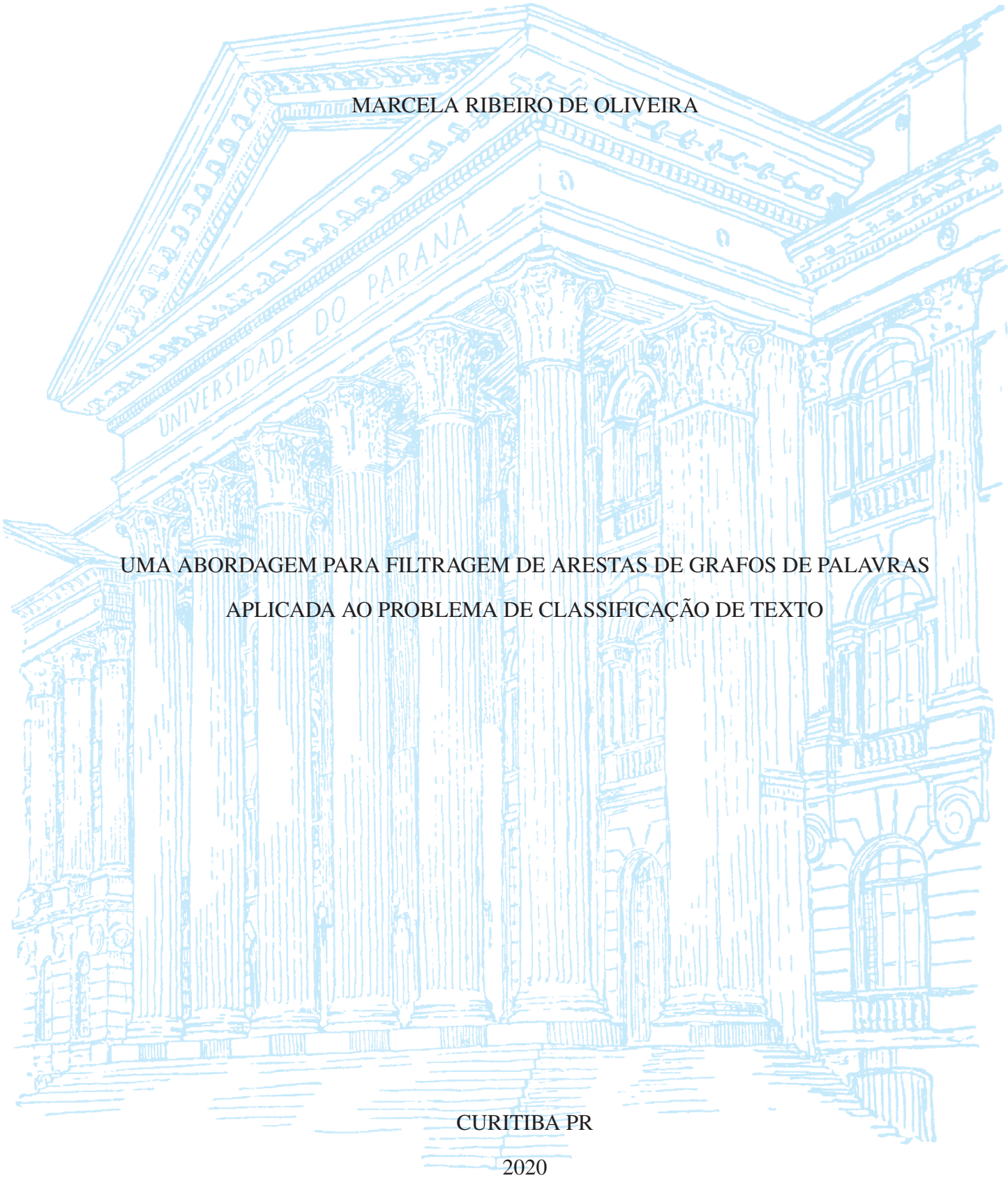
UNIVERSIDADE FEDERAL DO PARANÁ

MARCELA RIBEIRO DE OLIVEIRA



UMA ABORDAGEM PARA FILTRAGEM DE ARESTAS DE GRAFOS DE PALAVRAS APLICADA AO PROBLEMA DE CLASSIFICAÇÃO DE TEXTO

CURITIBA PR

2020

MARCELA RIBEIRO DE OLIVEIRA

UMA ABORDAGEM PARA FILTRAGEM DE ARESTAS DE GRAFOS DE PALAVRAS APLICADA AO PROBLEMA DE CLASSIFICAÇÃO DE TEXTO

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Informática no Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação*.

