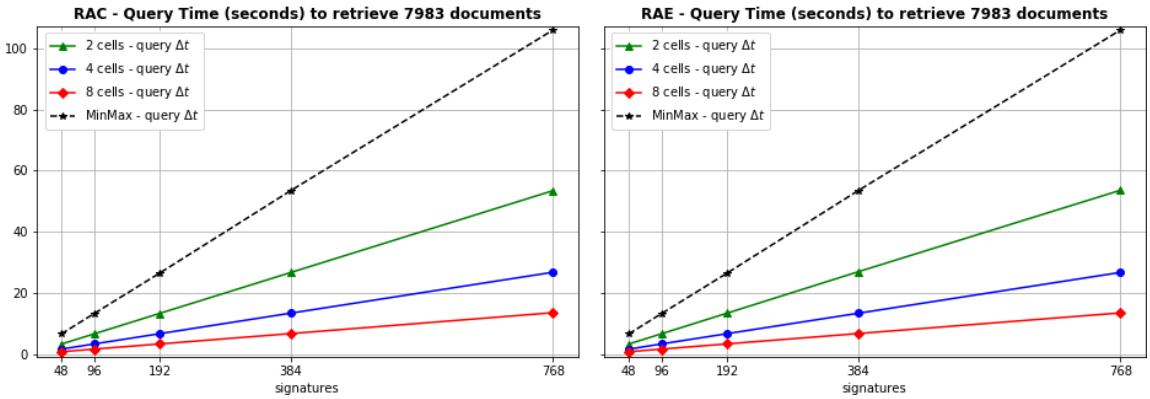


Figura B.2 – Tempo total de busca para RaC, RaE e Minmax para selecionar 7983 documentos.

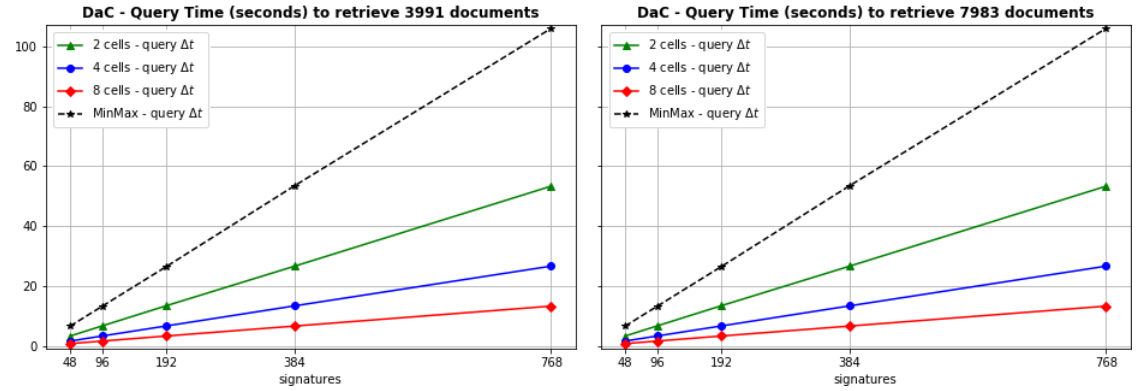


Figura B.3 – Tempo total de busca para DaC selecionar 3991 e 7983 documentos.

