UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

JONATHAN SILVA MARTINS

ACERPI-Block: aplicação de técnicas de blocagem à abordagem ACERPI

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação

Orientador: Prof[a]. Dr[a]. Renata Galante
Co-orientador: Christian Schmitz

Porto Alegre
2021

AGRADECIMENTOS

O primeiro e mais importante agradecimento é para minha família. Mais especificamente meu pai, José Pereira Martins, e minha mãe, Elaine Jaqueline Santos da Silva, que sempre me apoiaram, me motivaram e me ensinaram as coisas mais importantes da vida. Sem o esforço incansável deles nenhuma das minhas conquistas seria possível. Obrigado por serem a maior sorte que eu tive nessa vida

Agradeço também a minha namorada, Stefany Costa Nascimento, pelo amor, carinho e apoio que tornam todos os desafios da vida menos assustadores.

Agradeço também por todo o apoio, incentivo e ajuda provido à mim por André Kissmann e Marcelo Benedeti Palermo. Vocês fizeram o desafio de escrever este trabalho em um semestre e, ao mesmo tempo, trabalhar em tempo integral menos árduo e assustador.

Agradeço, também, aos professores que tive durante minha jornada na graduação da UFRGS que me ensinaram muito sobre diversos tópicos diferentes da computação e da vida. Mesmo aqueles que às vezes faziam a jornada mais difícil, me ensinaram lições de resiliência e trabalho duro.

Agradeço especialmente à professora Renata de Matos Galante, orientadora deste trabalho, que demonstrou todos os dias empatia e preocupação não só comigo mas com todos os seus alunos e fez deste trabalho uma experiência agradável e menos assustadora, também à Christian Schmitz, que esteve sempre presente e cujos conselhos, apoio e ensinamentos ajudaram a elevar a qualidade deste trabalho, bem como ao professor Edimar Manica, que sempre trouxe sugestões construtivas e demonstrações de educação e respeito.

Muito obrigado.

RESUMO

Resolução de entidades é uma etapa importante de diversas aplicações. Ela aparece tanto como parte essencial quanto como uma etapa de limpeza de dados para aumentar a eficiência ou eficácia dessas aplicações. Resolução de entidades é uma etapa essencial da abordagem ACERPI, que foi proposta com objetivo de gerar um banco de dados NoSQL com informações disponibilizadas por instituições públicas através de portarias para possibilitar consultas complexas nesses dados. A abordagem ACERPI é ainda bem nova e apresenta espaço para otimizações. Uma das áreas que apresenta espaço para otimização é o algoritmo de resolução de entidades utilizado. Neste trabalho, são exploradas técnicas de blocagem como formas de otimização da eficiência do algoritmo de resolução de entidades da abordagem ACERPI. O trabalho aqui proposto é denominado ACERPI-Block, um acrônimo para aplicação de técnicas de blocagem à abordagem ACERPI. Para demonstrar e analisar a aplicação dessas técnicas, são realizados múltiplos experimentos e coletadas diversas métricas para análise. Experimentos demonstraram que a eficácia das técnicas de blocagem na eficiência dos algoritmos de resolução de entidades é considerável, e que quanto maior o impacto dessas técnicas, mais a eficiência do algoritmo depende de uma utilização otimizada dos recursos de CPU. Experimentos também demonstraram como técnicas de blocagem podem servir como filtro de ruído para algoritmos de comparação demasiado permissivos, aumentando a qualidade dos seus resultados.

Palavras-chave: Otimização. blocagem. resolução de entidades.

Application of blocking techniques as optimization for ACERPI's entity resolution algorithm

ABSTRACT

Entity resolution is an important part of many applications. It can be anything from an essential part to a step of data cleanse that improves efficiency or effectiveness of these applications. The ACERPI approach to generate a NoSQL database with information made available by public institutions in the form of *portarias* to enable complex queries on this data has entity resolution as an essential part of its process. The ACERPI approach is still quite new and presents opportunities for optimizations. One of the areas that presents opportunities for optimization is the entity resolution algorithm. In this text, multiple blocking techniques will be explored as options to optimize the efficiency of the ACERPI's entity resolution algorithm. To demonstrate and analyse the results of the application of these techniques, multiple experiments will be performed and metrics will be collected for analysis. Experiments showed that the effectiveness of blocking techniques is considerable, and that the bigger the impact of these techniques, the more the efficiency of the algorithm depends of an optimized use of the CPU resources. Experiments also showed how blocking techniques can be used as a noise filter for matching algorithms that are too permissive, increasing the quality of its results.

Keywords: optimization, blocking, entity resolution.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CPU	*Central Processing Unit* - Unidade central de processamento
GCP	*Google Cloud Platform* - Plataforma de nuvem da Google
PDF	*Portable Document Format* - Formato Portátil de Documento
XML	*Extensible Markup Language* - Linguagem de Marcação Extensível

SUMÁRIO

1 INTRODUÇÃO

Resolução de entidades e deduplicação de dados são partes importantes, às vezes essenciais, de diversas aplicações atuais. Resolução de entidades consiste na identificação e agrupamento entre si de elementos em um conjunto de dados que representam uma mesma entidade do mundo real (PAPADAKIS et al., 2019). Uma aplicação importante desses algoritmos é limpeza de dados para aplicações de aprendizado de máquina (HERNÁNDEZ; STOLFO, 1998), na qual a duplicação de dados pode causar desde problemas de performance, por estar processando dados duplicados várias vezes, até problemas de eficácia, por ter dados duplicados que geram resultados tendenciosos no algoritmo.

Resolução de entidades também é uma parte essencial da abordagem ACERPI (SCHMITZ et al., 2021), que tem como objetivo a criação de um banco de dados que permita consultas complexas sobre servidores públicos baseados em informações disponibilizadas pelas instituições públicas através de portarias. Resumidamente, a abordagem ACERPI consiste de três etapas: coleta dos documentos; extração de informação; e resolução de entidade. A etapa de coleta coleta as portarias, no formato PDF, dos repositórios e converte seu conteúdo para o formato XML. A etapa de extração de informação usa técnicas de identificação de entidades nomeadas para reconhecer os servidores públicos em uma determinada portaria, no format XML, e guarda as informações sobre a referência identificada em um banco de dados NoSQL. Por último, a etapa de resolução de entidades, que é o foco do ACERPI-Block, vai usar as informações sobre as referências para identificar e agrupar essas referências que se referem a uma entidade do mundo real comum.

O algoritmo usado para resolução de entidades na abordagem ACERPI produz bons resultados em termos de eficácia, porém ainda não tem uma eficiência suficientemente alta para que possa ser considerada uma solução prática para grandes volumes de dados. Isso não é, de maneira alguma, uma falha do autor da abordagem, mas uma consequência do escopo do trabalho que originou a abordagem ACERPI, que tinha como objetivo a criação do banco de dados NoSQL que permita as consultas complexas aos dados disponibilizados na forma de portarias, e não a otimização deste processo.

O objetivo deste trabalho é explorar a aplicação de técnicas clássicas de otimização de algoritmos de resolução de entidades, blocagem e filtragem, ao algoritmo de resolução de entidades usado na abordagem ACERPI, explorando os efeitos dessas técnicas na eficiência e eficácia do algoritmo através de diversas métricas sobre a execução do programa

de resolução de entidades. O trabalho aqui proposto é denominado ACERPI-Block, um acrônimo para aplicação de técnicas de blocagem à abordagem ACERPI. A análise dos resultados é baseada em uma série de experimentos que envolvem múltiplas execuções do programa de resolução de entidades, com diversas técnicas e combinações de técnicas diferentes de blocagem aplicadas ao algoritmo de resolução de entidades e depois a análise de métricas como tempo de execução, número de comparações, utilização da CPU e métricas de qualidade. Resultados demonstraram que a aplicação de técnicas de blocagem são mais eficazes com algoritmos que utilizem comparações mais restritivas para agrupamento de referências, nos quais o impacto na qualidade do resultado é mínimo. Também é demonstrado como técnicas de blocagem podem ser usadas como filtro de ruído para algoritmos de comparação excessivamente permissivos, além da importância do uso de um conjunto de dados de teste extenso e abrangente.

O restante deste trabalho está organizado da seguinte maneira: o Capítulo 2 aborda os conceitos, as ferramentas e as tecnologias que permitem a execução e análise das técnicas abordadas neste trabalho; o Capítulo 3 explora outros trabalhos relacionados ao tópico estudado e também descreve o funcionamento da abordagem ACERPI, utilizada como base para o desenvolvimento deste trabalho; o Capítulo 4 descreve a proposta ACERPI-Block que é uma aplicação que explora mais a fundo as técnicas aplicadas e a estrutura dos dados processados pelo algoritmo de resolução de entidades neste trabalho; o Capítulo 5 descreve os experimentos executados durante o desenvolvimento deste trabalho além da metodologia usada e a análise desses resultados; e, por fim, o Capítulo 6 revisa as conclusões e contribuições deste trabalho e apresenta sugestões de possíveis trabalhos futuros.

2 CONCEITOS E TECNOLOGIAS UTILIZADAS

Neste capítulo, são apresentados os conceitos e as tecnologias que foram usadas para realizar a resolução de entidades e as ferramentas que foram usadas para analisar a eficiência dos algoritmos usados para a resolução de entidades.

2.1 Conceitos

Nesta seção, são explorados alguns conceitos importantes para a compreensão deste trabalho.

2.1.1 Resolução de Entidades

Resolução de entidades, ou deduplicação, tem como objetivo identificar os elementos de um conjunto de dados que se referem à mesma entidade do mundo real (KÖPCKE; RAHM, 2010). Esta é uma parte essencial da abordagem ACERPI (SCHMITZ et al., 2021) que é base para este trabalho. A abordagem ACERPI apresenta um algoritmo de resolução de entidades para identificar e agrupar, em entidades, os elementos do seu banco de dados que se referem ao mesmo servidor público.

2.1.2 Blocagem

Blocagem (PAPADAKIS et al., 2019) é o nome dado ao conjunto de técnicas de otimização de resolução de entidades, ou deduplicação, que consiste em tentar identificar de maneira eficiente conjuntos, ou blocos, de elementos que têm baixa probabilidade de serem referências à mesma entidade do mundo real e descartar estes elementos, evitando a aplicação de operações de comparação custosas em elementos que provavelmente não são referências à mesma entidade do mundo real.

Neste trabalho, são exploradas técnicas de blocagem como otimizações para o algoritmo de resolução de entidades da abordagem ACERPI

2.1.3 Filtragem

Filtragem (PAPADAKIS et al., 2019), assim como blocagem, é um conjunto de técnicas de otimização de resolução de entidades, ou deduplicação. Mas, diferentemente de blocagem, filtragem descarta apenas elementos que não tem qualquer possibilidade de serem referências às mesmas entidades do mundo real. Desta maneira, filtragem evita qualquer perda de qualidade no resultado.

2.2 Tecnologias

Nesta seção são apresentadas as tecnologias utilizadas neste trabalho para possibilitar a execução e análise dos experimentos.

2.2.1 MongoDB

Bancos de dados *NoSQL* são bancos de dados não relacionais que conseguem processar grande volume de dados não estruturados e que passam por mudanças frequentes (MICROSOFT, 2021).

Existem diferentes tipos de bancos de dados não relacionais. Alguns deles são:

- Chave-valor: bancos de dados que fazem o pareamento de chaves e valores usando uma tabela de *hash*. São bastante eficientes e altamente particionáveis.
- Documento: guardam dados em coleções de documentos. Permitem consulta em qualquer atributo do documento e são bastante flexíveis.
- Grafo: usam um modelo baseado em grafos, com nós e bordas, para representar dados ou elementos interconectados.

MongoDB (MONGODB, 2020a) é um banco de dados *NoSQL* baseado em documentos (MONGODB, 2020b) estruturados em uma maneira similar à estrutura JSON (*JavaScript Object Notation*). MongoDB é usado neste trabalho para armazenar os dados usados pelos algoritmos de resolução de entidades.

2.2.2 cProfile

cProfile é um *profiler* para programas escritos em Python (PARMAR, 2021). cProfile é implementado em C para que o impacto na performance do programa seja mínimo, desta maneira permitindo que seja usado para programas longos e complexos.

Esta ferramenta permite a análise do número de chamadas para cada função em um programa Python, além da análise do tempo levado para execução dessas funções.

2.2.3 Pacote *time* do Linux

O pacote *time* do Linux é um programa que permite a coleta de dados sobre um processo, como tempo de execução, utilização de CPU, utilização de memória e número de *page faults*. Esse pacote foi utilizado neste trabalho para coletar métricas sobre o processo durante os experimentos.

2.2.4 *Google Cloud Platform*

A *Google Cloud Platform* (PERVEEZ, 2021) é uma plataforma de serviços que fornece diversos serviços, como máquinas virtuais, redes virtuais e bancos de dados. Neste trabalho são usadas somente as máquinas virtuais para executar os programas de resolução de entidades e analisar os resultados.

A *Google Cloud Platform* foi usada para executar todos os experimentos em ambientes similares entre si e para permitir a execução desses experimentos em paralelo, o que possibilitou a exploração de um número muito maior de experimentos.

2.3 Considerações Finais

Neste capítulo foram apresentados os conceitos e as ferramentas que foram usados no desenvolvimento deste trabalho para a execução e análise dos algoritmos de resolução de entidades estudados. São usadas as ferramentas, MongoDB, para armazenamento de dados; *cProfile*, para coleta de métricas de execução a nível de função; *time*, para coleta de métricas de execução a nível de processo; e GCP, para execução dos experimentos em paralelo e em um ambiente padronizado.

3 TRABALHOS RELACIONADOS

Neste capítulo, são apresentados alguns trabalhos relacionados. Primeiramente, a abordagem ACERPI Schmitz et al. (2021), que é base para este trabalho, é apresentada em detalhes. Em seguida, são discutidos os trabalhos relacionados.

3.1 ACERPI

ACERPI (SCHMITZ et al., 2021) é uma abordagem para a criação de um banco de dados NoSQL a partir das informações divulgadas por instituições públicas, na forma de portarias, para tornar possível consultas complexas às informações divulgadas como portarias. Para fazer isso, é necessário coletar as portarias dos sites das instituições públicas, extrair as informações relevantes dos documentos coletados utilizando reconhecimento de entidades nomeadas para um banco de dados e, por último, aplicar resolução de entidades para agrupar os documentos que referenciam os mesmos funcionários públicos.

A abordagem ACERPI é dividida em três etapas distintas, como descrito a seguir:

- **Coleta de dados:** A coleta de dados no trabalho ACERPI (SCHMITZ et al., 2021) é feita usando técnicas de *Web scraping* (ScrapingHub, 2020) para procurar pelos documentos relevantes nos repositórios de instituições públicas, como o repositório UFRGS (2016), e fazer o *download* e conversão desses documentos, que normalmente são disponibilizados no formato PDF, para um formato intermediário, XML, que facilita o processamento dos dados nas próximas etapas.
- **Extração de dados:** Após a coleta de dados, são aplicadas técnicas de reconhecimento de entidades e bibliotecas de processamento de linguagem natural para reconhecer os servidores públicos mencionados nas portarias. As informações extraídas são então armazenadas em um bando de dados NoSQL, neste caso MongoDB (MONGODB, 2020a), com as informações relevantes para cada entidade reconhecida. Essas informações são o documento em que a entidade foi encontrada, o nome e a SIAPE da entidade. Essas informações serão posteriormente usadas na etapa de resolução de entidades.
- **Resolução de entidades:** A última etapa da abordagem ACERPI (SCHMITZ et al., 2021) é a realização da resolução de entidades. Nesta etapa, são aplicadas técnicas de resolução de entidades por agrupamento para agrupar todas entidades

reconhecidas que referenciam a mesma entidade do mundo real. É nesta etapa que a aplicação ACERPI-block atua, aplicando técnicas de blocagem para aumentar a eficiência da etapa de resolução de entidades sem diminuir a eficácia à um nível inaceitável.

3.2 Trabalhos Relacionados

Há dois trabalhos diretamente relaciondos com o ACERPI-Block proposto neste trabalho. O primeiro trabalho, intitulado *A Survey of Blocking and Filtering Techniques for Entity Resolution* (PAPADAKIS et al., 2019), explora diversas técnicas de blocagem e filtragem, as categorizando e explorando as diferenças entre essas categorias de técnicas de blocagem. O *survey* também apresenta definições para termos importantes utilizados ao longo deste trabalho e é referenciado, assim como outros trabalhos semelhantes, em vários pontos durante este trabalho.