Orientador: Eduardo Jaques Spinosa.

CURITIBA PR

2020

Catalogação na Fonte: Sistema de Bibliotecas, UFPR
Biblioteca de Ciência e Tecnologia

O48a Oliveira, Marcela Ribeiro de
Uma abordagem para filtragem de arestas de grafos de palavras aplicada ao problema de classificação de texto [recurso eletrônico] / Marcela Ribeiro de Oliveira. – Curitiba, 2020.

Dissertação - Universidade Federal do Paraná, Setor de Ciências Exatas, Programa de Pós-Graduação em Informática, 2020.

Orientador: Eduardo Jaques Spinosa

1. Teoria dos grafos. 2. Processamento eletrônico de dados. 3. Algoritmos. I. Universidade Federal do Paraná. II. Spinosa, Eduardo Jaques. III. Título.

CDD: 511.5

Bibliotecário: Elias Barbosa da Silva CRB-9/1894

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em INFORMÁTICA da Universidade Federal do Paraná foram convocados para realizar a arguição da dissertação de Mestrado de **MARCELA RIBEIRO DE OLIVEIRA** intitulada: **Uma abordagem para filtragem de arestas de grafos de palavras aplicada ao problema de classificação de texto**, sob orientação do Prof. Dr. EDUARDO JAQUES SPINOSA, que após terem inquirido a aluna e realizada a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.

A outorga do título de mestre está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

CURITIBA, 09 de Dezembro de 2020.

Assinatura Eletrônica
09/12/2020 18:06:19.0
EDUARDO JAQUES SPINOSA
Presidente da Banca Examinadora (UNIVERSIDADE FEDERAL DO PARANÁ)

Assinatura Eletrônica
09/12/2020 16:11:41.0
EDUARDO TODT
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

Assinatura Eletrônica
09/12/2020 17:31:21.0
MYRIAM REGATTIERI DE BIASE DA SILVA DELGADO
Avaliador Externo (UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ)

Tudo quanto fizerdes, fazei-o de todo o coração, como para o Senhor e não para homens. (Colossenses 3:23)

AGRADECIMENTOS

A Deus por seu infinito amor, seu cuidado e pela graça imerecida.

Ao meu pai Manoel Antonio de Oliveira e a minha mãe Josefa Ribeiro de Oliveira, vocês são incríveis de uma maneira que eu não consigo expressar em palavras. Eu sei que nunca vou conseguir retribuir tudo o que vocês fizeram e fazem por mim mas saibam que a minha gratidão é imensa.

Ao meu orientador por todo o tempo disponibilizado para orientação, todos os esclarecimentos e por sempre me motivar a seguir em frente.

As minhas amigas Alane, Alissar, Ana Bovs, Cristina e Thainá do famoso grupinho "Bebê Daltônico", pelos momentos divertidos, lanches, doces, cinemas, podcasts e tudo mais. Vocês foram e continuam sendo muito importantes para mim e eu amo muito cada uma de vocês e tenho muito orgulho das mulheres que vocês são.

Aos meus líderes de célula e grandes amigos Linecker Amorim, Nayara Amorim e Jéssica Barros por sempre se importarem com como estava meu mestrado e sempre orarem por mim durante os momentos fáceis e também durante os momentos difíceis.

Aos meus colegas de laboratório pelo companheirismo, as conversas, disponibilidade para ouvir ideias e todo o tempo compartilhado.

A todos os meus amigos pela amizade e pelo apoio durante todo o período de mestrado.

Finalmente, mas de forma nenhuma menos importante, agradeço ao CNPq pelo auxílio financeiro durante o desenvolvimento desse trabalho.

RESUMO

Classificação de texto é um problema clássico na área de Processamento de Linguagem Natural. Uma tarefa essencial na classificação de texto é a construção da representação, que deve prover informações relevantes para o classificador. Um dos modelos de representação mais efetivos utiliza grafos para representar textos. Esta pesquisa propõe uma abordagem que utiliza esse modelo de representação e medidas de associatividade de palavras para incorporar mais informações aos grafos. Essas medidas servem como um guia para identificar e remover arestas entre as palavras com baixo valor de associatividade. Então, utilizando o node2vec, extraímos as características de cada grafo e utilizamos uma rede neural convolucional de texto para realizar a classificação. Conduzimos experimentos para comparar diferentes tipos de modelagem dos grafos em termos de acerto na classificação e da proporção de arestas que foram removidas. Os resultados obtidos indicaram que essa abordagem torna possível reduzir a quantidade de arestas no grafo mantendo o desempenho da classificação.