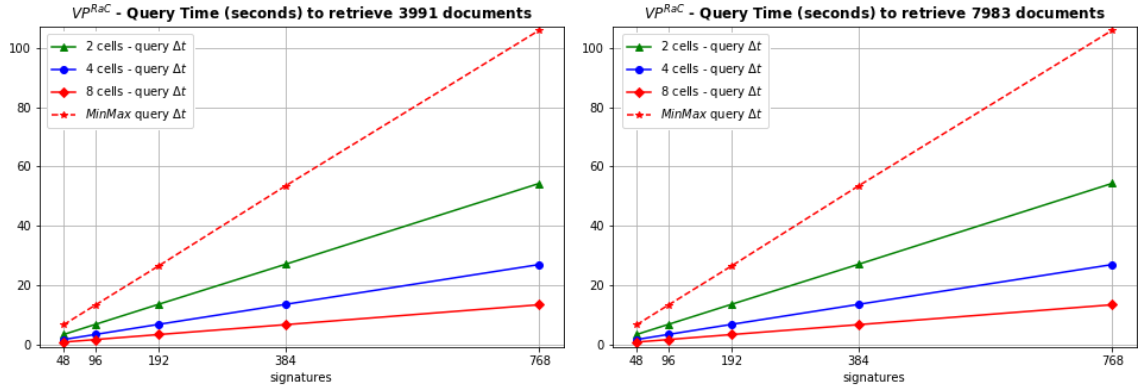


Figura B.4 – Tempo total de busca de VP^{RaC} para selecionar 3991 e 7983 documentos.

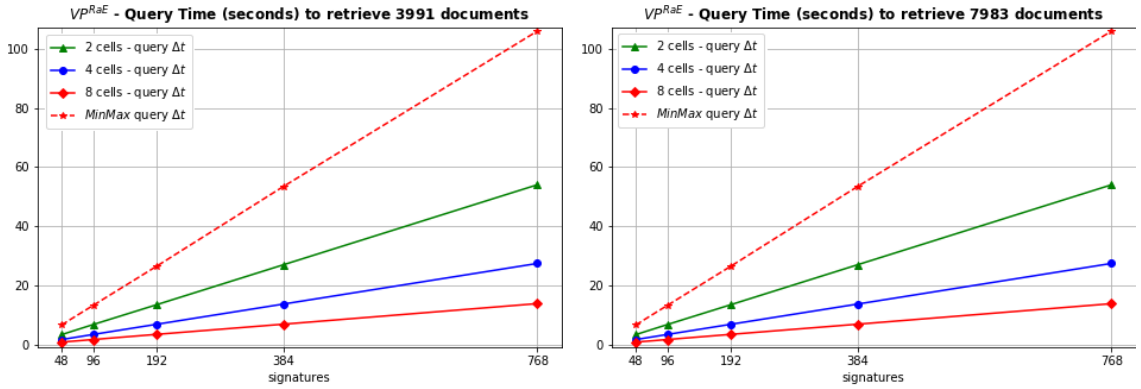


Figura B.5 – Tempo total de busca de VP^{RaE} para selecionar 3991 e 7983 documentos.

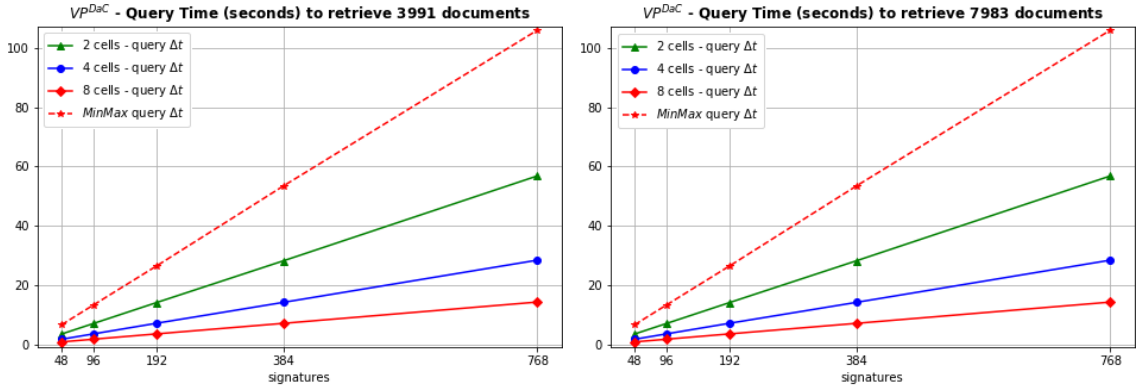


Figura B.6 – Tempo total de busca de VP^{DaC} para selecionar 3991 e 7983 documentos.