O trabalho intitulado *Entity Resolution Evaluation Measures* (MAIDASANI et al., 2012) agrega os resultados e discussões de múltiplos trabalhos realizados na área de avaliação de algoritmos de resolução de entidades e compila essas informações em diferentes categorias. Diversas definições foram usadas ao longo deste trabalho acompanhadas de referências ao trabalho original.

3.3 Considerações Finais

Neste capítulo, foram apresentados trabalhos relacionados e explorado o que esses trabalhos discutem e como eles se relacionam com o trabalho desenvolvido. O principal trabalho relacionado é Schmitz et al. (2021), e por isso também foi o trabalho explorado em maiores detalhes, para fornecer o contexto necessário para a proposta ACERPI-Block.

4 ACERPI-BLOCK

ACERPI-Block é o nome dado para o conjunto de variações da abordagem ACERPI (SCHMITZ et al., 2021) que usam diferentes técnicas de blocagem, ou *blocking* em inglês, com o objetivo de aumentar a eficiência do seu algoritmo de resolução de entidades.

Este capítulo apresenta as técnicas de blocagem e filtragem, e discute como elas são aplicadas ao algoritmo de resolução de entidades da abordagem ACERPI, apresentando as diferentes implementações dessas técnicas e o que é esperado dos resultados da aplicação dessas técnicas.

4.1 Descrição do Problema

Neste trabalho, é usada a definição descrita no trabalho de Benjelloun et al. (2009) que define resolução de entidades como o processo de encontrar duplicatas, não necessariamente idênticas, daqui em diante chamadas de **referências**, em uma relação e agrupar estas duplicatas em uma única **entidade**. Resolução de entidades é uma parte importante de vários processos em computação, por exemplo, o processo de limpeza de dados (HERNÁNDEZ; STOLFO, 1998) que é usado para melhorar a qualidade ou eficiência de diversos algoritmos de análise de dados ou aprendizado de máquina.

Uma das etapas do processo de resolução de entidades é a seleção de referências que são possíveis duplicatas de uma certa entidade para executar as operações que vão decidir se essas referências são de fato duplicatas de uma entidade conhecida. É nesta etapa que é feita a aplicação de técnicas de blocagem para amenizar o custo do processo de resolução de entidades, que possui, por natureza, complexidade de tempo quadrática devido a necessidade de comparar todas as referências entre si. O objetivo das técnicas de blocagem é reduzir o número de referências que são comparadas entre si, usando propriedades do conjunto de dados dessas referências para conseguir determinar quais referências não são partes de uma entidade sem a necessidade de executar operações custosas de comparação.

Para analisar de maneira adequada o impacto que essas técnicas tem na eficácia do algoritmo de resolução de entidades da abordagem ACERPI, também é necessário um algoritmo para avaliação da qualidade do resultado gerado pelo algoritmo de resolução de entidades. Já existe um algoritmo que é capaz de fazer essa avaliação de qualidade para

o algoritmo de resolução de entidades da abordagem ACERPI e foi criado também para o trabalho de Schmitz et al. (2021). Este algoritmo é adequado para avaliar o resultado de uma execução do algoritmo de resolução de entidades ACERPI que usa um conjunto de dados de teste, composto de 613 referências para as quais são conhecidas as entidades do mundo real às quais cada uma se refere. Porém, esse algoritmo é incapaz de avaliar adequadamente o resultado do algoritmo de resolução de entidades quando este é aplicado a um conjunto maior de dados, por limitações relacionadas à eficiência do algoritmo, ou à um conjunto de dados dos quais só são conhecidas as entidades do mundo real para um subconjunto do conjunto de dados completo, por limitações do *design* do algoritmo. Para poder avaliar adequadamente os resultados, é proposto, também neste trabalho, um novo algoritmo de avaliação que soluciona essas limitações.

O objetivo deste trabalho é analisar o impacto dessas técnicas no algoritmo de resolução de entidades da abordagem ACERPI (SCHMITZ et al., 2021) e, como parte desse objetivo, é apresentado um algoritmo eficiente para avaliar a qualidade do resultado da execução do algoritmo de resolução de entidades, mesmo que só se saiba as respostas corretas para parte do conjunto de dados utilizado.

4.2 Algoritmo de resolução de entidades utilizado na abordagem ACERPI

A abordagem ACERPI (SCHMITZ et al., 2021) utiliza técnicas de *clustering* para performar a resolução de entidades. Essa abordagem gera um algoritmo bastante simples e intuitivo que compara cada referência do conjunto de dados com todas as entidades conhecidas, agrupando com uma das entidades conhecidas ou criando uma nova entidade caso nenhuma entidade conhecida seja apropriada. Este algoritmo é descrito no Algoritmo 1.

Este algoritmo é implementado em Python com as entidades e referências sendo mantidas em coleções do banco de dados MongoDB. Os conjuntos de referências e entidades são buscados nessas coleções através de consultas ao banco de dados. A consulta que busca o conjunto de referências é bastante simples e não possui nenhum parâmetro, porque é preciso buscar todas as referências no banco de dados para agrupá-las em alguma entidade. A consulta que busca as entidades com as quais são comparadas as referências é onde existe espaço para a aplicação de técnicas de blocagem.

Algoritmo 1: Algoritmo para a resolução de entidades.

Entrada: conjunto de referências
Saída : conjunto de entidades

```
1  referencias ← conjunto de referências;
2  entidades ← ∅;

3  para cada referencia r em referencias faça
4  |  agrupado ← falso;
5  |  para cada entidade e em entidades faça
6  |  |  se match(r, e) então
7  |  |  |  e ← entidades ∪ r;
8  |  |  |  agrupado ← verdadeiro;
9  |  |  |  break;
10 |  |  fim
11 |  fim
12 |  se não agrupado então
13 |  |  entidades ← entidades ∪ nova_entidade(r);
14 |  fim
15 fim
```

4.3 Algoritmo de comparação entre referências e entidades

Na Seção 4.2, foi mostrado o algoritmo para a resolução de entidades na abordagem ACERPI. A parte mais importante do Algoritmo 1 é o algoritmo que decide se uma referência deve ser agrupada com uma entidade. Esse algoritmo é implementado na função *match* no Algoritmo 1.

A eficácia e eficiência da função *match* impactam fortemente a eficácia e eficiência do algoritmo de resolução de entidades como um todo. Neste trabalho, é explorado o impacto da aplicação de técnicas de blocagem em duas versões da função *match*. Ambas versões são idênticas em tudo com a exceção do seu mecanismo de comparação das *strings* que representam os nomes das referências.

Para entender o algoritmo da função *match*, é necessário antes entender como as referências e as entidades são representadas no banco de dados. Ambas as estruturas das referências e entidades são bastante simples. As estruturas de referências e entidades seguem o modelo descrito nas Listagens 4.1 e 4.2.

Listagem 4.1 – Estrutura de dados das referências no banco de dados

```
{
    "id": Identificador unico atribuido manualmente (int),
    "name": Nome reconhecido para esta referencia (string),
    "siape": Lista das SIAPEs (lista de strings),
    "document": {
        "name": Codigo que identifica o documento que gerou esta
            referencia (string)
    }
}
```

Listagem 4.2 – Estrutura de dados das entidades no banco de dados

```
{
    "records": Lista dos identificadores das referencias agrupadas
        nessa entidade (lista de ints),
    "names": Lista dos nomes das referencias agrupadas nessa
        entidade (lista de strings),
    "siapes": Lista das SIAPEs das referencias agrupadas nessa
        entidade (lista de ints)
}
```

As Listagens 4.3 e 4.4 descrevem dois exemplos de referências ao professor Renato Perez Ribas do instituto de informática da UFRGS no formato usado no banco de dados.

Listagem 4.3 – Exemplo 1 de uma referência no banco de dados

```
{
    "id": 168816,
    "name": 'RENATO PEREZ RIBAS',
    "siape": [],
    "document": { name: '55572' }
}
```

Listagem 4.4 – Exemplo 2 de uma referência no banco de dados

```
{
    "id": 104511,
    "name": 'Renato Perez Ribas',
    "siape": [],
```

```
5       "document": { name: '36516' }
6     }
```

E a Listagem 4.5 mostra a entidade que representa o professor Renato Perez Ribas no formato usado no banco de dados e com os códigos das duas referências mostradas nas Listagens 4.3 e 4.4 na lista de códigos.

Listagem 4.5 – Exemplo 1 de uma entidade no banco de dados

```
1  {
2      "records": [
3          10317,  11353,  12606,  21941,
4          23127,  39679,  40966,  49122,
5          56142,  57230,  64111,  65361,
6          75580,  98051, 104511, 104654,
7         117747, 131362, 153038, 168816,
8          184549
9      ],
10     "names": [
11       'RENATO PEREZ RIBAS',
12       'Renato Perez Ribas',
13       'RENATO  PEREZ  RIBAS'
14     ],
15     "siapes": [ '1310880' ],
16 }
```

Como mostrado nas Figuras 4.3 e 4.4, existem apenas duas informações nas referências que podem ser usadas para identificar a entidade com a qual a referência deve ser agrupada: o nome e o SIAPE. O algoritmo da função *match* agrupa a referência e a entidade se ambos tem o mesmo SIAPE único ou se os nomes forem suficientemente semelhantes. O Algoritmo 2 descreve esse algoritmo.

A última parte do algoritmo que ainda não foi explicitamente descrita é o mecanismo de comparação dos nomes. É nesse mecanismo que as duas versões do algoritmo de resolução de entidades diferem.

A primeira versão, apresentada no trabalho de Schmitz et al. (2021), utiliza uma comparação exata de *strings* após normalizar os argumentos convertendo todas as letras para minúsculas e removendo espaços. Esta versão é chamada de **comparação exata** a

Algoritmo 2: Algoritmo para a função *match*

Entrada: referência, entidade
Saída : decisão

```
se tamanho(entidade["siapes"]) == 1 e tamanho(referência["siape"]) == 1 e
  entidade["siapes"][0] == referência["siape"][0] então
    retorna verdadeiro;
fim

para cada nome n em entidade["names"] faça
    se comparar(n, referencia["name"]) então
        retorna verdadeiro;
    fim
fim

retorna falso;
```

partir deste ponto.

A segunda versão, que é chamada de **Jaro-Winkler** a partir deste ponto, usa o algoritmo Jaro-Winkler para a comparação de *strings* após uma etapa de normalização que converte letras que não existem no conjunto de caracteres ASCII para caracteres equivalentes que existam neste conjunto. Por exemplo, "à"seria convertido para "a".

A versão que usa Jaro-Winkler foi inspirada pelo trabalho de Eich (2021) que estava sendo desenvolvido ao mesmo tempo que ACERPI-Block. A versão usada no ACERPI-Block foi escolhida de uma versão intermediária do ACERPI-Link (EICH, 2021), quando parecia ser a versão mais promissora, e não necessariamente será a versão considerada melhor quando o trabalho ACERPI-Link estiver finalizado. A versão usada no ACERPI-Block usa `0.9` como a similaridade mínima para agrupar uma referência a uma entidade. Se a similaridade, calculada pelo algoritmo Jaro-Winkler, entre o nome da referência e qualquer um dos nomes da entidade com a qual esta referência está sendo comparada for maior que `0.9`, essa referência é agrupada na entidade.

4.4 Aplicação de blocagem

O objetivo das técnicas de blocagem e filtragem é reduzir o número de comparações necessárias para a execução do algoritmo de resolução de entidades. Essas técnicas diferem na sua abordagem para a redução do número de comparações. Enquanto a filtragem usa regras que garantem que não haverá redução da eficácia do algoritmo de reso-

lução de entidades, a blocagem aceita que haja uma pequena queda na eficácia em troca de um aumento da eficiência (PAPADAKIS et al., 2019). Neste trabalho, são somente exploradas técnicas de blocagem.

Para o algoritmo de resolução de entidades da abordagem ACERPI, descrito na Seção 4.2, as técnicas de blocagem são melhor aplicadas na consulta ao banco de dados para buscar as entidades que devem ser comparadas com uma determinada referência. A versão original desta consulta não possui nenhum filtro e, simplesmente, retorna todas as entidades no banco de dados.

Neste trabalho são exploradas três maneiras diferentes em que essas técnicas podem ser aplicadas. São elas:

- Buscando apenas as entidades cujo primeiro nome da lista de nomes começa com a mesma letra, independentemente de ser maiúscula ou minúscula, que o nome da referência. Então, se a referência tivesse o nome "RENATO PEREZ RIBAS", ela seria comparada somente com entidades cujo primeiro nome da lista de nomes também começa com as letras "R"ou "r".
- Buscando apenas as entidades cujo primeiro nome da lista de nomes começa com uma das iniciais do nome da referência, independentemente de ser maiúscula ou minúscula. Logo, se a referência tivesse o nome "RENATO PEREZ RIBAS", ela seria comparada somente com entidades cujo primeiro nome começasse com "R", "r", "P"ou "p".
- Buscando apenas as entidades cujo primeiro nome da lista de nomes tem tamanho maior que um limite inferior, baseado no tamanho do nome da referência, e menor que um limite superior, também baseado no tamanho do nome da referência. Neste trabalho, são usados os valores 33%, 50% e 66% para o limite inferior e os valores 150%, 200% e 300% para o limite superior.

Essas técnicas são combinadas em 8 maneiras diferentes. São elas:

- Filtrando somente pela primeira letra.
- Filtrando pela primeira letra e pelo tamanho com limites de 33% e 300%.
- Filtrando pela primeira letra e pelo tamanho com limites de 50% e 200%.
- Filtrando pela primeira letra e pelo tamanho com limites de 66% e 150%.
- Filtrando somente pelas iniciais.
- Filtrando pelas iniciais e pelo tamanho com limites de 33% e 300%.
- Filtrando pelas iniciais e pelo tamanho com limites de 50% e 200%.

- Filtrando pelas iniciais e pelo tamanho com limites de 66% e 150%.

Cada uma dessas combinações é aplicada tanto à versão do algoritmo usando comparação exata, quanto à versão usando Jaro-Winkler, gerando um total de dezesseis diferentes versões do algoritmo.

Para filtrar os resultados das consultas à coleção de entidades, foi usado o operador de expressões regulares do banco de dados MongoDB para aplicar expressões regulares simples.

4.5 Algoritmo para avaliação de qualidade

A qualidade do resultado do algoritmo de resolução de entidades neste trabalho é medida usando a técnica de avaliação por pares (MENESTRINA; WHANG; GARCIA-MOLINA, 2010). Esta métrica foi escolhida para manter compatibilidade com o trabalho que originou a abordagem ACERPI (SCHMITZ et al., 2021).

Para avaliar os grupos gerados pelo algoritmo de resolução de entidade usando a abordagem de avaliação por pares, os grupos são interpretados como uma lista de pares com a combinação em pares de todos os elementos que pertencem a um mesmo grupo. Ou seja, para qualquer grupo G, são gerados todos os pares de elementos Ri e Rj tal que $Ri, Rj \in G$ e $Ri \neq Rj$. Um grupo $G = \{ref1, ref2, ref3\}$, por exemplo, seria interpretado como uma lista contendo os pares $(ref1, ref2), (ref1, ref3) e (ref2, ref3)$, pois todos estes pares contém elementos que são referências à mesma entidade do mundo real.

A avaliação baseada em pares gera os pares para todos os grupos resultantes da execução do algoritmo de resolução de entidades e os compara com um conjunto de pares, chamado a partir deste ponto de **conjunto de ouro**, que contém apenas pares que se sabe com certeza que fazem parte do mesmo grupo, ou seja, são referências da mesma entidade do mundo real. A avaliação gera as seguintes métricas:

- Verdadeiros positivos: número de pares que existem no resultado da execução do algoritmo de resolução de entidades e no conjunto de ouro. Isto significa que o par de referências agrupado pelo algoritmo realmente representa uma mesma entidade do mundo real.
- Falsos positivos: número de pares que existem no resultado da execução do algoritmo de resolução de entidades, mas não existe no conjunto de ouro. Isso significa que o par de referências agrupado pelo algoritmo não representa uma mesma enti-

dade do mundo real e foi erroneamente agrupado.

- Falsos negativos: número de pares que não existem no resultado da execução do algoritmo de resolução de entidades, mas existem no conjunto de ouro. Isto significa que o par realmente representa uma mesma entidade do mundo real, mas não foi agrupado pelo algoritmo, sugerindo que o algoritmo considera que os elementos do par são referências a entidades distintas do mundo real.