Palavras-chave: classificação de texto, grafo de palavras, aprendizagem de representação de grafos.

ABSTRACT

Text classification is a classic problem in Natural Language Processing. An essential task in text classification is the construction of the representation, which must provide relevant information to the classifier. One of the most effective representation model uses graphs to represent texts. This research proposes an approach that uses this representation model and word association measures to incorporate more information into the graphs. These measures then serve as a guide to identify and remove edges between words with low association levels. Then, using node2vec, we extract the features of each graph and use a text convolutional neural network for classification. We conducted experiments in order to compare different kinds of graph modeling in terms of classification score and the proportion of edges that were removed. The results obtained indicate that this approach makes it possible to reduce the amount of edges in the graphs maintaining classification performance.

Keywords: text classification, graph of words, graph representation learning.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÔNIMOS

DINF	Departamento de Informática
PPGINF	Programa de Pós-Graduação em Informática
UFPR	Universidade Federal do Paraná
PLN	Processamento de Linguagem Natural
NLP	Natural Language Processing
IMP	Informação Mútua Ponto-a-Ponto
PMI	Pointwise Mutual Information
GP	Grafo de Palavras
GOW	Graph of Words
BoW	Bag of Words
TF-IDF	Term Frequency-Inverse Document Frequency
CNN	Convolutional Neural Network
GNN	Graph Neural Network
GCN	Graph Convolutional Network
SGC	Simple Graph Convolution
SGD	Stochastic Gradient Descent
ReLU	Rectified Linear Unit
SVM	Suporte Vector Machine
KNN	K-Nearest Neighbors
CBOW	Continuous Bag of Words
LSTM	Long Short-Term Memory
VSM	Vector Space Model
OCR	Optical Character Recognition
MPM	Message Passing Mechanism

LISTA DE SÍMBOLOS

α	alfa, viés de busca do node2vec
π	pi, probabilidade de transição do node2vec

SUMÁRIO

1 INTRODUÇÃO

Esta dissertação apresenta uma proposta de filtragem de arestas de grafos de palavras. Através da atribuição de pesos utilizando diversas medidas de associatividade de palavras e depois ordenando esses pesos, com base em limiares de corte, removemos dos grafos de palavras as arestas com os menores pesos. Esta proposta foi aplicada ao problema de classificação de texto.

Neste capítulo apresentamos o problema de classificação de texto, como é feita a representação de texto utilizando grafos, a motivação para o desenvolvimento desta pesquisa, seus objetivos geral e específicos, além da organização do restante deste documento.

1.1 PROBLEMA DE CLASSIFICAÇÃO DE TEXTO

Classificação de texto é um dos problemas amplamente explorados em Processamento de Linguagem Natural (NLP). NLP é uma área de pesquisa onde existe uma gama de técnicas computacionais para análise e representação automáticas da linguagem humana (Cambria e White, 2014). O problema de classificação de texto visa atribuir automaticamente a um dado texto, um rótulo de uma classe dentre um conjunto de classes predefinidas (Russell e Norvig, 2016). Esse problema pode ser aplicado em diversas tarefas como filtragem de notícias, organização de documentos e filtragem de *spam*.

De forma geral, o problema de classificação de texto é composto pelos seguintes passos: pré-processamento, extração de características, classificação e avaliação. Em particular, o passo de extração de características transforma os dados em uma representação que algoritmos podem processar. Dentre as formas de representação de texto, como o *bag-of-words* e *n-grams*, uma das representações mais efetivas e que tem ganhado atenção atualmente é baseada em grafos, que consiste no foco deste trabalho.

1.2 REPRESENTAÇÃO DE TEXTOS COMO GRAFOS

Grafos são um tipo de estrutura de dados composta por vértices que são conectados por arestas. As arestas representam as relações entre os vértices. Utilizando grafos é possível obter a representação dos relacionamentos entre os vértices vizinhos bem como o relacionamento de toda a estrutura globalmente.

Muitas aplicações do mundo real podem ser modeladas como grafos. Por exemplo: redes sociais são grafos onde cada usuário é um vértice e suas interações com os outros usuários são as arestas.