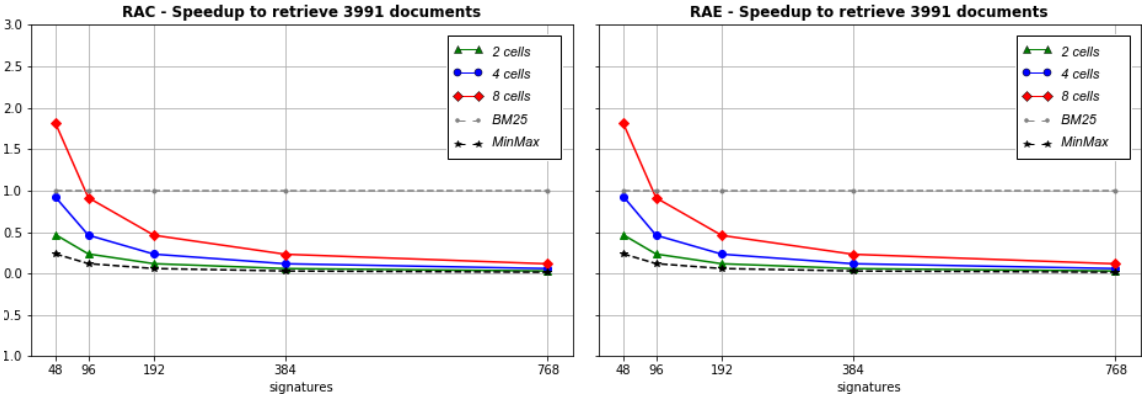


Figura B.7 – Speedup de RaC, RaE, Minmax com relação ao tempo de busca do BM25 para 3991 documentos.

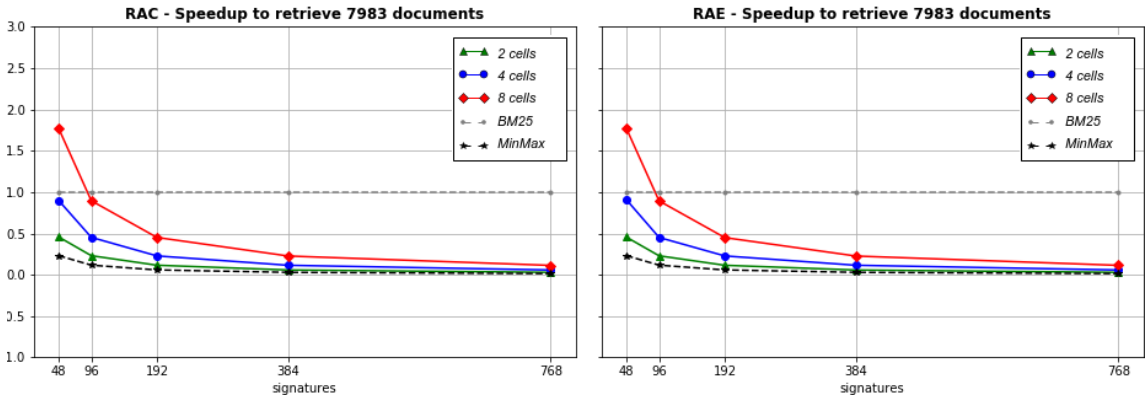


Figura B.8 – Speedup de RaC, RaE, Minmax com relação ao tempo de busca do BM25 para 7983 documentos.

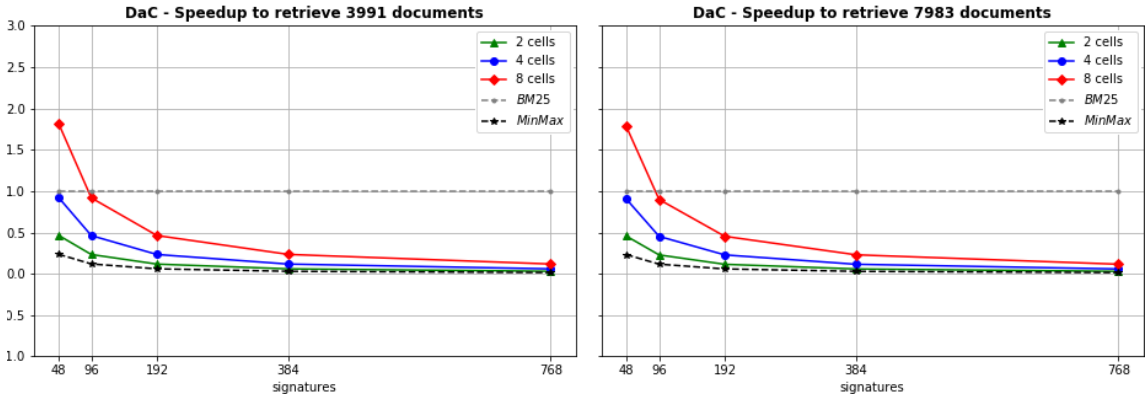


Figura B.9 – Speedup de DaC com relação ao tempo de busca do BM25 para 3991 e 7983 documentos.