A partir desta métrica, é possível gerar métricas novas e mais fáceis de analisar. Estas métricas são:

- Revocação: intuitivamente, é o percentual de duplas corretamente agrupadas entre todas as duplas que de fato representam uma mesma entidade do mundo real. Matematicamente, é o resultado da divisão do número de verdadeiros positivos pela soma dos verdadeiros positivos e falso negativos: $\mathit{revocação} = \frac{vp}{vp+fn}$.
- Precisão: intuitivamente, é o percentual de duplas corretamente agrupadas entre todas as duplas agrupadas pelo algoritmo. Matematicamente, é o resultado da divisão do número de verdadeiros positivos pela soma dos verdadeiros positivos e falsos positivos: $\mathit{precisão} = \frac{vp}{vp+fp}$.
- Medida F1: intuitivamente, é a média harmônica entre a revocação e precisão. Matematicamente, é descrita como o dobro da divisão entre a multiplicação da revocação e precisão pela soma da revocação e precisão: $F1 = 2 \cdot \frac{\mathit{precisão} \cdot \mathit{revocação}}{\mathit{precisão} + \mathit{revocação}}$.

Para a realização deste trabalho, foi criado um novo algoritmo para calcular essas métricas, ainda usando a abordagem por pares. O algoritmo antigo, usado no trabalho de Schmitz et al. (2021), tinha duas limitações quando aplicado no contexto de blocagem que implicaram na criação de um novo algoritmo:

- Ele não foi feito para ser usado em um cenário no qual o conjunto de ouro não possui a resposta para todos as referências analisadas pelo algoritmo de resolução de entidades, apenas uma parte delas. Isso criava uma situação na qual não era possível analisar a qualidade do resultado da aplicação do algoritmo de resolução de entidades ao conjunto de dados completo, pois só são conhecida as respostas corretas para uma pequena parte deste conjunto.
- Como ele não foi criado para ser usado em um conjunto de dados muito grande, a eficiência do algoritmo não era uma prioridade. Este fato tornou a execução do algoritmo para o conjunto de dados completo impraticável, pois levaria muito tempo e utilizaria muita memória.

Para que o algoritmo gerasse métricas corretas quando o conjunto de ouro contém somente um subconjunto das referências utilizadas na execução do algoritmo de resolução de entidades, foi adicionada uma nova regra para gerar os pares a partir dos grupos gerados pelo algoritmo. Antes de começar o processo de geração de pares, retira-se do grupo G todas as referências Ri tal que $Ri \notin O$, onde O é o conjunto de todas as referências que existem em algum par do conjunto de ouro. Desta maneira, evita-se adicionar métricas para referências cuja entidade do mundo real é desconhecida.

O Algoritmo 3 descreve o algoritmo usado para a avaliação dos resultados da execução do algoritmo de resolução de entidades. Assim como o algoritmo criado originalmente no trabalho de Schmitz et al. (2021), o Algoritmo 3 é baseado na abordagem por pares (MENESTRINA; WHANG; GARCIA-MOLINA, 2010). Diferentemente do algoritmo usado originalmente no trabalho de Schmitz et al. (2021), o Algoritmo 3, que é usado neste trabalho, usa conjuntos, na forma da estrutura de dados *set* da linguagem Python, e a fórmula do coeficiente binomial para calcular os pares corretos e incorretos de maneira eficiente e sem precisar gerar todos os pares para o conjunto de ouro e a entidade sendo avaliada.

4.6 Considerações Finais

Neste capítulo, foram apresentados os algoritmos de resolução de entidades e de avaliação do resultado do algoritmo de resolução de entidades que são usados e também como são aplicadas as técnicas de blocagem. Também foram apresentados alguns dos conceitos básicos por trás das técnicas que são aplicadas e exemplos do conjunto de dados que é utilizado. Cabe ressaltar que constitui contribuição deste trabalho: a implementação de versões otimizadas do algoritmo de resolução de entidades da abordagem ACERPI, a execução de experimentos e análise dos resultados destes experimentos para demonstrar o efeito da aplicação destas técnicas de blocagem na eficiência e eficácia do algoritmo de resolução de entidade, e a criação de um algoritmo mais eficiente para a avaliação da qualidade do resultado do programa de resolução de entidades, que também seja capaz de analisar a qualidade do resultado quando sabemos os agrupamentos corretos para apenas um subconjunto do conjunto de dados completo.

Algoritmo 3: Algoritmo para avaliação de qualidade

Entrada: conjunto de ouro, referências existentes no conjunto de ouro, agrupamentos
Saída : verdadeiros positivos, falsos positivos, falsos negativos

```
1  conj_ouro ← conjunto de ouro;
2  ref_ouro ← referências existentes no conjunto de ouro;
3  entidades ← agrupamentos;

4  verdadeiros_positivos ← 0;
5  falsos_positivos ← 0;
6  falsos_negativos ← 0;

7  para cada entidade e em entidades faça
8      entidade_avaliavel ← e ∩ conj_ouro;

9      para cada entidade entidade_ouro em conj_ouro faça
10         membros_verdadeiros ← ref_ouro ∩ entidade_avaliavel;
11         entidade_avaliavel ← entidade_avaliavel - membros_verdadeiros;

12         verdadeiros_positivos ← verdadeiros_positivos +
             coeficiente_binomial(tamanho(membros_verdadeiros), 2);

13         falsos_positivos ← falsos_positivos +
             tamanho(membros_verdadeiros) · tamanho(entidade_avaliavel);
14     fim
15 fim

16 verdadeiros_positivos_possiveis ← 0;
17 para cada entidade entidade_ouro em conj_ouro faça
18     verdadeiros_positivos_possiveis ← verdadeiros_positivos_possiveis +
         coeficiente_binomial(tamanho(entidade_ouro), 2);
19 fim
20 falsos_negativos ← verdadeiros_positivos_possiveis - verdadeiros_positivos;

21 retorna verdadeiros_positivos, falsos_positivos, falsos_negativos ;
```

5 AVALIAÇÃO EXPERIMENTAL

Este capítulo descreve os experimentos desenvolvidos para avaliar a eficácia das técnicas de blocagem aplicadas ao algoritmo de resolução de entidades da abordagem ACERPI. Inicialmente, é apresentada a maneira em que as técnicas são aplicadas ao algoritmo de resolução de entidades da abordagem ACERPI. Em seguida, é apresentada a metodologia usada para realizar os experimentos. Depois, são apresentadas as métricas coletadas durante os experimentos e a relevância das mesmas para a análise da eficácia da aplicação das técnicas de blocagem. Por fim, os resultados dos experimentos são exaustivamente analisados e descritos.

5.1 Visão Geral

Nesta seção, são detalhados os experimentos realizados no trabalho. São eles a execução e avaliação do algoritmo:

1. Sem a aplicação de blocagem para gerar o nosso caso base.
2. Aplicando técnicas de blocagem pela primeira letra do nome.
3. Aplicando técnicas de blocagem pela primeira letra e tamanho do nome.

5.2 Configurações Gerais dos Experimentos

As partes em comum a todos os experimentos são explicadas nesta seção, enquanto as particularidades são abordadas individualmente no detalhamento de cada experimento.

5.2.1 Fontes de Dados

Para todos os experimentos, foi utilizada a mesma fonte de dados:

- Banco de dados com 194,000 referências obtidas através da aplicação das técnicas de coleta de documentos e reconhecimento de Entidades Nomeadas da abordagem ACERPI nas fontes de dados que foram descritas da seguinte maneira no trabalho Schmitz et al. (2021):

- DOCS-UFRGS - Documentos públicos da Universidade Federal do Rio Grande do Sul. Desta fonte, foram extraídos documentos, em sua maioria Portarias, do repositório da Universidade. Os documentos podem ser acessados através de endereços generalizados a partir do padrão de navegação (Palmieri Lage et al., 2004) `https://www1.ufrgs.br/sistemas/sde/gerencia-documentos/index.php/publico/ExibirPDF?documento=[0-9]{1,6}`, no qual, para este trabalho, a expressão regular foi substituída pelos valores 18000 a 105995. Para esta fonte, cada arquivo contém no máximo uma portaria emitida pela Universidade.
- DOCS-IFRS - Documentos públicos do Instituto Federal do Rio Grande do Sul. Para esta fonte, foram extraídos documentos de três repositórios. São eles:

  1. Repositório antigo IFRS, *Campus* Ibirubá (IFRS, Campus Ibirubá, 2011). Este repositório é composto por arquivos apenas do *Campus* Ibirubá de 2013 a 2017 (inclusive).
  2. Repositório atual IFRS, *Campus* Ibirubá (IFRS, Campus Ibirubá, 2018). Este repositório complementa o anterior com os documentos a partir de 2017.
  3. Repositório geral do IFRS (IFRS, 2020). Este repositório é composto por documentos do IFRS, contemplando portarias relacionadas aos servidores dos 17 campi deste instituto desde 2017.

  Os dois últimos seguem sendo atualizados conforme novas publicações acontecem. Para esta fonte, cada arquivo pode conter uma quantidade arbitrária de portarias emitidas pela instituição. O banco de dados usado foi obtido através da aplicação em dados coletados até 3 de Março de 2020 para a fonte DOCS-UFRGS e 17 de Fevereiro de 2020 para a fonte DOCS-IFRS.

Esses dados foram coletados durante o desenvolvimento do trabalho Schmitz et al. (2021) e cedidos pelo autor, Christian Schmitz, para o desenvolvimento deste trabalho.

5.2.2 Ambiente de configuração

Todos os experimentos foram realizados em máquinas da Google Cloud Platform modelo n2-standard-4 com 4 CPUs nas arquiteturas Intel Cascade Lake, 16 GB de memória RAM, SSD local e internet com banda de 10 Gbps.

Estas máquinas foram configuradas com Linux Debian 10, e rodavam o banco de dados *MongoDB Community Server* (Seção 2.2.1) para o armazenamento dos dados, a extensão para Python *cProfile* (Seção 2.2.2) para a coleta de dados sobre chamadas de função e o pacote *time* (Seção 2.2.3) do Linux para a coleta de dados sobre tempo de execução e utilização de CPU.

5.2.3 Ferramentas Utilizadas

Para a execução e coleta de métricas dos experimentos, foram usadas as seguintes ferramentas:

- *MongoDB Community Server*: programa de banco de dados NoSQL que usa documentos baseados em JSON para armazenamento de dados. Esta ferramenta foi usada para armazenar os dados sobre os quais o algoritmo de resolução de dados opera.
- *cProfile*: uma extensão em C para Python que serve para coletar métricas como chamadas de funções e tempo de execução de pedaços de código. Essa extensão foi usada para coletar dados sobre o número de vezes que cada função foi chamada e quanto tempo o programa gastou executando o código dentro dessas funções.
- Pacote *time* do Linux: um utilitário de Linux para coletar dados sobre a execução de um processo, como tempo de execução, uso de CPU e uso de memória. Esse pacote foi usado para coletar métricas sobre o tempo de execução e utilização da CPU durante a execução do programa de resolução de entidades.

5.2.4 Algoritmo de resolução de entidades

Em todos os experimentos realizados neste trabalho, foi usado o algoritmo de resolução de entidades descrito na Seção 4.2, assim como no trabalho de Schmitz et al. (2021). Porém, para este trabalho, foram feitas otimizações no código que implementa

esse algoritmo, que resultaram em uma diminuição do tempo de execução que pode chegar a **99.74%**, baseado em execuções com o conjunto de dados de teste.

Esta otimização consiste em remover uma consulta custosa ao banco de dados do código. Essa consulta é descrita na Listagem 5.1. Essa consulta é usada na versão original do código para buscar no banco de dados todas as referências que fazem parte da entidade que está sendo comparada com a referência atual. Esta consulta é desnecessária, pois as únicas informações que temos no banco de dados para cada referência são seu nome e SIAPE, e essa informação já está agregada na própria entidade. Isso significa que todas as informações que esta consulta busca já estão na própria entidade e podem ser usadas sem uma consulta ao banco de dados.

Listagem 5.1 – Consulta usada no programa original

```
1 referencias = colecao_de_referencias.find({
2     'id': {
3         '$in': entidade['records']
4     }
5 })
```

A consulta descrita na Listagem 5.1 é custosa porque usa o operador `"$in"`, que compara o atributo especificado, neste caso `"id"`, com todos os elementos da lista passada como argumento, neste caso `"entidade['records']"`, que representa a lista da `ids` de todas as referências que já foram agrupadas na entidade.

Todos os experimentos executados neste trabalho usam versões do algoritmo de resolução de entidade que têm a otimização descrita nesta seção. É importante frisar que esta otimização não altera o algoritmo, e, por consequência, não altera os resultados gerados pelo programa, apenas a eficiência do programa.

5.2.5 Metodologia

Em todos os experimentos, o utilitário *time* do Linux foi utilizado para coletar métricas sobre o uso de recursos do processo que executa o programa. E, em todos os experimentos, o algoritmo de avaliação do resultado do algoritmo de resolução de entidades foi executado após a execução do programa de resolução de entidades para coletar métricas sobre a qualidade do resultado.

A metodologia aplicada aos experimentos seguem o seguinte roteiro:

1. Rodar o algoritmo de resolução de entidades uma vez com o conjunto de dados de teste com a extensão *cProfile* desabilitada. O objetivo desta execução é somente coletar métricas de qualidade sobre a aplicação do algoritmo a um conjunto de dados pequeno.
2. Rodar o algoritmo de resolução de entidades uma vez no conjunto de dados completo com a extensão *cProfile* habilitada para coletar dados de tempo de execução ao nível de funções e número de chamadas para cada função executada.
3. Rodar o algoritmo de resolução de entidades dez vezes no conjunto de dados completo, desta vez com a extensão *cProfile* desabilitada para evitar que o *overhead* dessa extensão influencie o tempo de execução do algoritmo.

A execução com a extensão *cProfile* habilitada acontece apenas uma vez porque o objetivo do uso desta extensão é coletar o número de comparações que são realizadas em cada experimento e analisar os pontos do código que utilizam a maior parte do tempo da execução do algoritmo. Como o conjunto de dados utilizado é exatamente o mesmo em todas as execuções e o algoritmo é determinístico, o programa sempre executa o mesmo número de operações de comparação e, por isso, não precisa ser executado múltiplas vezes com a extensão habilitada.

As métricas de tempo providas pela extensão não são analisadas como métricas absolutas, pois a sobrecarga de executar as operações da extensão para coletar métricas do algoritmo pode distorcer as métricas de tempo. Por esta razão, essas métricas de tempo são analisadas apenas como parte do tempo total da execução do próprio experimento que as geraram. Essas métricas são usadas apenas para ilustrar quanto do tempo de execução é gasto fazendo comparações entre referências.

A análise do tempo de execução e utilização de CPU de cada versão do algoritmo são feita a partir das métricas coletadas pelo utilitário *time* do Linux nas dez execuções com o conjunto de dados completo e com a extensão *cProfile* desabilitada.

As métricas de qualidade geradas na execução do algoritmo com o conjunto de dados de teste são usadas para analisar o quanto a adição de mais dados causa confusão no algoritmo e, também, para manter um *link* com as métricas de qualidade utilizadas no trabalho que originalmente desenvolveu o ACERPI.

5.2.6 Métricas de qualidade

A qualidade do resultado do algoritmo de resolução de entidades neste trabalho foi medida usando a técnica de avaliação por pares descrita na Seção 4.5. A avaliação gera as seguintes métricas:

- Verdadeiros positivos: número de pares que foram corretamente agrupados.
- Falsos positivos: número de pares que incorretamente agrupados.
- Falsos negativos: número de pares que deveriam ter sido agrupados pelo algoritmo mas não foram.
- Revocação: intuitivamente, é o percentual de duplas corretamente agrupadas entre todas as duplas que de fato representam uma mesma entidade do mundo real. Matematicamente, é o resultado da divisão do número de verdadeiros positivos pela soma dos verdadeiros positivos e falso negativos: $\mathit{revocação} = \frac{vp}{vp+fn}$.
- Precisão: intuitivamente, é o percentual de duplas corretamente agrupadas entre todas as duplas agrupadas pelo algoritmo. Matematicamente, é o resultado da divisão do número de verdadeiros positivos pela soma dos verdadeiros positivos e falsos positivos: $\mathit{precisão} = \frac{vp}{vp+fp}$.
- Medida F1: intuitivamente, é a média harmônica entre a revocação e precisão. Matemáticamente, é descrita como o dobro da divisão entre a multiplicação da revocação e precisão pela soma da revocação e precisão: $F1 = 2 \cdot \frac{\mathit{precisão} \cdot \mathit{revocação}}{\mathit{precisão} + \mathit{revocação}}$.

5.2.7 Conjuntos de dados

Durante os experimentos, são mencionados dois conjuntos de dados. Um deles chamado de **conjunto de dados completo** e o outro de **conjunto de dados de teste**.