Em textos, grafos são capazes de capturar informações importantes, como coocorrência de palavras e relações entre palavras, informações que não são capturadas por modelos mais simples como o *bag-of-words* (Shanavas et al., 2019). Uma forma de modelar textos como grafos é conhecida como grafos de palavras, e essa é a modelagem utilizada neste trabalho. No modelo de grafos de palavras, cada texto da base de dados é representado por um grafo, os vértices do grafo representam as palavras de um texto e as arestas são construídas através de uma janela de coocorrência que passa pelo texto. As palavras que ocorrem dentro da janela têm arestas entre si.

1.3 MOTIVAÇÃO

Como anteriormente mencionado, grafos capturam relações. Utilizando grafos para representar textos, capturam-se as relações entre as palavras. Uma representação de texto em forma de grafo torna possível utilizar algoritmos de aprendizagem de representação de grafos para extrair as características do texto representado no grafo. Tais algoritmos, mapeiam cada vértice do grafo para um vetor de características em que as relações geométricas no espaço de *embedding* refletem a estrutura do grafo original (Hamilton et al., 2017).

Nos últimos anos, pesquisas sobre aprendizagem de representação de grafos têm ganhado cada vez mais atenção, uma vez que a maioria dos dados do mundo real pode ser representada por grafos (Chen et al., 2020). Dentre os trabalhos recentes utilizando grafos para representar textos estão: Bijari et al. (2020), Yao et al. (2019), Huang et al. (2019) e Yu et al. (2018).

Motivado pelo crescimento das pesquisas nessa área, este trabalho foca na construção dos grafos de palavras, que são a entrada para algoritmos de aprendizagem de representação de grafos. Assim, propomos uma representação mais concisa do grafo de palavras, obtida reduzindo a quantidade de arestas do grafo. Como já mencionado previamente, o grafo de palavras é construído colocando arestas entre palavras que coocorrem dentro da janela de coocorrência. Entretanto, conforme mostramos neste trabalho, nem todas as arestas entre as palavras que coocorrem precisam necessariamente estar no grafo para que se obtenha uma boa representação. Utilizando pesos e limiares de corte, é possível remover as arestas com os menores pesos, o que significa remover as relações mais fracas do grafo, obtendo uma representação mais concisa sem perda significativa no acerto da classificação.

1.4 OBJETIVOS

Neste trabalho, propomos uma abordagem para remover arestas de grafos de textos sem causar perda significativa na representação. Para isso, utilizamos medidas de associatividade já existentes para calcular os pesos das arestas dos grafos e identificamos as arestas que podem ser removidas para obter uma representação mais concisa. Utilizamos o corte baseado em limiares, tornando possível avaliar o impacto de diferentes limiares de corte na representação.

Dado o grafo de cada texto, para extrair as características de cada grafo, utilizamos o node2vec, um algoritmo de aprendizagem de representação de grafos. Para a classificação das representações dos grafos de textos, utilizamos um modelo de rede neural convolucional para texto, adaptado de Kim (2014).

Este trabalho tem o objetivo de, aplicado ao problema de classificação de texto, filtrar arestas do grafo de palavras de forma a obter uma representação mais concisa sem causar perda significativa no acerto da classificação.

Para alcançar o objetivo geral, temos os seguintes objetivos específicos:

- A partir do modelo de grafos de palavras, incorporar pesos aos grafos utilizando diversas medidas de associatividade de palavras.
- Ordenar os pesos e definir limiares de filtragem de arestas.
- Utilizar o algoritmo de aprendizagem de representação node2vec para extrair as características dos grafos.
- Realizar experimentos de forma a validar a proposta e avaliar na classificação das representações em quais casos houve perda, em quais houve ganho ou se obteve-se a estabilidade esperada.

- Medir o tempo de execução e o uso de memória durante a execução do node2vec de forma a avaliar os casos nos quais a filtragem de arestas impacta positivamente nesses dois aspectos.

1.5 ORGANIZAÇÃO DO DOCUMENTO

Este documento está organizado da seguinte maneira:

- **Capítulo 2 - Fundamentação teórica:** descreve os principais conceitos necessários para compreensão deste trabalho, incluindo: Processamento de Linguagem Natural, problema de classificação de texto, formas de modelar texto para que um algoritmo de aprendizagem de máquina possa interpretá-lo, como modelar texto na forma de grafos, conceitos relacionados à aprendizagem profunda, aprendizado de representação de grafos e medidas de associatividade de palavras baseadas em coocorrência.