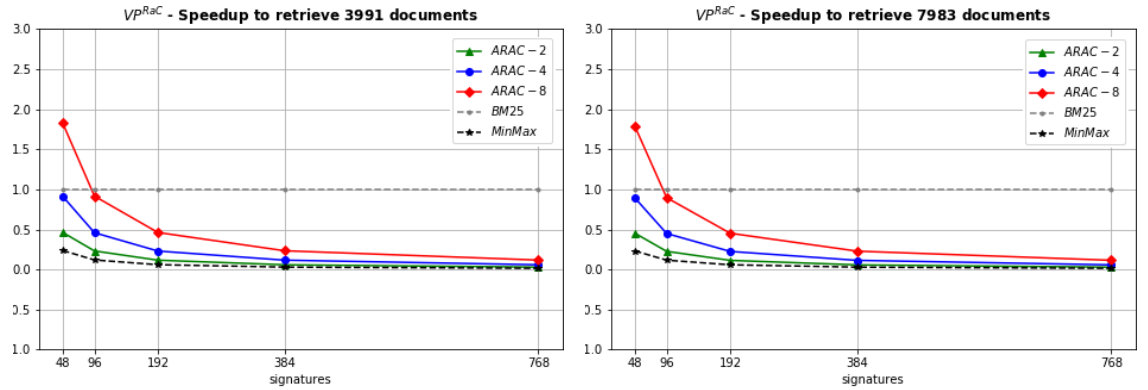


Figura B.10 – Speedup de VP^{RaC} e Minmax com relação ao tempo de busca do BM25 para 3991 e 7983 documentos.

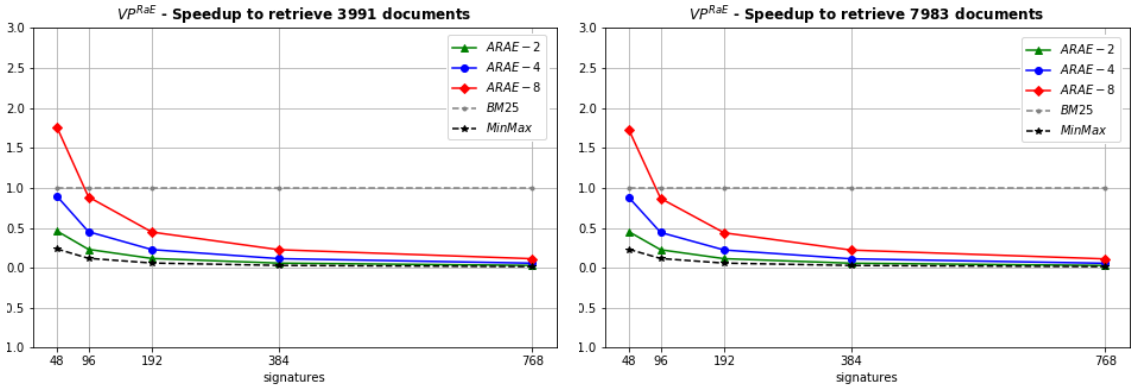


Figura B.11 – Speedup de VP^{RaE} e Minmax com relação ao tempo de busca do BM25 para 3991 e 7983 documentos.