O conjunto de dados de teste é composto por 613 referências para as quais são conhecidas as entidades do mundo real às quais elas se referem, ou seja, são conhecidos os grupos que devem ser gerados. Essas são referências aos servidores públicos da UFRGS.

O conjunto de dados completo é composto por 194,000 referências. O conjunto de dados de teste é um subconjunto do conjunto de dados completo. É importante notar que não são conhecidas as entidades do mundo real para todas essas referências, apenas para as referências que compõe o subconjunto que compõe o conjunto de teste. Todas as métricas de qualidade para as execuções com o conjunto de dados completo são baseadas

nos acertos e erros dos agrupamentos dos elementos do subconjunto de teste.

5.2.8 Convenções para nomenclatura de tabelas e figuras

Durante este capítulo, são apresentadas múltiplas tabelas e figuras. Para facilitar a legibilidade dos nomes destes recursos sem precisar sacrificar a quantidade de informação sobre o que estes recursos representam, são usadas convenções para o nome. Se o nome contém:

- **Comparação exata**, significa que os resultados mostrados no recurso foram coletados a partir da execução de versões do algoritmo que utilizam comparação exata de *strings*.
- **Jaro-Winkler**, significa que os resultados mostrados no recurso foram coletados a partir da execução de versões do algoritmo que utilizam o algoritmo Jaro-Winkler para comparação de *strings*.
- **Sem blocagem**, significa que os resultados mostrados no recurso foram coletados a partir da execução de versões do algoritmo que não tem nenhum tipo de blocagem aplicada.
- **Conjunto de dados de teste**, significa que os algoritmos foram aplicados ao conjunto de dados de teste, que já foi discutido na Seção 5.2.7. Caso não seja especificado no título da figura, é usado o conjunto de dados completo.
- **Conjunto de dados completo**, significa que os algoritmos foram aplicados ao conjunto de dados completo, que já foi discutido na Seção 5.2.7. Caso não seja especificado no título da figura, é usado o conjunto de dados completo.
- **Blocagem por letra**, significa que os resultados mostrados no recurso vieram da execução de versões do algoritmo que possuem técnicas de blocagem baseadas na primeira letra do primeiro nome das entidade. Essas técnicas podem ser tanto blocagem pela primeira letra do nome da referência como blocagem pelas iniciais do nome da referência.
- **Blocagem por tamanho**, significa que os resultados mostrados no recurso vieram da execução de versões do algoritmo que possuem técnicas de blocagem baseadas no tamanho do nome da referência. Blocagem por tamanho inclui aplicação da técnica de blocagem por todos os tamanhos explorados neste trabalho.

5.3 Experimento 1 - Sem aplicação de nenhuma técnica de blocagem

Para avaliar a eficácia das técnicas de blocagem, é necessário primeiro avaliar a performance do algoritmo de resolução de entidades sem a aplicação de nenhuma técnica de blocagem.

Neste experimento, é analisada a execução do algoritmo de resolução de entidades usando duas técnicas diferentes para comparação de *strings*: comparação exata e Jaro-Winkler

Este experimento foi executado utilizando a metodologia descrita na Seção 5.2.5.

5.3.1 Resultados

Nas subseções seguintes, são mostrados os resultados deste experimento através de diversas métricas. Essas métricas são apresentadas em tabelas e figuras. Para facilitar a leitura destas tabelas e figuras, são apresentadas algumas convenções usadas.

Nas subseções a seguir, é usada a seguinte convenção para o nome das colunas nas tabelas:

- A coluna **Comparação exata** representa a versão do algoritmo de resolução de entidades da abordagem ACERPI que usa comparação exata de *strings*.
- A coluna **Jaro-Winkler** representa a versão do algoritmo de resolução de entidades da abordagem ACERPI que usa o algoritmo Jaro-Winkler para comparar *strings*.

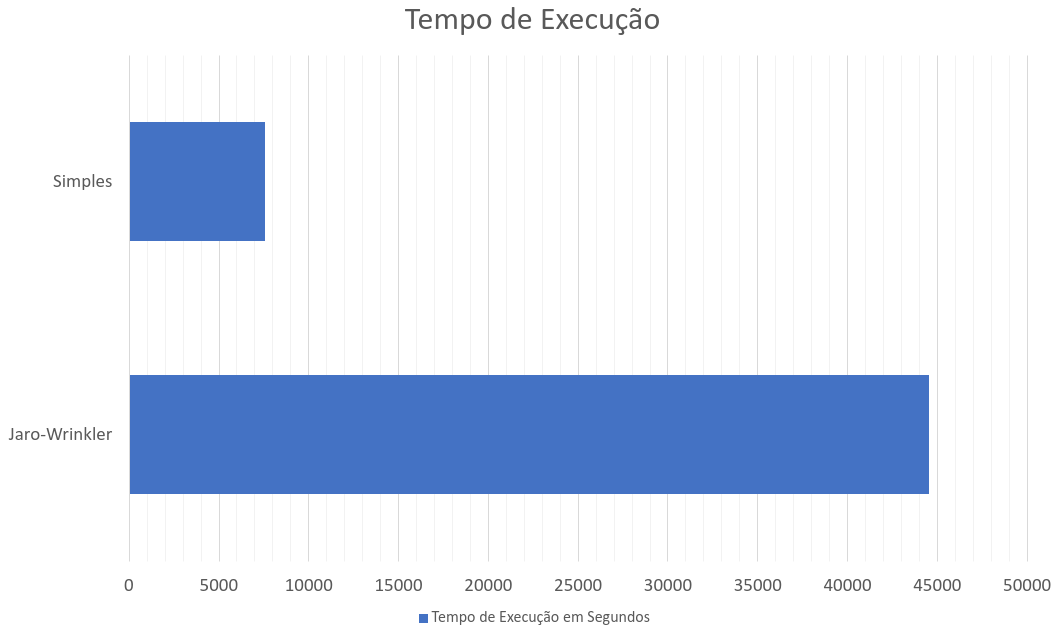
5.3.2 Métricas de tempo total de execução

A Tabela 5.1 mostra as métricas de tempo de execução, em segundos, das duas versões do algoritmo. Também mostra que usar o algoritmo de comparação de *strings* Jaro-Winkler tem um impacto negativo no tempo de execução. Apesar de possuir todos os outros aspectos do algoritmo em comum, usar Jaro-Winkler faz com que o programa leve, em média, 5.88 vezes mais tempo para executar.

Tabela 5.1 – Tempo de execução - Sem blocagem

Métrica de tempo	Comparação exata	Jaro-Winkler
Média	7585.8s	44571.1s
Máximo	7838.0s	46720.0s
Mínimo	7386.0s	43402.0s
Desvio padrão	164.5s	1154.1s

Figura 5.1 – Tempo médio de execução - Sem blocagem



5.3.3 Métricas sobre as operações de comparação

A Tabela 5.2 mostra as métricas mais relevantes coletadas pela extensão *cProfile*, número de comparações entre nomes e o tempo total que cada versão gastou realizando essas comparações. Lembrando sempre que essas métricas foram coletadas em um experimento separado, que não faz parte do conjunto de experimentos que geraram as outras métricas de tempo com o conjunto de dados completo.

Tabela 5.2 – Número de comparações - Sem blocagem

Métrica	**Comparação exata**	**Jaro-Winkler**
Comparações	1,259,040,135	1,439,499,577
Tempo das comparações	3553s	60720s
Tempo total	12866s	69348s

5.3.4 Métricas de utilização de CPU

O programa, escrito em Python, utiliza apenas uma *thread* e não implementa nenhum tipo de paralelismo em seu algoritmo. Isso significa que cada vez que faz uma consulta ao banco de dados, ele espera a resposta antes de retomar a execução. Isso causa vários momentos de inatividade durante a sua execução.

A Tabela 5.3 mostra as métricas relacionadas ao uso de CPU do processo rodando o algoritmo de resolução de entidades usando ambos algoritmos de comparação de *strings*. Um programa que tem uso da CPU em 90% passou 10% do seu tempo de execução esperando pela resposta de outros processos.

Tabela 5.3 – Utilização de CPU - Sem Blocagem

Métrica de uso de cpu	**Comparação exata**	**Jaro-Winkler**
Média	79.5%	96.5%
Máximo	80.0%	97.0%
Mínimo	79.0%	95.0%
Desvio padrão	0.5%	0.7%

5.3.5 Métricas de qualidade

As Tabelas 5.4 e 5.5 descrevem as métricas de qualidade do resultado de cada versão do algoritmo de acordo com as métricas de qualidade descritas na Seção 5.2.6.

A Tabela 5.4 apresenta as métricas de qualidade do resultado da execução do algoritmo com o conjunto de dados de teste. Este conjunto tem 613 elementos e é composto apenas de elementos que referenciam professores da UFRGS e para os quais o resultado correto é conhecido.

Tabela 5.4 – Métricas de qualidade - Sem blocagem - Conjunto de dados de teste

Métrica	**Comparação exata**	**Jaro-Winkler**
Verdadeiros Positivos	11597	13280
Falsos Positivos	0	0
Falsos Negativos	2481	798
Precisão	100.00%	100.00%
Revocação	82.38%	94.33%
Medida F1	90.34%	97.08%

A Tabela 5.4 mostra os resultados ao usar o algoritmo Jaro-Winkler, cuja suposição é : uma troca entre tempo e qualidade. A qualidade do resultado gerado pelo algoritmo que usa Jaro-Winkler para comparação dos nomes é melhor do que a qualidade do resultado do algoritmo que usa comparação exata. O motivo para isso é que o algoritmo Jaro-Winkler consegue agrupar elementos mesmo quando eles têm pequenas diferenças nos seus nomes, o que a comparação exata não consegue. Para o conjunto de dados de teste, isso aumenta o número de Verdadeiros Positivos e diminui o número de Falsos Negativos, sem aumentar o número de Falsos Positivos.

Os resultados da Tabela 5.4 mostram os pontos positivos do uso do algoritmo Jaro-Winkler para resolução de entidades. Mas esses resultados são gerados a partir de um conjunto de dados que não contém muitas referências, dessa maneira, tendo menos oportunidades para agrupamentos incorretos. Para uma análise melhor e um teste que se aproxime do uso desse algoritmo no mundo real, também foi executado o algoritmo para o conjunto de dados completo e teve sua qualidade final analisada. A Tabela 5.5 descreve a qualidade dos resultados de cada algoritmo para esse experimento usando o conjunto de dados completo.

Tabela 5.5 – Métricas de qualidade - Sem blocagem - Conjunto de dados completo

Métrica	**Comparação exata**	**Jaro-Winkler**
Verdadeiros Positivos	11597	9085
Falsos Positivos	0	16
Falsos Negativos	2481	4993
Precisão	100.00%	99.82%
Revocação	82.38%	64.53%
Medida F1	90.34%	78.39%

A Tabela 5.5 mostra um resultado que pode ser considerado inesperado. A quali-

dade da comparação exata é superior à qualidade da comparação usando Jaro-Winkler em todas as métricas. O que acontece aqui é que a mesma qualidade que fez Jaro-Winkler ter excelentes resultados quando é usado um conjunto pequeno, causa queda de qualidade quando se tem um conjunto de dados maior. Analisando os resultados do algoritmo que usa comparação exata de *strings* em ambos os casos, pode-se ver que a comparação exata se manteve imune ao aumento do número de referências devido à rigidez de sua comparação.

5.4 Experimento 2 - Aplicação de técnicas de blocagem pela primeira letra do nome

Este experimento envolve aplicar técnicas de blocagem aos algoritmos que foram utilizados na Seção 5.3 e analisar o efeito que essas técnicas tem em tempo de execução, número de comparações, utilização da CPU e qualidade do resultado.

São aplicadas duas versões diferentes da técnica de blocagem:

- **Blocagem pela primeira letra do nome:** quando são feitas consultas ao banco de dados por entidades para comparar com um determinado elemento. São consideradas apenas as entidades cujo primeiro nome da lista de nomes começa com a mesma letra, independentemente de ser maiúscula ou minúscula, que o nome do elemento.
- **Blocagem pelas iniciais do nome:** quando são feitas consultas ao banco de dados por entidades para comparar com um determinado elemento. São consideradas apenas as entidades cujo primeiro nome da lista de nomes começa com qualquer uma das iniciais do nome do elemento, independentemente de ser maiúscula ou minúscula.

5.4.1 Resultados

Em toda esta subseção, é usada a seguinte convenção para os nomes das colunas das tabelas:

- A coluna **Sem blocagem** representa os resultados quando não é aplicado nenhum tipo de blocagem.
- A coluna **Iniciais** representa os resultados da aplicação de blocagem na qual apenas as entidades cujo primeiro nome da lista de nomes começa com uma das iniciais do

nome da referência.

- A coluna **Primeira letra** representa os resultados da aplicação de blocagem na qual apenas as entidades cujo primeiro nome da lista de nomes começa com a mesma letra que o nome da referência.

5.4.2 Métricas de tempo total de execução

A primeira métrica a ser mostrada é a métrica que mede o tempo de execução do algoritmo de resolução de entidades. Essa métrica é coletada pelo programa *time* e é baseada no tempo de dez execuções do programa, com a extensão *cProfile* desabilitada e com o conjunto de dados completo. A Tabela 5.6 apresenta os resultados para o algoritmo de resolução de entidades utilizando comparação exata de *strings*.

Tabela 5.6 – Métricas de tempo - Blocagem por letra - Comparação exata

Métrica de tempo	Sem blocagem	Iniciais	Primeira letra
Média	7585.8s	2520.0s	1882.0s
Máximo	7838.0s	2563.0s	1976.0s
Mínimo	7386.0s	2458.0s	1742.0s
Desvio padrão	164.5s	32.0s	86.0s

A Tabela 5.7 mostra a mesma métrica, desta vez para o algoritmo de resolução de entidades que utiliza o algoritmo Jaro-Winkler para resolução de entidade.

Tabela 5.7 – Métricas de tempo - Blocagem por letra - Jaro-Winkler

Métrica de tempo	Sem blocagem	Iniciais	Primeira letra
Média	44571.1s	9265.0s	4330.8s
Máximo	46720.0s	10190.0s	4353.0s
Mínimo	43402.0s	8935.0s	4300.0s
Desvio padrão	1154.1s	430.8s	20.8s

5.4.3 Métricas sobre as operações de comparação

As métricas mostradas nas Tabelas 5.8 e 5.9 são referentes à função *match* descrita na Seção 4.3. A linha **Comparações** descreve o número de vezes que a função *match* foi chamada durante a execução do algoritmo. A linha **Tempo das comparações** descreve o tempo total que foi gasto durante a execução do programa executando a função *match*. Por último, a linha **Tempo total** representa o tempo de execução do programa de resolução

de entidades como um todo. Lembrando que as métricas de tempo que são descritas aqui não fazem parte do conjunto de métricas de tempo que geraram as Tabelas 5.6 e 5.7.

A Tabela 5.8 mostra as métricas das operações de comparação para o algoritmo de resolução de entidades que usa comparação exata de *strings*. Estas métricas foram coletadas pela extensão *cProfile* a partir de uma única execução com a extensão habilitada e o conjunto de dados completo.

Tabela 5.8 – Número de comparações - Blocagem por letra - Comparação exata

Métrica	**Sem blocagem**	**Iniciais**	**Primeira letra**
Comparações	1,259,040,135	226,749,153	84,487,924
Tempo das comparações	3553s	665s	248s
Tempo total	12866s	3450s	2425s

A Tabela 5.9 mostra as mesmas métricas que a Tabela 5.8, mas desta vez para o algoritmo de resolução de entidades que usa o algoritmo Jaro-Winkler para comparação de *strings*.

Tabela 5.9 – Número de comparações - Blocagem por letra - Jaro-Winkler

Métrica	**Sem blocagem**	**Iniciais**	**Primeira letra**
Comparações	1,439,499,577	261,655,437	96,432,691
Tempo das comparações	60720s	11553s	4077s
Tempo total	69348s	14474s	6113s

5.4.4 Métricas de utilização de CPU

As Tabelas 5.10 e 5.11 apresentam métricas referentes à utilização da CPU durante a execução do programa de resolução de entidades. A utilização da CPU pelo programa é uma representação do percentual do seu tempo de execução que foi gasto executando o programa e quanto foi gasto esperando por respostas de outros processos. Quanto mais alta a utilização CPU, maior o percentual do tempo que foi gasto executando o algoritmo. Estas métricas foram coletadas usando o programa *time* a partir de dez execuções do programa de resolução de entidades com a extensão *cProfile* desabilitada e o conjunto completo de dados.

A Tabela 5.10 apresenta as métricas referentes à utilização da CPU para o algoritmo de resolução de entidades que usa comparação exata de *strings*.