- **Capítulo 3 - Trabalhos relacionados:** aborda os trabalhos relacionados à abordagem proposta.Tais trabalhos se diferenciam principalmente na forma como são extraídas as características dos grafos de textos e na forma como essas características são classificadas, podendo assim ser agrupados em três categorias. A primeira categoria aborda pesquisas que não utilizam aprendizagem profunda e apresentam diferentes métodos para extrair características de textos modelados como grafos e classificá-los. A segunda categoria apresenta pesquisas que, apesar de distinguirem-se no método de representação de cada vértice do grafo, utilizam redes neurais convolucionais como classificador. E a terceira categoria contém os trabalhos que utilizam conceitos de convolução de grafos.

- **Capítulo 4 - Proposta:** apresenta a proposta deste trabalho. Primeiramente apresentamos uma visão geral da proposta e, no decorrer do capítulo, explicamos em detalhes como é feita a modelagem dos textos utilizando grafos, como os pesos das arestas são calculados e nossa abordagem para remoção de arestas. Depois, apresentamos a forma pela qual é feita a aprendizagem de representação dos grafos e a etapa de classificação de grafos de textos. Por fim, descrevemos o que é avaliado na proposta e as comparações realizadas.

- **Capítulo 5 - Experimentos e Resultados:** apresenta os experimentos realizados, os resultados obtidos e suas análises. Neste capítulo apresentamos os componentes da implementação proposta que foi desenvolvida, explicamos a metodologia com que os experimentos foram desenvolvidos e depois apresentamos todos os resultados e as análises sobre os mesmos.

- **Capítulo 6 - Conclusão:** apresenta as conclusões obtidas com o desenvolvimento deste trabalho e os trabalhos futuros identificados.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os conceitos principais que são necessários para uma melhor compreensão deste trabalho. Na Seção 2.1, apresentamos o campo de pesquisa de Processamento de Linguagem Natural. Na Seção 2.1.1, definimos o problema de classificação de texto. A Seção 2.2 apresenta formas tradicionais de modelar texto para que um algoritmo de aprendizagem de máquina possa interpretá-lo. Na Seção 2.3, apresentamos como modelar texto na forma de grafos. Na Seção 2.4, abordamos conceitos relacionados à aprendizagem profunda e na Seção 2.5 apresentamos o conceito de aprendizado de representação de grafos e o algoritmo utilizado neste trabalho para executar essa tarefa. A Seção 2.6 aborda medidas de associatividade de palavras baseadas em coocorrência. Finalmente, na Seção 2.7 apresentamos a validação cruzada estratificada.

2.1 PROCESSAMENTO DE LINGUAGEM NATURAL

Computadores não conseguem entender linguagem natural da mesma forma que seres humanos. Atualmente, com a quantidade de dados sempre em crescimento, os computadores precisam realizar tarefas que os humanos não conseguem fazer em tempo aceitável. Este é o objeto de estudo da área de Processamento de Linguagem Natural (do inglês, *Natural Languague Processing*, NLP).

Chowdhury (2003) define NLP como a área de pesquisa e aplicação que explora como computadores podem ser utilizados para entender e manipular texto ou fala em linguagem natural.

Um problema clássico e fundamental em NLP é o problema de classificação de texto (Yao et al., 2019), no qual, dados um texto de algum tipo e um conjunto pré-definido de classes, a tarefa é definir a qual classe o texto pertence (Russell e Norvig, 2016).

2.1.1 Problema de Classificação de Texto

O problema de classificação de texto foi definido formalmente por Ikonomakis et al. (2005) como:

"Se d_i é um documento pertencente a um conjunto de documentos D e $\{c_1, c_2, \ldots, c_n\}$ é o conjunto de todos os rótulos, a classificação de texto atribui um rótulo c_j para o documento d_i."

Como o rótulo correto de cada texto é conhecido, o problema de classificação de texto compõe um problema de aprendizado supervisionado. Neste tipo de aprendizado, há um *feedback* que fornece a resposta correta para os exemplos (Russell e Norvig, 2016).

A pesquisa em classificação de texto vai desde a extração de boas características até a escolha dos melhores classificadores de aprendizagem de máquina (Zhang et al., 2015). Por isso, esse problema pode ser dividido em alguns passos principais, como: extração de características ou representação, classificação e avaliação. Outro passo auxiliar mas muito importante é o pré-processamento.

Organização de documentos, filtragem de notícias, mineração de opinião, classificação de email e filtragem de spam são alguns exemplos em que a classificação de texto pode ser aplicada.

2.2 EXTRAÇÃO DE REPRESENTAÇÃO DE TEXTO

Textos não podem ser diretamente interpretados por um classificador (Sebastiani, 2002), por isso é necessário representá-los de uma forma que um algoritmo de aprendizado de máquina consiga interpretar. Para isso é comum extrair do texto características que o representam. Nesta seção, abordaremos a etapa de pré-processamento de textos e alguns métodos tradicionais de extração de características.