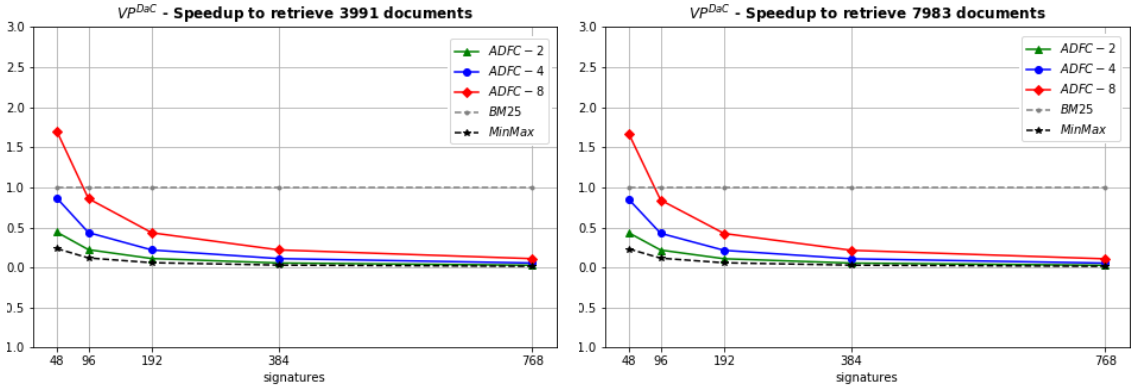


Figura B.12 – Speedup de VP^{DaC} e Minmax com relação ao tempo de busca do BM25 para 3991 e 7983 documentos.

Tabela B.2 – MAE do recall de RaC e RaE com relação ao Minmax

	$Minmax$	MAE^{RaC}			MAE^{RaE}		
k	*Recall*	2	4	8	2	4	8
			1597				
48	0.26	0.00	0.00	0.01	0.00	0.01	0.01
96	0.30	0.01	0.01	0.02	0.01	0.01	0.02
192	0.32	0.00	0.02	0.03	0.00	0.01	0.03
384	0.34	0.00	0.01	0.03	0.00	0.01	0.03
768	0.35	0.00	0.01	0.02	0.00	0.01	0.02
			3991				
48	0.43	0.00	0.02	0.00	0.00	0.01	0.00
96	0.46	0.01	0.02	0.02	0.01	0.02	0.02
192	0.48	0.00	0.02	0.03	0.00	0.02	0.03
384	0.48	0.00	0.00	0.03	0.00	0.01	0.02
768	0.49	0.00	0.00	0.02	0.00	0.00	0.02
			7983				
48	0.66	0.01	0.01	0.01	0.01	0.02	0.00
96	0.68	0.01	0.02	0.02	0.01	0.02	0.02
192	0.69	-0.01	0.01	0.01	-0.01	0.01	0.02
384	0.70	0.00	0.00	0.02	0.00	0.00	0.02
768	0.69	-0.01	0.00	0.01	-0.01	0.00	0.01
			11975				
48	0.85	0.00	0.01	0.01	0.00	0.00	0.01
96	0.86	0.00	0.00	0.01	0.00	0.01	0.01
192	0.87	0.00	0.00	0.01	0.00	0.00	0.01
384	0.87	0.00	0.00	0.01	0.00	0.00	0.01
768	0.88	0.00	0.00	0.00	0.00	0.00	0.00

Tabela B.3 – MAE do recall de RaC, RaE e Minmax com relação ao BM25

	$BM25$	$Minmax$	MAE^{RaC}			MAE^{RaE}		
k	*Recall*	MAE	2	4	8	2	4	8
1597								
48	0.38	0.11	0.11	0.12	0.12	0.12	0.12	0.12
96	0.38	0.08	0.08	0.09	0.10	0.08	0.09	0.10
192	0.38	0.05	0.05	0.07	0.08	0.05	0.07	0.08
384	0.38	0.04	0.04	0.05	0.07	0.04	0.05	0.07
768	0.38	0.03	0.03	0.04	0.05	0.03	0.04	0.05
3991								
48	0.53	0.09	0.09	0.11	0.10	0.10	0.11	0.10
96	0.53	0.06	0.07	0.08	0.08	0.07	0.08	0.09
192	0.53	0.04	0.05	0.06	0.07	0.05	0.06	0.07
384	0.53	0.04	0.04	0.05	0.07	0.04	0.05	0.07
768	0.53	0.04	0.04	0.05	0.06	0.04	0.04	0.06
7983								
48	0.83	0.17	0.18	0.19	0.18	0.18	0.19	0.18
96	0.83	0.15	0.16	0.17	0.16	0.16	0.17	0.16
192	0.83	0.14	0.14	0.15	0.16	0.13	0.15	0.16
384	0.83	0.14	0.13	0.14	0.15	0.13	0.14	0.15
768	0.83	0.14	0.13	0.14	0.15	0.13	0.14	0.15
11975								
48	0.95	0.11	0.11	0.11	0.11	0.11	0.11	0.11
96	0.95	0.09	0.09	0.09	0.10	0.09	0.10	0.10
192	0.95	0.08	0.08	0.08	0.09	0.08	0.08	0.09
384	0.95	0.08	0.08	0.08	0.09	0.08	0.08	0.09
768	0.95	0.08	0.08	0.08	0.08	0.08	0.08	0.08