A Tabela 5.11 apresenta as métricas referentes à utilização da CPU para o algoritmo de resolução de entidades que usa o algoritmo de comparação de *strings* Jaro-

Tabela 5.10 – Utilização de CPU - Blocagem por letra - Comparação exata

Métrica de tempo	Sem blocagem	Iniciais	Primeira letra
Média	79.5%	54.2%	37.9%
Máximo	80.0%	55.0%	39.0%
Mínimo	79.0%	54.0%	37.0%
Desvio padrão	0.5%	0.4%	0.9%

Winkler.

Tabela 5.11 – Utilização de CPU - Blocagem por letra - Jaro-Winkler

Métrica de tempo	Sem blocagem	Iniciais	Primeira letra
Média	96.5%	90.1%	75.0%
Máximo	97.0%	87.0%	75.0%
Mínimo	95.0%	54.0%	75.0%
Desvio padrão	0.7%	1.4%	0.0%

5.4.5 Métricas de qualidade usando o conjunto de dados de teste

As Tabelas 5.12 e 5.13 apresentam as métricas de qualidade descritas na Seção 5.2.6. Estas métricas foram calculadas a partir do programa de avaliação, também descrito na Seção 5.2.6, após a execução do programa de resolução de entidades com o conjunto de dados de teste.

A Tabela 5.12 apresenta as métricas de qualidade para o algoritmo de resolução de entidades usando comparação exata de *strings* aplicado ao conjunto de dados de teste.

Tabela 5.12 – Métricas de qualidade - Blocagem por letra - Comparação exata - Conjunto de dados de teste

Métrica	Sem blocagem	Iniciais	Primeira letra
Verdadeiros Positivos	11597	11597	11597
Falsos Positivos	0	0	0
Falsos Negativos	2481	2481	2481
Precisão	100.00%	100.00%	100.00%
Revocação	82.38%	82.38%	82.38%
Medida F1	90.34%	90.34%	90.34%

A Tabela 5.12 apresenta as métricas de qualidade para o algoritmo de resolução de entidades usando comparação exata de *strings* aplicado ao conjunto de dados de teste.

Tabela 5.13 – Métricas de qualidade - Blocagem por letra - Jaro-Winkler - Conjunto de dados de teste

Métrica	Sem blocagem	Iniciais	Primeira letra
Verdadeiros Positivos	13280	13280	13280
Falsos Positivos	0	0	0
Falsos Negativos	798	798	798
Precisão	100.00%	100.00%	100.00%
Revocação	94.33%	94.33%	94.33%
Medida F1	97.08%	97.08%	97.08%

5.4.6 Métricas de qualidade usando o conjunto de dados completo

As Tabelas 5.14 e 5.15 apresentam as métricas de qualidade descritas na Seção 5.2.6. Essas métricas foram calculadas a partir do programa de avaliação, também descrito na Seção 5.2.6, após a execução do programa de resolução de entidades com o conjunto de dados completo.

A Tabela 5.14 apresenta as métricas de qualidade para o algoritmo de resolução de entidades usando comparação exata de *strings* aplicado ao conjunto de dados completo.

Tabela 5.14 – Métricas de qualidade - Blocagem por letra - Comparação exata - Conjunto de dados completo

Métrica	Sem blocagem	Iniciais	Primeira letra
Verdadeiros Positivos	11597	11597	11597
Falsos Positivos	0	0	0
Falsos Negativos	2481	2481	2481
Precisão	100.00%	100.00%	100.00%
Revocação	82.38%	82.38%	82.38%
Medida F1	90.34%	90.34%	90.34%

A Tabela 5.15 apresenta as métricas de qualidade para o algoritmo de resolução de entidades usando o algoritmo Jaro-Winkler para comparação de *strings* aplicado ao conjunto de dados completo.

Tabela 5.15 – Métricas de qualidade - Blocagem por letra - Jaro-Winkler - Conjunto de dados completo

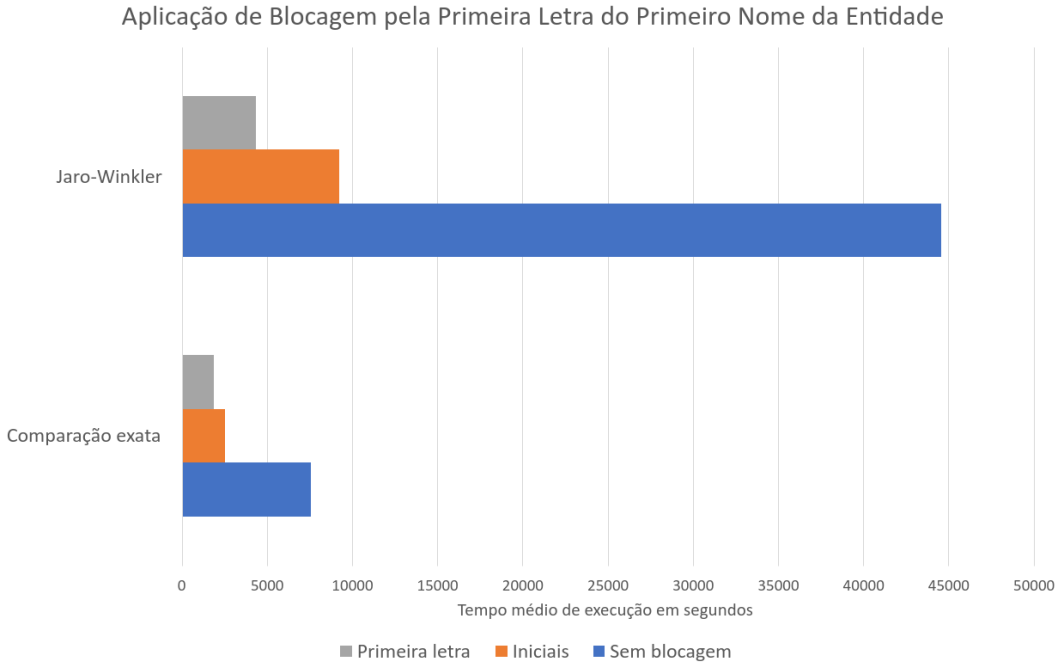
Métrica	Sem blocagem	Iniciais	Primeira letra
Verdadeiros Positivos	9085	9410	9406
Falsos Positivos	16	0	12
Falsos Negativos	4993	4668	4672
Precisão	99.82%	100.00%	99.87%
Revocação	64.53%	66.84%	66.81%
Medida F1	78.39%	80.13%	80.06%

5.4.7 Análise

Os resultados mostrados na Seção 5.4.1 levam à algumas constatações interessantes. Começando pelas métricas de tempo, mostradas nas Tabelas 5.6 e 5.7, é possível ver uma redução do tempo de execução usando blocagem para filtrar as entidades de comparação. A Figura 5.2 ilustra a diferença entre os tempos de execução médio para os algoritmos com a aplicação de cada uma das técnicas de blocagem apresentadas na Seção 5.4.

É possível ver na Figura 5.2 que, apesar de ambas as versões do algoritmo terem tido uma redução no seu tempo médio de execução, a versão que usa o algoritmo Jaro-Winkler para comparação de *strings* teve uma redução maior no seu tempo de execução do que a versão que usa comparação exata de *strings*. A versão que usa Jaro-Winkler teve uma redução de **79.2%** entre a versão sem blocagem e a versão com blocagem baseada na primeira letra do nome e de **90.3%** entre a versão sem blocagem e a versão com blocagem baseada nas iniciais do nome, enquanto essas reduções foram de **66.8%** e **75.2%** nas versões usando comparação exata.

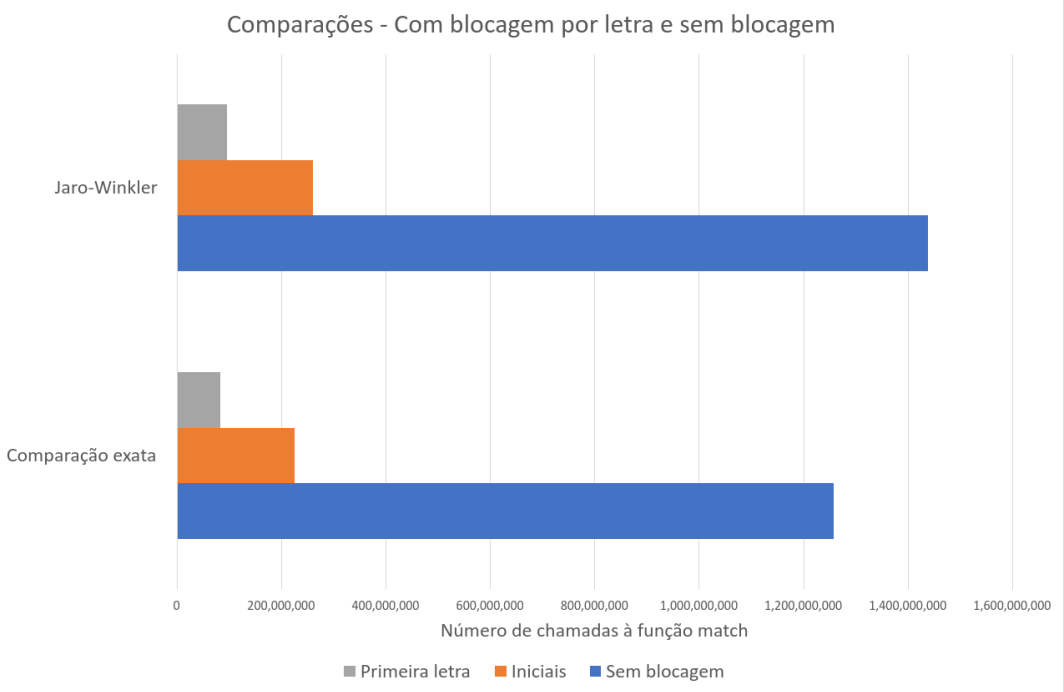
Figura 5.2 – Tempo médio de execução - Com e sem blocagem por letra



Essa diferença não vem do número de comparações, já que é possível ver nas Tabelas 5.8 e 5.9 que o número de comparações foi reduzido de maneira muito semelhante

entre as duas versões, como ilustrado na Figura 5.3. O número de comparações na versão que utiliza o algoritmo Jaro-Winkler foi reduzido em **81.2%** com a aplicação de blocagem baseada nas iniciais do nome da referência e **93.3%** com a aplicação de blocagem baseada somente na primeira letra do nome da referência, enquanto a versão que utiliza comparação exata de *strings* teve uma redução de **82%** e **93.3%**. O número de comparações foi reduzido de maneira quase idêntica em ambas as versões, então este não é o fator que leva à diferença na redução do tempo médio de execução.

Figura 5.3 – Número de comparações - Com e sem blocagem por letra



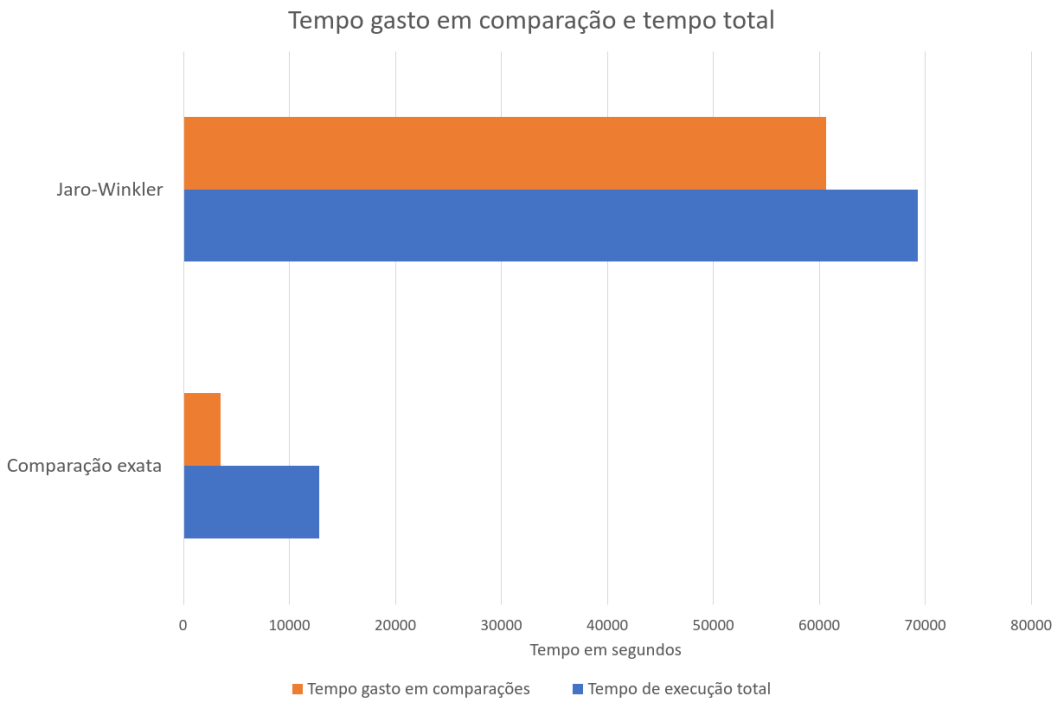
As Tabelas 5.8 e 5.9 também têm outras métricas que ajudam a entender a causa da disparidade nas reduções do tempo médio de cada versão, como o tempo de execução gasto em comparações e tempo total de execução. Quando comparados, é possível ver uma diferença entre qual percentual do tempo de execução é gasto com comparações nas duas versões, como ilustrado na Figura 5.4.

Sem o uso de blocagem, a versão do algoritmo de resolução de entidades que usa Jaro-Winkler usa cerca de **87.5%** do seu tempo de execução fazendo comparações entre referências e entidades enquanto a versão que usa comparação exata de *strings* usa apenas **27.6%** do seu tempo de execução fazendo essas comparações. Isso significa que, supondo

que todas as outras operações do programa continuem exatamente iguais, diminuir o número de comparações da versão que usa Jaro-Winkler pela metade iria reduzir o tempo de execução total em **43.75%**, enquanto a mesma redução no número de comparações iria reduzir em somente **13.8%** o tempo total de execução para a versão que usa comparação exata.

Essa diferença é uma consequência do alto custo da operação de comparação de *strings* usando o algoritmo Jaro-Winkler. O tempo de execução das buscas aos bancos de dados é quase constante, pois estão sendo feitas consultas quase idênticas, para ambos os algoritmos. Isso significa que o aumento do tempo de comparações que é ocasionado com o uso do algoritmo Jaro-Winkler, causa uma mudança na distribuição do trabalho gerado por esse programa, entre o programa em si e o banco de dados.

Figura 5.4 – Tempo gasto em comparações e tempo total - Com e sem blocagem por letra

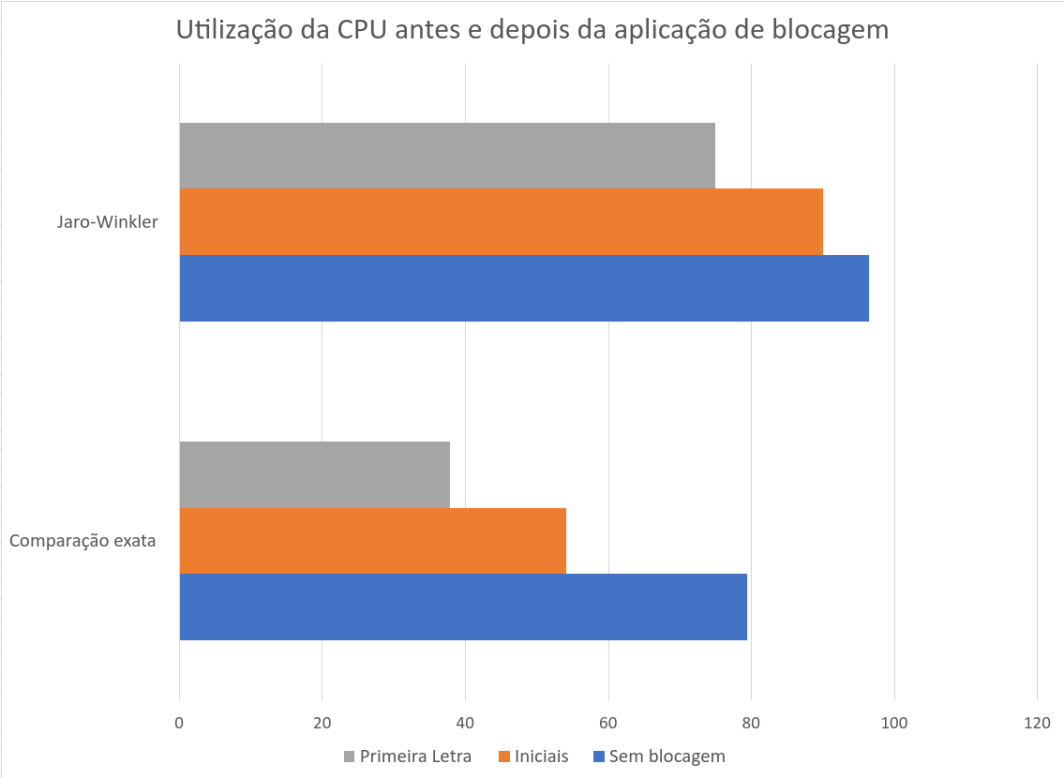


Usando os dados das Tabelas 5.10 e 5.11 é possível ver outra consequência da mudança na distribuição da carga de trabalho. A utilização da CPU pelo programa é uma representação do percentual do seu tempo de execução que foi gasto executando o programa e quanto foi gasto esperando por respostas de outros processos. Quanto mais alta a utilização da CPU, maior o percentual do tempo que foi gasto executando o programa. Pode-se ver nas Tabelas 5.10 e 5.11, e na Figura 5.5 que a versão do programa que usa

o algoritmo Jaro-Winkler passa um percentual muito maior do seu tempo de execução total executando as operações do algoritmo do que a versão que usa comparação exata de *strings*.