2.2.1 Pré-processamento

Um dos componentes chave em um *framework* típico de classificação de texto é a etapa de pré-processamento (Uysal e Gunal, 2014). Essa etapa aplica transformações nos textos para que possam ser adequadamente representados e processados por um algoritmo. Alguns componentes do pré-processamento são:

Conversão para letras minúsculas (*Lowercase conversion*): é o processo de converter todas as letras maiúsculas para letras minúsculas.

Tokenização (*Tokenization*): é o procedimento de dividir um texto ou uma frase em palavras ou outras partes significativas, como *tokens* (Uysal e Gunal, 2014).

Por exemplo, na frase: *"Look, this is an example of tokenization."*, o resultado do processo de tokenização é: [*'Look', 'this', 'is', 'an', 'example', 'of', 'tokenization'*].

Remoção de *stop words*: *Stop words* são palavras frequentes que não contêm informações, como preposições e conjunções de pronomes (Srividhya e Anitha, 2010). Exemplos de tais palavras no inglês incluem *'the', 'of', 'and', 'to'* (Gaigole et al., 2013). *Stop words* geralmente são consideradas irrelevantes nos estudos de classificação de texto e, por isso, são removidas (Uysal e Gunal, 2014).

Remoção de ruído: esse processo inclui remover pontuação, caracteres especiais e caracteres não alfabéticos.

Stemming: esse processo visa obter o radical das palavras derivadas (Uysal e Gunal, 2014). A hipótese por trás do *stemming* é que palavras com o mesmo radical geralmente descrevem os mesmos conceitos ou conceitos relativamente próximos no texto (Gaigole et al., 2013). O radical das palavras *"flies"* ("voa") e *"dies"* ("morre"), por exemplo, são *"fli"* e *"die"*.

Algumas etapas de pré-processamento, como a remoção de *stop words* e o *stemming* mudam de acordo com o idioma do estudo.

2.2.2 Métodos tradicionais de representação de texto

Uma forma de representação de documentos de texto geralmente utilizada é modelo de espaço vetorial (*Vector Space Model*, VSM). Nele, cada documento é representado por um vetor em um espaço n-dimensional, em que cada dimensão corresponde a uma palavra do vocabulário geral da coleção de documentos (Skianis, 2019). Dentre os métodos tradicionais de representação de texto dentro do modelo VSM estão:

Bag-of-words: é uma técnica de extração de características que cria uma representação baseada na frequência de ocorrência das palavras (Ganegedara, 2018). As palavras são as dimensões do vetor e os valores correspondem à frequência das mesmas (Aggarwal, 2018). Por exemplo nas frases:

- Bob foi para a loja para comprar algumas flores.
- Bob comprou flores para dar para Alice.

O vocabulário é: ["Bob", "foi", "para", "a", "loja", "comprar", "algumas", "flores", "comprou", "dar", "Alice"]

Para cada sentença, no *bag-of-words*, cria-se um vetor de características do tamanho do vocabulário. Este vetor mostra quantas vezes cada palavra do vocabulário apareceu na sentença. Nas frases do exemplo acima esses vetores seriam:

[1, 1, 2, 1, 1, 1, 1, 1, 0, 0, 0]

[1, 0, 2, 0, 0, 0, 0, 1, 1, 1, 1]

Uma limitação crucial do *bag-of-words* é que a ordem das palavras não é preservada (Ganegedara, 2018). Outra desvantagem dessa abordagem é a alta dimensionalidade da representação.

TF-IDF: É a sigla para *Term Frequency-Inverse Document Frequency*, ou frequência do termo - inverso da frequência nos documentos. O TF-IDF mede o quanto uma determinada palavra é relevante em um documento específico. Para isso mede-se a frequência da palavra no documento em comparação à proporção inversa desta palavra sobre todos os documentos (Ramos et al., 2003).

A equação 2.1 apresenta a fórmula para cálculo do TF-IDF, onde:

- $w_{i,j}$ é o peso da palavra i no documento j;
- $tf_{i,j}$ é a frequência da palavra i no documento j;
- N é o número de documentos na base de dados;
- df_i é a frequência da palavra i em todos os documentos da base de dados.

$$w_{i,j} = tf_i * log(\frac{N}{df_i}) \tag{2.1}$$

Apesar de apresentar vantagens com relação ao *bag-of-words*, como considerar a relevância do termo com relação ao documento, o TF-IDF possui a limitação de assumir que os termos são independentes.