Tabela B.4 – MAE do recall de DaC com relação ao Minmax

	$Minmax$	MAE^{DaC}		
k	*Recall*	2	4	8
1597				
48	0.26	0.05	0.09	0.12
96	0.30	0.06	0.10	0.13
192	0.32	0.05	0.09	0.13
384	0.34	0.04	0.08	0.12
768	0.35	0.03	0.07	0.10
3991				
48	0.43	0.05	0.08	0.12
96	0.46	0.05	0.08	0.12
192	0.48	0.03	0.07	0.11
384	0.48	0.02	0.05	0.09
768	0.49	0.01	0.03	0.05
7983				
48	0.66	0.05	0.10	0.17
96	0.68	0.03	0.07	0.13
192	0.69	0.01	0.05	0.08
384	0.70	0.01	0.02	0.06
768	0.69	0.01	0.02	0.03
11975				
48	0.85	0.04	0.08	0.17
96	0.86	0.02	0.05	0.11
192	0.87	0.01	0.03	0.06
384	0.87	0.00	0.01	0.03
768	0.88	0.00	0.00	0.01

Tabela B.5 – MAE do recall de DaC e Minmax com relação ao BM25

	$BM25$	$Minmax$	MAE^{DaC}		
k	*Recall*	MAE	2	4	8
1597					
48	0.38	0.16	0.20	0.23	0.11
96	0.38	0.14	0.17	0.21	0.08
192	0.38	0.10	0.15	0.19	0.05
384	0.38	0.08	0.12	0.16	0.04
768	0.38	0.05	0.09	0.13	0.03
3991					
48	0.53	0.14	0.17	0.21	0.09
96	0.53	0.11	0.15	0.18	0.06
192	0.53	0.08	0.12	0.16	0.04
384	0.53	0.06	0.09	0.13	0.04
768	0.53	0.05	0.07	0.09	0.04
7983					
48	0.83	0.22	0.27	0.34	0.17
96	0.83	0.17	0.22	0.28	0.15
192	0.83	0.15	0.19	0.23	0.14
384	0.83	0.14	0.16	0.19	0.14
768	0.83	0.14	0.15	0.17	0.14
11975					
48	0.95	0.14	0.19	0.27	0.11
96	0.95	0.11	0.14	0.20	0.09
192	0.95	0.09	0.11	0.15	0.08
384	0.95	0.08	0.09	0.11	0.08
768	0.95	0.08	0.08	0.09	0.08