O baixo uso do seu tempo de execução total para a execução das operações do programa sugerem que a aplicação de técnicas de computação paralela e assíncrona teria um efeito positivo na performance do programa. Se o programa não precisasse esperar as respostas de uma consulta ao banco de dados para continuar o processamento dos dados, poderia processar muito mais dados em um tempo menor. E a utilização de paralelismo poderia ter um efeito enorme na redução do tempo de execução e poderia ser auxiliado, por exemplo, pela blocagem que leva em consideração a primeira letra do nome da referência para tentar criar operações que possam ser feitas em paralelo. A exploração de paralelismo e computação assíncrona estão fora do escopo deste trabalho e ficam como sugestões para trabalhos posteriores.

Figura 5.5 – Utilização da CPU em porcentagem - Com e sem blocagem por letra



Ainda é necessário levar em consideração as métricas de qualidade, descritas nas Tabelas 5.12, 5.13, 5.14 e 5.15. O aumento da eficiência do algoritmo de resolução de entidades teria pouco valor se para isso fosse necessário reduzir sua qualidade.

As Tabelas 5.12 e 5.13 mostram um cenário muito positivo para as técnicas de blocagem aplicadas até aqui. Para ambas as versões do código, não houve nenhuma alteração na qualidade quando aplicadas ao conjunto de dados de teste. Isso pode levar a conclusão de que essas técnicas podem ser consideradas técnicas de filtragem, visto que não há queda alguma na qualidade, o que seria uma indicação de que todas as comparações que não foram feitas não gerariam um *match* de qualquer maneira. Mas é importante lembrar que este conjunto de dados de teste possui poucas instâncias e é possível ter casos de agrupamentos perdidos aplicando qualquer uma das técnicas exploradas em um conjunto de dados mais amplo.

Para um melhor estudo do efeito dessas técnicas na qualidade do resultado, seria necessário um conjunto de dados de teste maior e mais variado. Para isso, seria necessário estudar várias instâncias do conjunto de dados completo para descobrir seus agrupamentos corretos. Esse esforço está fora do escopo deste trabalho e fica como sugestão para estudos futuros que busquem aprimorar as técnicas de resolução de entidades da abordagem ACERPI para maximizar a qualidade do resultado.

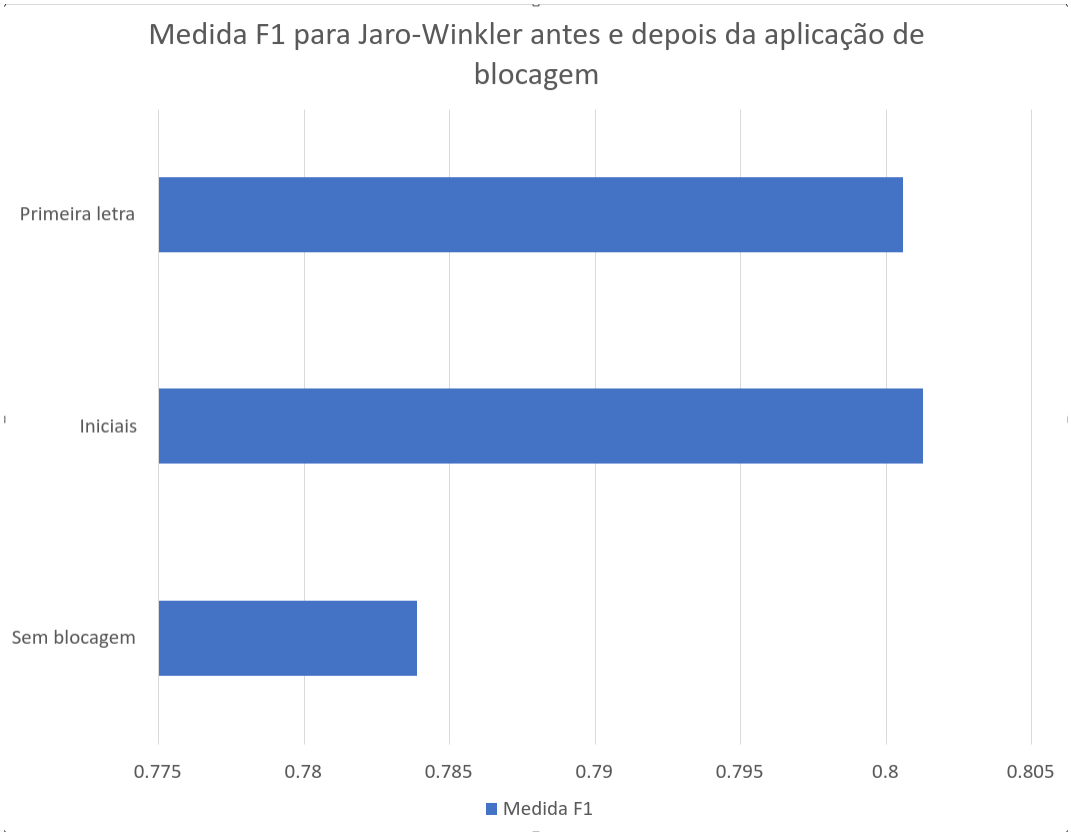
A Tabela 5.14 mostra um cenário parecido com aquele mostrado pelas Tabelas 5.12 e 5.13. Para a versão do programa que usa comparação exata de *strings*, não houve nenhuma alteração na qualidade do resultado que pudesse ser medida pelo programa de avaliação. Para uma análise mais profunda e detalhada dos efeitos das técnicas de blocagem descritas neste trabalho, seria necessário um conjunto de dados de teste mais abrangente.

A Tabela 5.15 tem os resultados mais interessantes sobre o efeito da aplicação das técnicas de blocagem na qualidade dos resultados. Normalmente, em um algoritmo de resolução de entidades bastante eficaz, é esperado que a aplicação de técnicas de blocagem tenham um efeito negativo na qualidade do resultado. No caso do algoritmo apresentado aqui, o contrário é observado. A aplicação das técnicas de blocagem tem um efeito positivo em todas as métricas de qualidade. Isso é devido à comparação usando Jaro-Winkler que usa parâmetros muito permissivos para agrupar as referências. Isso significa que essa versão do código é muito suscetível a ser enganada por ruído. E, quando blocagem é aplicada ao algoritmo, também está sendo criado um filtro de ruído que ajuda o algoritmo a evitar falsos agrupamentos, melhorando a qualidade do resultado.

Também é importante lembrar que a definição da distância mínima entre dois nomes para serem agrupados foi feita antes de ter um algoritmo de avaliação capaz de avaliar o resultado da aplicação dos algoritmos ao conjunto de dados completo. Isso também

serve pra ilustrar a importância de um conjunto de dados de teste amplo e abrangente para não se tirar conclusões equivocadas de resultados não confiáveis.

Figura 5.6 – Medida F1 - Com e sem blocagem por letra - Jaro-Winkler - Conjunto de dados completo



5.5 Experimento 3 - Aplicação de técnicas de blocagem pela primeira letra e tamanho do nome

Nesta seção, são explorados os resultados da execução dos algoritmos de resolução de entidades com a aplicação de técnicas de blocagem baseadas no tamanho do nome da referência sendo analisada, combinadas com as técnicas de blocagem exploradas na Seção 5.4

5.5.1 Resultados

Em toda esta subseção, é usada a seguinte convenção para os nomes das colunas das tabelas:

- A coluna **Sem limites** representa os resultados quando não é aplicado nenhum tipo de blocagem por tamanho do nome da referência.
- A coluna **33% até 300%** representa os resultados da aplicação de blocagem na qual apenas as entidades cujo primeiro nome da lista de nomes tem tamanho **maior que 33% e menor que 300%** do tamanho do nome da referência.
- A coluna **50% até 200%** representa os resultados da aplicação de blocagem na qual apenas as entidades cujo primeiro nome da lista de nomes tem tamanho **maior que 50% e menor que 200%** do tamanho do nome da referência.
- A coluna **66% até 150%** representa os resultados da aplicação de blocagem na qual apenas as entidades cujo primeiro nome da lista de nomes tem tamanho **maior que 66% e menor que 150%** do tamanho do nome da referência.

5.5.2 Métricas de tempo total de execução

A primeira métrica a ser mostrada é a métrica que mede o tempo de execução do algoritmo de resolução de entidades. Essa métrica é coletada pelo programa *time* e é baseada no tempo de dez execuções do programa, com a extensão *cProfile* desabilitada e com o conjunto de dados completo.

A Tabela 5.16 apresenta os resultados da métrica de tempo para o algoritmo de resolução de entidades utilizando **comparação exata de *strings*** com aplicação de blocagem pela **primeira letra** do nome da referência.

Tabela 5.16 – Métricas de tempo - Blocagem pela primeira letra - Com e sem blocagem por tamanho - Comparação exata

Métrica de tempo	Sem limites	33% até 300%	50% até 200%	66% até 150%
Média	1882.3s	1749.8s	1619.5s	1562.3s
Máximo	1976.4s	1764.0s	1629.2s	1571.7s
Mínimo	1742.0s	1740.3s	1609.5s	1555.0s
Desvio padrão	86.0s	7.0s	6.3s	4.7s

A Tabela 5.17 apresenta os resultados da métrica de tempo para o algoritmo de resolução de entidades utilizando **comparação exata de *strings*** com aplicação de blocagem

pelas **iniciais** do nome da referência.

Tabela 5.17 – Métricas de tempo - Blocagem pelas iniciais - Com e sem blocagem por tamanho - Comparação exata

Métrica de tempo	Sem limites	33% até 300%	50% até 200%	66% até 150%
Média	2519.6s	2431.0s	2188.5s	1916.7s
Máximo	2562.9s	2565.3s	2202.3s	1925.6s
Mínimo	2458.4s	2325.0s	2178.0s	1910.2s
Desvio padrão	32.1s	101.4s	7.7s	5.9s

A Tabela 5.18 apresenta os resultados da métrica de tempo para o algoritmo de resolução de entidades utilizando o algoritmo **Jaro-Winkler** para comparações de *strings* com aplicação de blocagem pela **primeira letra** do nome da referência.

Tabela 5.18 – Métricas de tempo - Blocagem pela primeira letra - Com e sem blocagem por tamanho - Jaro-Winkler

Métrica de tempo	Sem limites	33% até 300%	50% até 200%	66% até 150%
Média	4330.8s	3808.5s	3473.4s	2705.8s
Máximo	4353.0s	3859.0s	3563.1s	2753.5s
Mínimo	4300.0s	3752.0s	3416.0s	2693.5s
Desvio padrão	20.8s	31.6s	50.6s	17.9s

A Tabela 5.19 apresenta os resultados da métrica de tempo para o algoritmo de resolução de entidades utilizando o algoritmo **Jaro-Winkler** para comparações de *strings* com aplicação de blocagem pelas **iniciais** do nome da referência.

Tabela 5.19 – Métricas de tempo - Blocagem pelas iniciais - Com e sem blocagem por tamanho - Jaro-Winkler

Métrica de tempo	Sem limites	33% até 300%	50% até 200%	66% até 150%
Média	9264.6s	8929.4s	7948.3s	6333.6s
Máximo	10190.0s	9148.0s	8045.0s	6523.0s
Mínimo	8935.0s	8732.0s	7797.0s	6263.0s
Desvio padrão	430.9s	149.6s	78.7s	74.5s

5.5.3 Métricas sobre as operações de comparação

As métricas mostradas nas Tabelas 5.20, 5.21, 5.22 e 5.23 são referentes à função *match* descrita na Seção 4.3. A linha **Comparações** descreve o número de vezes que a função *match* for chamada durante a execução do algoritmo. A linha **Tempo das comparações** descreve o tempo total que foi gasto durante a execução do programa executando a função *match*. Por último, a linha **Tempo total** representa o tempo de execução do programa de resolução de entidades como um todo. Lembrando que as métricas de tempo

que são descritas aqui não fazem parte do conjunto de métricas de tempo que geraram as Tabelas 5.16, 5.17, 5.18 e 5.19.

A Tabela 5.20 mostra as métricas das operações de comparação para o algoritmo de resolução de entidades que usa **comparação exata de *strings*** com aplicação de blocagem pela **primeira letra** do nome da referência. Essas métricas foram coletadas pela extensão *cProfile* a partir de uma única execução com a extensão habilitada e o conjunto de dados completo.

Tabela 5.20 – Número de comparações - Blocagem pela primeira letra - Com e sem blocagem por tamanho - Comparação exata

Métrica	**Sem limites**	**33% até 300%**	**50% até 200%**	**66% até 150%**
Comparações	84,487,924	78,058,713	64,007,610	43,958,753
Tempo gasto em comparações	248.0s	227.5s	185.1s	127.4s
Tempo Total	2424.8s	2153.7s	1953.6s	1823.4s

A Tabela 5.21 mostra as métricas das operações de comparação para o algoritmo de resolução de entidades que usa **comparação exata de *strings*** com aplicação de blocagem pelas **iniciais** do nome da referência. Essas métricas foram coletadas pela extensão *cProfile* a partir de uma única execução com a extensão habilitada e o conjunto de dados completo.

Tabela 5.21 – Número de comparações - Blocagem pelas iniciais - Com e sem blocagem por tamanho - Comparação exata

Métrica	**Sem limites**	**33% até 300%**	**50% até 200%**	**66% até 150%**
Comparações	226,749,153	210,091,823	179,150,783	130,962,611
Tempo gasto em comparações	664.6s	615.4s	517.8s	377.6s
Tempo Total	3449.7s	3445.4s	2987.5s	2524.2s

A Tabela 5.22 mostra as métricas das operações de comparação para o algoritmo de resolução de entidades que usa o algoritmo **Jaro-Winkler para comparação de *strings*** com aplicação de blocagem pela **primeira letra** do nome da referência. Essas métricas foram coletadas pela extensão *cProfile* a partir de uma única execução com a extensão habilitada e o conjunto de dados completo.

Tabela 5.22 – Número de comparações - Blocagem pela primeira letra - Com e sem blocagem por tamanho - Jaro-Winkler

Métrica	**Sem limites**	**33% até 300%**	**50% até 200%**	**66% até 150%**
Comparações	96,432,691	90,408,057	72,704,794	48,783,963
Tempo gasto em comparações	4077.3s	3785.1s	3174.7s	2187.8s
Tempo Total	6112.8s	5534.1s	4930.1s	3655.2s

A Tabela 5.23 mostra as métricas das operações de comparação para o algoritmo de resolução de entidades que usa o algoritmo **Jaro-Winkler para comparação**

de *strings* com aplicação de blocagem pelas **iniciais** do nome da referência. Essas métricas foram coletadas pela extensão *cProfile* a partir de uma única execução com a extensão habilitada e o conjunto de dados completo.

Tabela 5.23 – Número de comparações - Blocagem pelas iniciais - Com e sem blocagem por tamanho - Jaro-Winkler

Métrica	**Sem limites**	**33% até 300%**	**50% até 200%**	**66% até 150%**
Comparações	261,655,437	245,840,640	203,783,357	146,326,897
Tempo gasto em comparações	11553.2s	10751.7s	9313.9s	7007.6s
Tempo Total	14474.1s	13229.4s	11992.9s	9171.9s

5.5.4 Métricas de utilização de CPU

As Tabelas 5.24, 5.25, 5.26 e 5.27 apresentam métricas referentes à utilização da CPU durante a execução do programa de resolução de entidades. A utilização da CPU pelo programa é uma representação do percentual do seu tempo de execução que foi gasto executando o programa e quanto foi gasto esperando por respostas de outros processos. Quanto mais alta a utilização da CPU, maior o percentual do tempo que foi gasto executando o programa. Estas métricas foram coletadas usando o programa *time* a partir de dez execuções do programa de resolução de entidades com a extensão *cProfile* desabilitada e o conjunto completo de dados.