2.2.3 Word2vec

O word2vec, proposto por (Mikolov et al., 2013a), é uma abordagem baseada em redes neurais para aprender a representação de palavras. As representações de palavras que o word2vec aprende são conhecidas como *word embeddings*.

Um *word embedding* é uma forma de representar uma palavra por meio de um vetor de dimensão d. Esses vetores, *word embeddings*, capturam tanto informação sintática quanto semântica das palavras (Liu et al., 2015).

Essencialmente, o word2vec aprende representações de palavras observando as palavras ao redor, isto é, o contexto no qual a palavra é usada. O contexto de uma palavra são as suas vizinhas à esquerda e à direita.

Mais especificamente, dada uma palavra (a central), tenta-se prever seu contexto, ou vice-versa, por meio de uma rede neural, o que força a aprender bons *word embeddings* (Ganegedara, 2018).

No word2vec os vetores de palavras com contextos similares são posicionados próximos no espaço vetorial. Além disso, quando bem treinado, o word2vec é capaz de aprender vários tipos de relações entre palavras. Por exemplo, utilizando expressões algébricas em que se substrai uma relação entre os vetores de duas palavras e adiciona-se o resultado ao vetor de outra palavra,

o resultado é outra relação que faz sentido. Como: Paris − França + Itália = Roma (Mikolov et al., 2013a).

Duas arquiteturas são propostas por Mikolov et al. (2013a) para aprender representações distribuídas de palavras baseadas no contexto: o *Continuous Bag-of-Words* (CBOW) e o *skip-gram*. No CBOW, dado como entrada o contexto, a rede neural deve prever a palavra central correspondente a esse contexto. Já no *skip-gram* dada a palavra central, a rede neural tenta prever seu contexto (Ganegedara, 2018). Como o algoritmo node2vec, apresentado em detalhes na seção 2.5.1, utiliza o modelo *skip-gram*, esse modelo será abordado com detalhes na seção 2.2.3.1.

Por considerar o contexto ao construir a representação das palavras, de forma que os vetores de palavras de significados semelhantes são mapeados para pontos próximos no espaço vetorial, o word2vec possui vantagens com relação aos métodos tradicionais que não consideram a posição das palavras no texto.

2.2.3.1 O modelo Skip-Gram

O skip-gram é uma rede neural com uma camada escondida, e, é utilizado para treinar o word2vec.

Dada uma palavra central (entrada), a rede neural do skip-gram é treinada para dar a probabilidade de cada palavra do vocabulário ser sua vizinha. Dessa forma, as probabilidades de saída estão relacionadas à probabilidade de encontrar cada palavra do vocabulário próxima à palavra de entrada (McCormick, 2016).

As amostras (entradas) de treino do skip-gram, são os pares de palavras encontrados nos documentos de treino. A Figura 2.1, apresenta alguns exemplos de treinamento obtidos da frase: *"A rápida raposa marrom salta sobre o cão preguiçoso"*. Nesse exemplo o tamanho da janela de contexto é dois. A palavra destacada em verde é a palavra central, dada como entrada.

Assim, a rede aprenderá as estatísticas do número de vezes que cada par de palavras aparece (McCormick, 2016).