Tabela B.6 – MAE do recall de VP^{DaC}, VP^{RaC}, VP^{RaE} com relação ao Minmax

k	$MinMax$	$Mae^{VP^{DaC}}$			$Mae^{VP^{RaC}}$			$Mae^{VP^{RaE}}$		
		2	4	8	2	4	8	2	4	8
1597										
48	0.26	0.00	0.01	0.00	0.00	0.03	0.04	0.00	0.00	-0.01
96	0.30	0.00	0.01	0.02	0.00	0.03	0.06	0.00	0.01	0.01
192	0.32	0.00	0.01	0.01	0.00	0.02	0.05	0.00	0.00	0.00
384	0.34	0.00	0.00	0.01	0.00	0.01	0.04	0.00	-0.01	0.00
768	0.35	-0.01	-0.01	0.00	0.00	0.00	0.02	0.00	-0.01	0.00
3991										
48	0.43	0.01	0.02	0.00	0.00	0.03	0.04	0.00	0.01	-0.01
96	0.46	0.01	0.01	0.02	0.00	0.03	0.05	0.00	0.01	0.01
192	0.48	0.00	0.02	0.02	0.00	0.02	0.04	0.00	0.01	0.01
384	0.48	0.00	-0.01	0.01	-0.01	0.01	0.02	-0.01	0.00	0.01
768	0.49	0.00	-0.01	0.00	-0.01	0.00	0.00	-0.01	-0.01	-0.01
7983										
48	0.66	0.01	0.01	0.00	0.00	0.04	0.08	0.00	0.01	0.00
96	0.68	0.01	0.01	0.02	0.01	0.03	0.05	0.01	0.01	0.02
192	0.69	-0.01	0.00	0.01	-0.01	0.01	0.03	-0.01	0.01	0.00
384	0.70	0.00	0.00	0.01	-0.01	0.00	0.01	-0.01	0.00	0.01
768	0.69	0.00	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	0.00
11975										
48	0.85	0.01	0.00	0.00	0.00	0.03	0.07	0.00	0.00	0.00
96	0.86	0.00	0.00	0.00	0.00	0.02	0.03	0.00	0.00	0.01
192	0.87	0.00	0.00	0.00	0.00	0.01	0.02	0.00	0.00	0.01
384	0.87	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00
768	0.88	0.00	0.00	-0.01	0.00	0.00	0.00	0.00	0.00	0.00

Tabela B.7 – MAE do recall de VP^{DaC}, VP^{RaC}, VP^{RaE} e Minmax com relação ao BM25

k	$BM25$	$MinMax$	$Mae^{VP^{DaC}}$			$Mae^{VP^{RaC}}$			$Mae^{VP^{RaE}}$		
			2	4	8	2	4	8	2	4	8
1597											
48	0.38	0.11	0.12	0.12	0.11	0.15	0.16	0.11	0.12	0.11	0.11
96	0.38	0.08	0.09	0.09	0.08	0.11	0.14	0.08	0.09	0.08	0.08
192	0.38	0.05	0.06	0.07	0.05	0.08	0.10	0.05	0.06	0.06	0.05
384	0.38	0.03	0.03	0.05	0.03	0.05	0.07	0.03	0.03	0.04	0.04
768	0.38	0.02	0.02	0.02	0.03	0.03	0.05	0.03	0.02	0.02	0.03
3991											
48	0.53	0.10	0.11	0.09	0.09	0.12	0.13	0.09	0.10	0.09	0.09
96	0.53	0.07	0.07	0.08	0.07	0.09	0.11	0.07	0.08	0.07	0.06
192	0.53	0.05	0.06	0.07	0.05	0.06	0.08	0.05	0.06	0.06	0.04
384	0.53	0.04	0.04	0.05	0.03	0.05	0.06	0.03	0.04	0.05	0.04
768	0.53	0.04	0.03	0.04	0.04	0.04	0.04	0.04	0.03	0.04	0.04
7983											
48	0.83	0.18	0.18	0.18	0.18	0.21	0.26	0.18	0.18	0.18	0.17
96	0.83	0.16	0.16	0.16	0.16	0.18	0.20	0.16	0.16	0.16	0.15
192	0.83	0.13	0.15	0.15	0.14	0.15	0.17	0.14	0.15	0.14	0.14
384	0.83	0.13	0.13	0.14	0.13	0.14	0.15	0.13	0.13	0.14	0.14
768	0.83	0.14	0.13	0.13	0.13	0.13	0.13	0.13	0.13	0.13	0.14
11975											
48	0.95	0.11	0.11	0.11	0.11	0.13	0.18	0.11	0.11	0.11	0.11
96	0.95	0.09	0.09	0.09	0.09	0.11	0.12	0.09	0.09	0.10	0.09
192	0.95	0.08	0.09	0.08	0.08	0.09	0.10	0.08	0.08	0.09	0.08
384	0.95	0.08	0.08	0.08	0.08	0.08	0.09	0.08	0.08	0.08	0.08
768	0.95	0.08	0.08	0.07	0.08	0.08	0.08	0.08	0.07	0.08	0.08