A Tabela 5.24 apresenta as métricas referentes à utilização da CPU para o algoritmo de resolução de entidades que usa **comparação exata de *strings*** com blocagem pela **primeira letra** do nome da referência.

Tabela 5.24 – Utilização de CPU - Blocagem pela primeira letra - Com e sem blocagem por tamanho - Comparação exata

Métrica de Utilização de CPU	**Sem limites**	**33% até 300%**	**50% até 200%**	**66% até 150%**
Média	37.9%	35.0%	33.0%	28.0%
Máximo	39.0%	35.0%	33.0%	28.0%
Mínimo	37.0%	35.0%	33.0%	28.0%
Desvio padrão	0.9%	0.0%	0.0%	0%

A Tabela 5.25 apresenta as métricas referentes à utilização da CPU para o algoritmo de resolução de entidades que usa **comparação exata de *strings*** com blocagem pelas **iniciais** do nome da referência.

A Tabela 5.26 apresenta as métricas referentes à utilização da CPU para o algoritmo de resolução de entidades que usa o algoritmo **Jaro-Winkler** para comparação de *strings* com blocagem pela **primeira letra** do nome da referência.

Tabela 5.25 – Utilização de CPU - Blocagem pelas iniciais - Com e sem blocagem por tamanho - Comparação exata

Métrica de Utilização de CPU	Sem limites	33% até 300%	50% até 200%	66% até 150%
Média	54.2%	51.4%	49.1%	44.0%
Máximo	55.0%	52.0%	50.0%	44.0%
Mínimo	54.0%	50.0%	49.0%	44.0%
Desvio padrão	0.4%	0.7%	0.3%	0.0%

Tabela 5.26 – Utilização de CPU - Blocagem pela primeira letra - Com e sem blocagem por tamanho - Jaro-Winkler

Métrica de Utilização de CPU	Sem limites	33% até 300%	50% até 200%	66% até 150%
Média	75.0%	74.3%	69.9%	65.1%
Máximo	75.0%	75.0%	70.0%	66.0%
Mínimo	75.0%	74.0%	69.0%	65.0%
Desvio padrão	0.0%	0.5%	0.3%	0.3%

A Tabela 5.27 apresenta as métricas referentes à utilização da CPU para o algoritmo de resolução de entidades que usa o algoritmo **Jaro-Winkler** para comparação *strings* com blocagem pelas **iniciais** do nome da referência.

Tabela 5.27 – Utilização de CPU - Blocagem pelas iniciais - Com e sem blocagem por tamanho - Jaro-Winkler

Métrica de Utilização de CPU	Sem limites	33% até 300%	50% até 200%	66% até 150%
Média	90.0%	87.6%	85.5%	82.1%
Máximo	91.0%	88.0%	86.0%	83.0%
Mínimo	87.0%	87.0%	85.0%	82.0%
Desvio padrão	1.4%	0.5%	0.5%	0.3%

5.5.5 Métricas de qualidade usando o conjunto de dados de teste

As Tabelas 5.28, 5.29, 5.30, e 5.31 apresentam as métricas de qualidade descritas na Seção 4.5. Essas métricas foram calculadas a partir do programa de avaliação, também descrito na Seção 4.5, após a execução do programa de resolução de entidades com o conjunto de dados de teste.

A Tabela 5.28 apresenta as métricas de qualidade para o algoritmo de resolução de entidades usando **comparação exata de *strings*** com blocagem pela **primeira letra** do nome da referência aplicado ao **conjunto de dados de teste**.

A Tabela 5.29 apresenta as métricas de qualidade para o algoritmo de resolução de entidades usando **comparação exata de *strings*** com blocagem pelas **iniciais** do nome da referência aplicado ao **conjunto de dados de teste**.

A Tabela 5.30 apresenta as métricas de qualidade para o algoritmo de resolução de entidades usando o algoritmo **Jaro-Winkler** para comparação de *strings* com blocagem

Tabela 5.28 – Métricas de qualidade - Blocagem pela primeira letra - Com e sem blocagem por tamanho - Comparação exata - Conjunto de dados de teste

Métrica	Sem limites	33% até 300%	50% até 200%	66% até 150%
Verdadeiros Positivos	11597	11596	11388	11388
Falsos Positivos	0	0	0	0
Falsos Negativos	2481	2482	2690	2690
Precisão	100.00%	100.00%	100.00%	100.00%
Revocação	82.38%	82.37%	80.89%	80.89%
Medida F1	90.34%	90.33%	89.44%	89.44%

Tabela 5.29 – Métricas de qualidade - Blocagem pelas iniciais - Com e sem blocagem por tamanho - Comparação exata - Conjunto de dados de teste

Métrica	Sem limites	33% até 300%	50% até 200%	66% até 150%
Verdadeiros Positivos	11597	11596	11388	11388
Falsos Positivos	0	0	0	0
Falsos Negativos	2481	2482	2690	2690
Precisão	100.00%	100.00%	100.00%	100.00%
Revocação	82.38%	82.37%	80.89%	80.89%
Medida F1	90.34%	90.33%	89.44%	89.44%

pela **primeira letra** do nome da referência aplicado ao **conjunto de dados de teste**.

Tabela 5.30 – Métricas de qualidade - Blocagem pela primeira letra - Com e sem blocagem por tamanho - Jaro-Winkler - Conjunto de dados de teste

Métrica	Sem limites	33% até 300%	50% até 200%	66% até 150%
Verdadeiros Positivos	13280	13279	12996	12287
Falsos Positivos	0	0	0	0
Falsos Negativos	798	799	1082	1791
Precisão	100.00%	100.00%	100.00%	100.00%
Revocação	94.33%	94.32%	92.31%	87.28%
Medida F1	97.08%	97.08%	96.00%	93.21%

A Tabela 5.31 apresenta as métricas de qualidade para o algoritmo de resolução de entidades usando o algoritmo **Jaro-Winkler** para comparação de *strings* com blocagem pelas **iniciais** do nome da referência aplicado ao **conjunto de dados de teste**.

Tabela 5.31 – Métricas de qualidade - Blocagem pelas iniciais - Com e sem blocagem por tamanho - Jaro-Winkler - Conjunto de dados de teste

Métrica	Sem limites	33% até 300%	50% até 200%	66% até 150%
Verdadeiros Positivos	13280	13279	12996	12287
Falsos Positivos	0	0	0	0
Falsos Negativos	798	799	1082	1791
Precisão	100.00%	100.00%	100.00%	100.00%
Revocação	94.33%	94.32%	92.31%	87.28%
Medida F1	97.08%	97.08%	96.00%	93.21%

5.5.6 Métricas de qualidade usando o conjunto de dados completo

As Tabelas 5.32, 5.33, 5.34, e 5.35 apresentam as métricas de qualidade descritas na Seção 5.2.6. Essas métricas foram calculadas a partir do programa de avaliação, também descrito na Seção 5.2.6, após a execução do programa de resolução de entidades com o conjunto de dados completo.

A Tabela 5.32 apresenta as métricas de qualidade para o algoritmo de resolução de entidades usando **comparação exata de *strings*** com blocagem pela **primeira letra** do nome da referência aplicado ao **conjunto de dados completo**.

Tabela 5.32 – Métricas de qualidade - Blocagem pela primeira letra - Com e sem blocagem por tamanho - Comparação exata - Conjunto de dados completo

Métrica	**Sem limites**	**33% até 300%**	**50% até 200%**	**66% até 150%**
Verdadeiros Positivos	11597	11596	11388	11388
Falsos Positivos	0	0	0	0
Falsos Negativos	2481	2482	2690	2690
Precisão	100.00%	100.00%	100.00%	100.00%
Revocação	82.38%	82.37%	80.89%	80.89%
Medida F1	90.34%	90.33%	89.44%	89.44%

A Tabela 5.33 apresenta as métricas de qualidade para o algoritmo de resolução de entidades usando **comparação exata de *strings*** com blocagem pelas **iniciais** do nome da referência aplicado ao **conjunto de dados completo**.

Tabela 5.33 – Métricas de qualidade - Blocagem pelas iniciais - Com e sem blocagem por tamanho - Comparação exata - Conjunto de dados completo

Métrica	**Sem limites**	**33% até 300%**	**50% até 200%**	**66% até 150%**
Verdadeiros Positivos	11597	11596	11388	11388
Falsos Positivos	0	0	0	0
Falsos Negativos	2481	2482	2690	2690
Precisão	100.00%	100.00%	100.00%	100.00%
Revocação	82.38%	82.37%	80.89%	80.89%
Medida F1	90.34%	90.33%	89.44%	89.44%

A Tabela 5.34 apresenta as métricas de qualidade para o algoritmo de resolução de entidades usando o algoritmo **Jaro-Winkler** para comparação de *strings* com blocagem pela **primeira letra** do nome da referência aplicado ao **conjunto de dados completo**.

A Tabela 5.35 apresenta as métricas de qualidade para o algoritmo de resolução de entidades usando o algoritmo **Jaro-Winkler** para comparação de *strings* com blocagem pelas **iniciais** do nome da referência aplicado ao **conjunto de dados completo**.

Tabela 5.34 – Métricas de qualidade - Blocagem pela primeira letra - Com e sem blocagem por tamanho - Jaro-Winkler - Conjunto de dados completo

Métrica	Sem limites	33% até 300%	50% até 200%	66% até 150%
Verdadeiros Positivos	9406	9244	9610	10749
Falsos Positivos	12	12	0	0
Falsos Negativos	4672	4834	4468	3329
Precisão	99.87%	99.87%	100.00%	100.00%
Revocação	66.81%	65.66%	68.26%	76.35%
Medida F1	80.06%	79.23%	81.14%	86.59%

Tabela 5.35 – Métricas de qualidade - Blocagem pelas iniciais - Com e sem blocagem por tamanho - Jaro-Winkler - Conjunto de dados completo

Métrica	Sem limites	33% até 300%	50% até 200%	66% até 150%
Verdadeiros Positivos	9410	9248	9610	10749
Falsos Positivos	0	0	0	0
Falsos Negativos	4668	4830	4468	3329
Precisão	100.00%	100.00%	100.00%	100.00%
Revocação	66.84%	65.69%	68.26%	76.35%
Medida F1	80.13%	79.29%	81.14%	86.59%

5.5.7 Análise

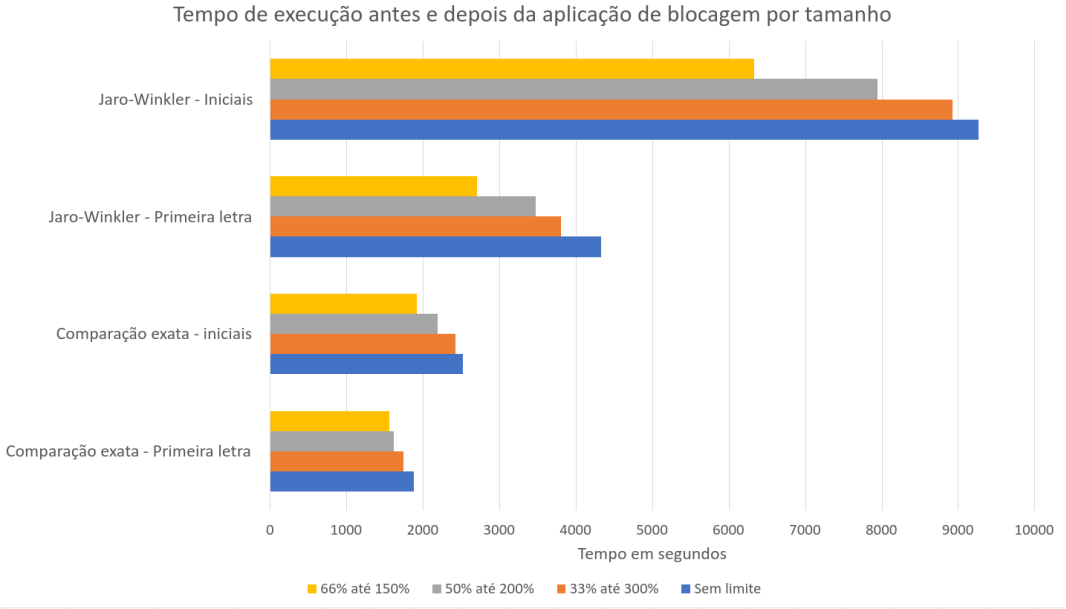
Analisando os tempos de execução do programa antes e depois da aplicação de blocagem pelo tamanho do nome da referência nas Tabelas 5.16, 5.17, 5.18 e 5.19 pode-se ver um impacto positivo, como ilustrado na Figura 5.7. Também pode-se ver na Tabela 5.36 que o comportamento é consistente com o esperado, em que quanto mais restritivo é a condição da blocagem, maior é a redução no tempo médio. Também é possível ver o mesmo efeito que foi visto na Seção 5.4.7, onde o algoritmo tem uma redução maior do seu tempo de execução quando usa o algoritmo Jaro-Winkler para fazer a comparação entre os nomes.

Tabela 5.36 – Redução em percentual do tempo médio de execução - Com e sem blocagem por tamanho

Versão	33% até 300%	50% até 200%	66% até 150%
Comparação exata - Primeira letra	7.04%	13.96%	17.00%
Comparação exata - Iniciais	3.52%	13.14%	23.92%
Jaro-Winkler - Primeira letra	12.06%	19.80%	37.52%
Jaro-Winkler - Iniciais	3.62%	14.21%	31.63%

A Figura 5.8 e a Tabela 5.37, baseadas nos resultados mostrados nas Tabelas 5.20, 5.21, 5.22 e 5.23, corroboram a conclusão de que o maior fator para a diferença no impacto da aplicação de blocagem pelo tamanho do nome da referência no tempo total de

Figura 5.7 – Tempo de execução - Com e sem blocagem por tamanho



execução é o percentual do tempo de execução que é gasto executando comparações, pois a redução em comparações é bastante similar entre todas as versões do algoritmo.

Figura 5.8 – Número de comparações - Com e sem blocagem por tamanho

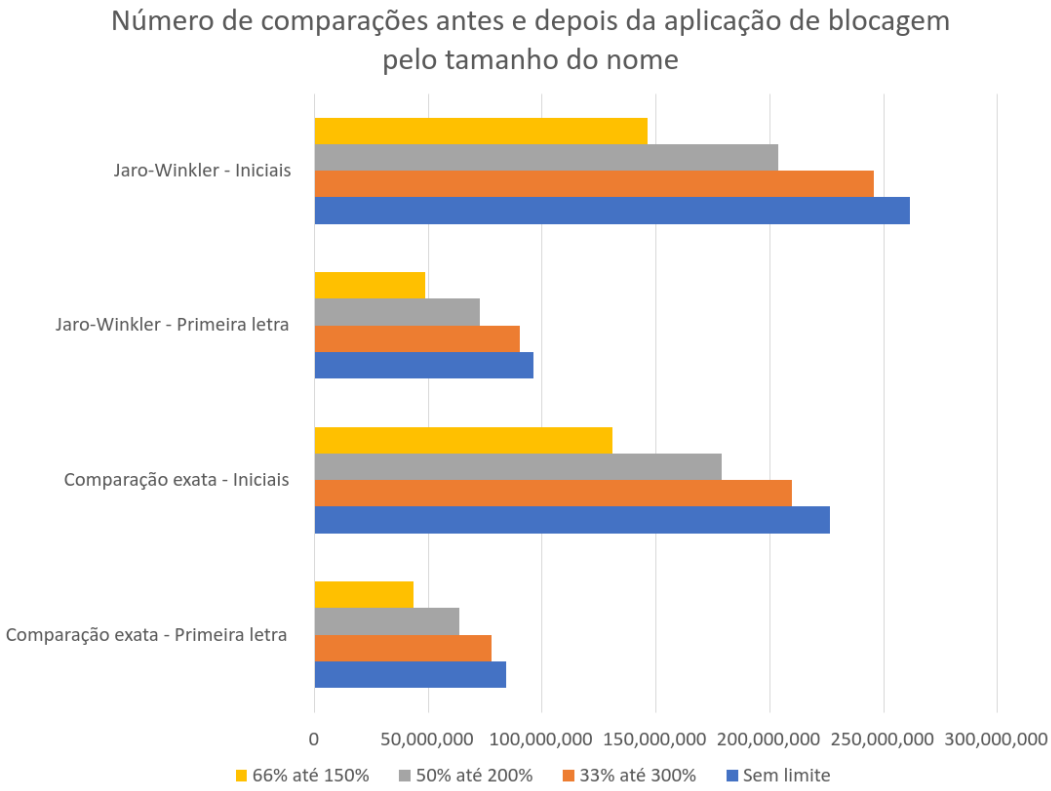
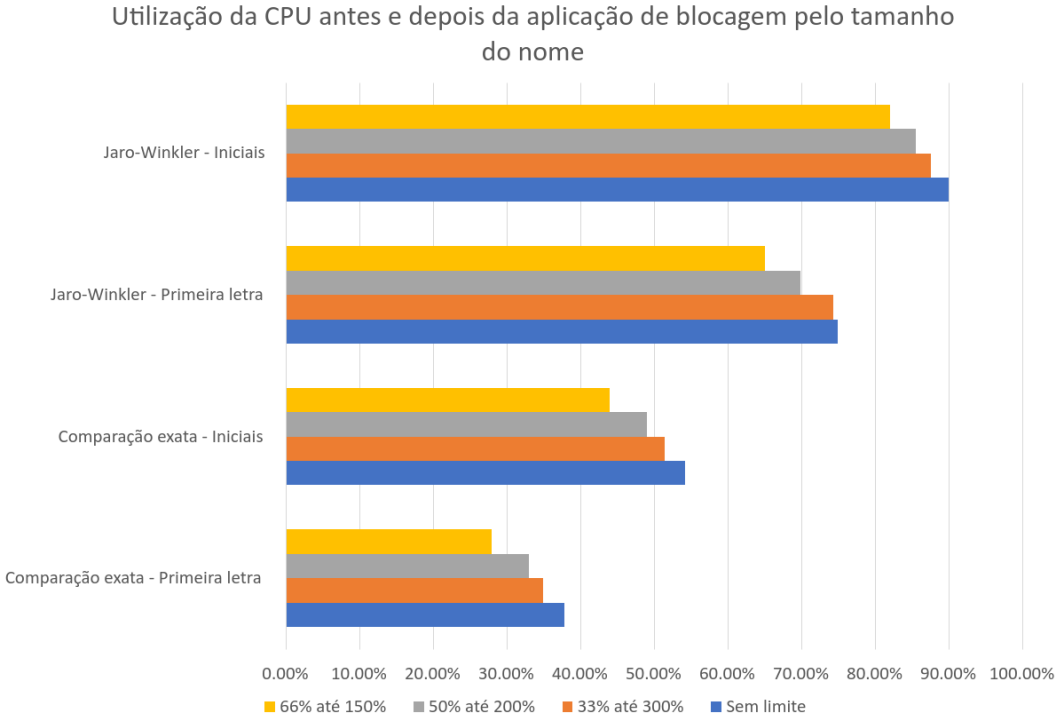


Tabela 5.37 – Redução em percentual do número de comparações - Com e sem blocagem por tamanho

Versão	33% até 300%	50% até 200%	66% até 150%
Comparação exata - Primeira letra	7.61%	24.24%	47.97%
Comparação exata - Iniciais	7.35%	20.99%	42.24%
Jaro-Winkler - Primeira letra	6.25%	24.61%	49.41%
Jaro-Winkler - Iniciais	6.04%	22.12%	44.08%

A partir das Tabelas 5.24, 5.25, 5.26 e 5.27 pode-se observar que, conforme as regras da blocagem pelo tamanho do nome da referência ficam mais restritivas, o banco de dados começa a se tornar um gargalo para o programa. A Figura 5.9 mostra essa tendência. Pode-se observar que a partir da aplicação da blocagem pelo tamanho do nome da referência com limites de 50% e 200% ambas versões do algoritmo que usam comparação exata passam mais tempo esperando resposta do banco de dados do que executando o programa. Estas versões têm excelente potencial de melhora na eficiência através da aplicação de computação paralela e assíncrona, pois estão muito abaixo da utilização ideal da CPU.

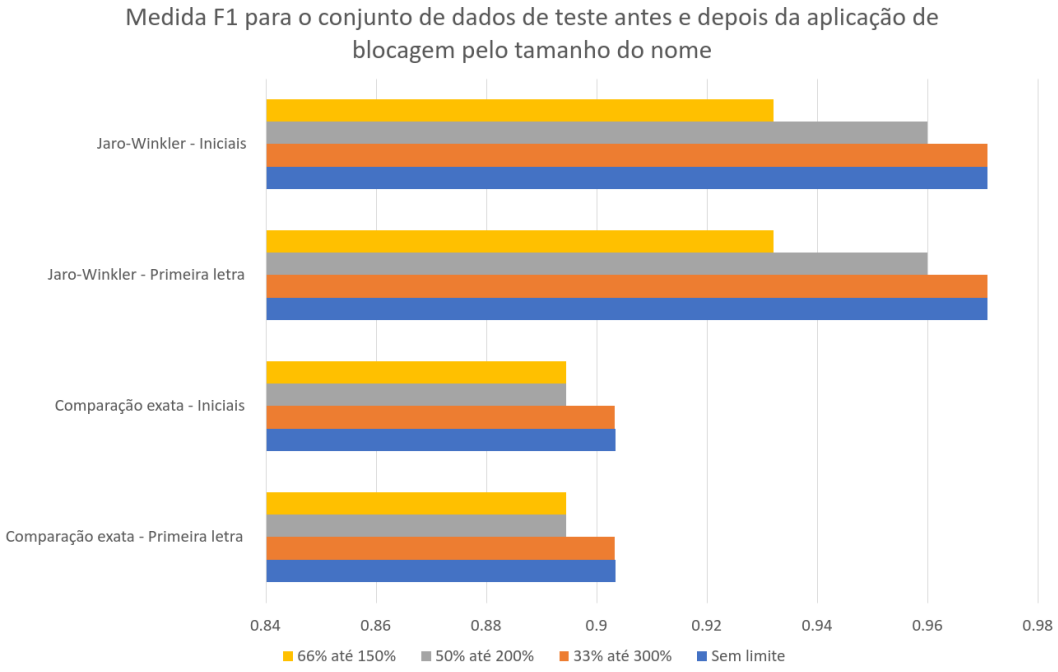
Figura 5.9 – Utilização da CPU em porcentagem - Com e sem blocagem por tamanho



A partir das Tabelas 5.28, 5.29, 5.30 e 5.31 é possível ver que existe um custo para a aplicação da blocagem pelo tamanho do nome da referência. A queda em qualidade é

provavelmente causada pelo descarte de comparações que gerariam agrupamentos. Esse resultado é pior, em termos de qualidade, que a blocagem baseada na primeira letra do primeiro nome da entidade. A Figura 5.10 ilustra a queda consistente na medida F1 para o conjunto de dados de testes conforme as regras da aplicação da blocagem por tamanho do nome fica mais restritiva.

Figura 5.10 – Medida F1 - Conjunto de dados de teste - Com e sem blocagem por tamanho

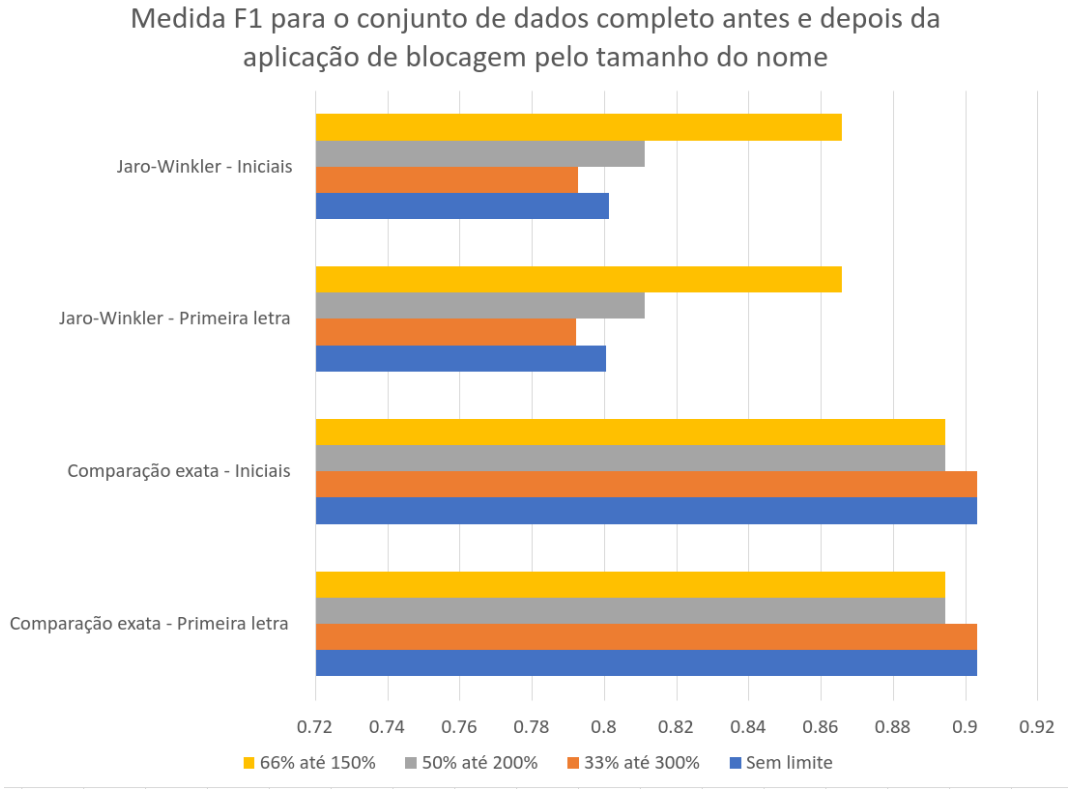


A partir das Tabelas 5.32, 5.33, 5.34 e 5.35 pode-se ver que, assim como visto na Seção 5.4.7, a aplicação de blocagem mais restritiva causa um aumento na medida F1 para os algoritmos que usam Jaro-Winkler. Isso é devido ao fato que a calibração dos limites da distância gerada pelo algoritmo Jaro-Winkler que geram um agrupamento está muito permissiva e as técnicas de blocagem agem como redução de ruído, o que evita muitos dos agrupamentos equivocados que seriam gerados pelo algoritmo.

Para o algoritmo que usa comparação exata de *strings*, o resultado é o mesmo que foi visto para o conjunto de dados de teste. Isso se deve ao fato de que o algoritmo que usa comparação exata de *strings* é muito mais restritivo e, portanto, muito mais resistente à ruído. Neste caso, a medida F1 tem uma leve queda, pois a aplicação de blocagem pelo tamanho do nome não evita nenhum agrupamento indevido, mas elimina alguns agrupamentos corretos.

A Figura 5.11 mostra o impacto da aplicação de blocagem pelo tamanho do nome da referência na medida F1 quando aplicado ao conjunto de dados completo

Figura 5.11 – Medida F1 - Conjunto de dados completo - Com e sem blocagem por tamanho



5.6 Considerações Finais

Neste capítulo, foram apresentados todos os experimentos realizados durante o desenvolvimento deste trabalho e a análise dos seus resultados. O primeiro conjunto de experimentos teve como objetivo analisar a eficiência e a eficácia dos algoritmos de resolução de entidades que foram usados para analisar o efeito da aplicação de diferentes tipos de blocagem. O segundo conjunto de experimentos teve como objetivo analisar o impacto da aplicação de técnicas de blocagem baseadas na primeira letra do primeiro nome das entidades conhecidas. O terceiro, e último, conjunto de experimentos teve como objetivo analisar o impacto da aplicação de técnicas de blocagem baseadas no tamanho do nome das referências.

6 CONCLUSÃO

Neste trabalho, foi explorada a aplicação de diversas técnicas de blocagem ao algoritmo de resolução de entidades da abordagem ACERPI. O efeito de cada uma dessas técnicas na eficiência e eficácia desses algoritmos foi analisada através de métricas de tempo de execução, número de comparações, utilização de CPU e métricas de qualidade extraídos a partir de múltiplos experimentos realizados ao decorrer do desenvolvimento deste trabalho.

As principais contribuições deste trabalho foram o desenvolvimento e análise detalhada da eficiência e eficácia de versões otimizadas, através da aplicação de técnicas de blocagem, do algoritmo de resolução de entidades ACERPI, que permite que o programa seja executado em uma fração do tempo que a versão original levava, possibilitando o uso desse algoritmos para conjuntos de dados maiores e mais complexos, e também o desenvolvimento de um algoritmo de avaliação do resultado do algoritmo de resolução de entidade da abordagem ACERPI que é mais flexível e eficiente.

Por fim, também foram identificados alguns pontos de melhorias para a abordagem ACERPI e na ACERPI-Block que podem ser explorados em trabalhos futuros. São estes:

- O aperfeiçoamento da etapa de extração de dados para que durante a etapa de resolução de entidades tenham mais parâmetros disponíveis para aplicação de técnicas de blocagem e filtragem.
- Exploração de computação paralela e assíncrona como otimização para resolução de entidades com blocagem.

REFERÊNCIAS

BENJELLOUN, O. et al. Swoosh: a generic approach to entity resolution. **VLDB J.**, v. 18, n. 1, p. 255–276, 2009. Disponível em: <https://doi.org/10.1007/s00778-008-0098-x>.

EICH, L. Acerpi-link: Uma alternativa de resolução de entidades em portarias institucionais utilizando linkagem de registros. 2021.

HERNÁNDEZ, M. A.; STOLFO, S. J. Real-world data is dirty: Data cleansing and the merge/purge problem. **Data Min. Knowl. Discov.**, v. 2, n. 1, p. 9–37, 1998. Disponível em: <https://doi.org/10.1023/A:1009761603038>.

IFRS. Documentos. 2020. Disponível em: <https://ifrs.edu.br/documentos/>.

IFRS, Campus Ibirubá. Boletins de Serviço. 2011. Disponível em: <https://ibiruba.ifrs.edu.br/site/conteudo.php?cat=50>.

IFRS, Campus Ibirubá. Boletim de Serviço. 2018. Disponível em: <https://ifrs.edu.br/ibiruba/documentosoficiais/boletim-de-servico/>.

KÖPCKE, H.; RAHM, E. Frameworks for entity matching: A comparison. **Data Knowl. Eng.**, v. 69, n. 2, p. 197–210, 2010. Disponível em: <https://doi.org/10.1016/j.datak.2009.10.003>.

MAIDASANI, H. et al. Entity resolution evaluation measures. 2012.

MENESTRINA, D.; WHANG, S.; GARCIA-MOLINA, H. Evaluating entity resolution results. **Proc. VLDB Endow.**, v. 3, n. 1, p. 208–219, 2010. Disponível em: <http://www.vldb.org/pvldb/vldb2010/pvldb_vol3/R18.pdf>.

MICROSOFT. O que são os bancos de dados nosql. 2021. Disponível em: <https://azure.microsoft.com/pt-br/overview/nosql-database/>.

MONGODB. The database for modern applications. 2020. Disponível em: <https://www.mongodb.com/>.

MONGODB. What is a Document Database? 2020. Disponível em: <https://www.mongodb.com/document-databases>.

Palmieri Lage, J. et al. Automatic generation of agents for collecting hidden web pages for data extraction. **Data Knowledge Engineering**, v. 49, n. 2, p. 177 – 196, 2004. ISSN 0169-023X. Web Information and Data Management. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0169023X03001769>.

PAPADAKIS, G. et al. A survey of blocking and filtering techniques for entity resolution. **CoRR**, abs/1905.06167, 2019. Disponível em: <http://arxiv.org/abs/1905.06167>.

PARMAR, N. Why python cprofile is the recommended profiling interface. 2021. Disponível em: <https://stackify.com/why-python-cprofile-is-the-recommended-profiling-interface>.

PERVEEZ, S. H. What is google cloud platform? 2021. Disponível em: <https://www.simplilearn.com/google-cloud-platform-article>.

SCHMITZ, C. et al. Acerpi: An approach for ordinances collection, information extraction and entity resolution. In: **Anais do XXXV Simpósio Brasileiro de Bancos de Dados, SBBD 2021, online, October 4 - October 8, 2021**. [S.l.]: SBC, 2021.

ScrapingHub. What is web scraping? 2020. Disponível em: <https://www.scrapinghub.com/what-is-web-scraping/>.

UFRGS. Consulta a Portarias geradas pela Reitoria da UFRGS. 2016. Disponível em: <https://www1.ufrgs.br/sistemas/sde/gerencia-documentos/index.php/publico/consultar/>.