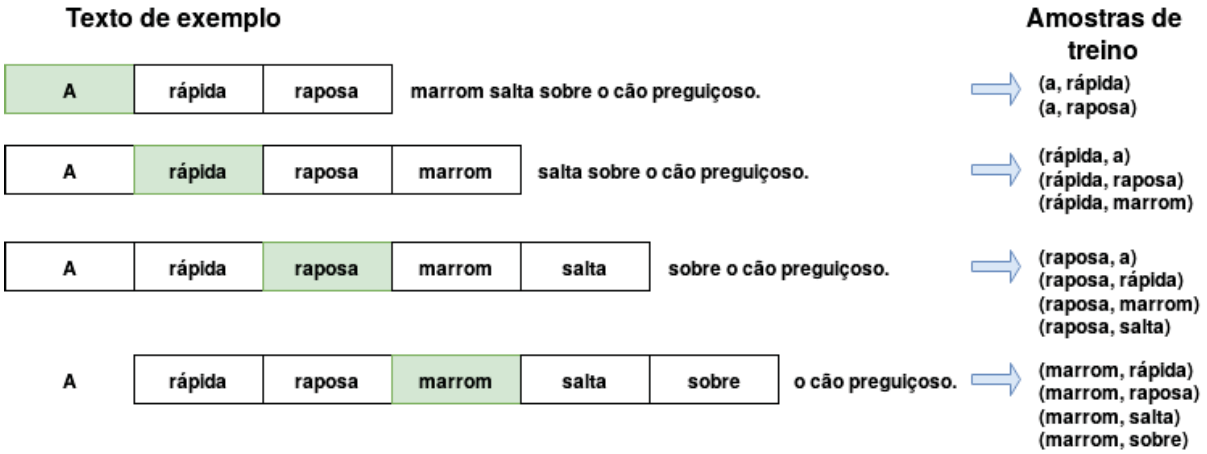


Figura 2.1: Exemplo de geração de amostras de treino para o skip-gram com uma janela de contexto de tamanho 2. A palavra colorida com verde é dada como entrada a rede. A rede é otimizada para prever a palavra na vizinhança com maior probabilidade. Figura adaptada de (McCormick, 2016).

A rede neural do skip-gram, apresentada na Figura 2.2, possui a camada de entrada, uma camada escondida e a camada de saída. A entrada da rede são palavras codificadas na forma de vetores *one-hot* (one-hot encoded). Um vetor codificado em *one-hot* é um vetor do tamanho do dicionário onde todas as posições são preenchidas com o número zero, exceto uma delas que

é preenchida com o número um. A posição que contém o número um é a posição da palavra codificada no dicionário (Godec, 2018).

A camada escondida da rede neural do skip-gram não tem uma função de ativação, a sua saída é um *embedding* da palavra. Já a camada de saída é um classificador *softmax* o qual prevê as palavras da vizinhança (Godec, 2018).

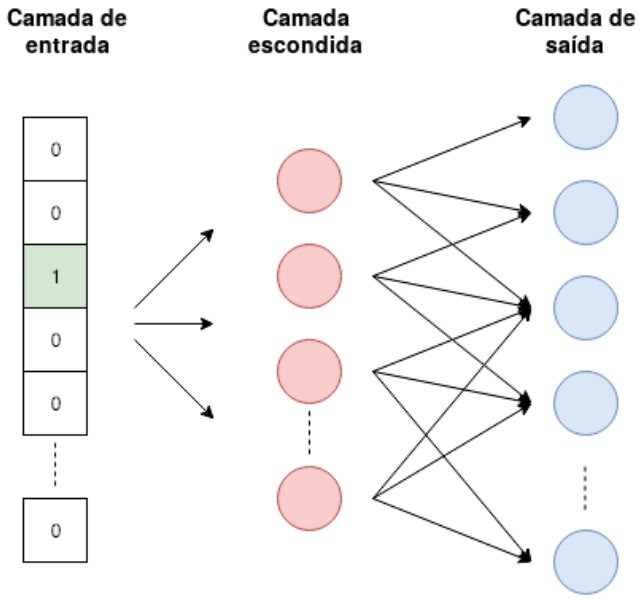


Figura 2.2: Rede neural do skip-gram, contendo a camada de entrada, uma camada escondida e a camada de saída.Figura adaptada de (Godec, 2018).

2.3 REPRESENTAÇÃO BASEADA EM GRAFOS

Nessa seção apresentaremos como textos são representados utilizando grafos. Para isso, primeiramente são definidos alguns conceitos de teoria dos grafos.

2.3.1 Conceitos fundamentais sobre grafos

Um grafo G é uma maneira de codificar relações de pares entre um conjunto de objetos. Ele consiste de uma coleção V de vértices e uma coleção E de arestas, onde cada aresta consiste da junção de dois vértices. Assim, uma aresta $e \in E$ é um subconjunto de dois elementos de $V : e = \{u, v\}$ para algum $u, v \in V$ (Kleinberg e Tardos, 2006). Um grafo (V, E) é um grafo com pesos (*weighted graph*) se existe uma função de peso $w : E \implies \mathbb{R}$ associada a ele (Rosen, 1999).

As arestas em um grafo representam relações simétricas entre os vértices. Muitas vezes, é necessário representar relações assimétricas. Para isso usa-se um grafo direcionado. Um grafo direcionado G' consiste de um conjunto de vértices V e um conjunto de arestas com direção E'. Cada aresta $e' \in E'$ é um par ordenado (u, v); em outras palavras, os papéis de u e v não são permutáveis, chamamos u de cauda e v de cabeça (Kleinberg e Tardos, 2006).

Duas das propriedades do grafo que dizem respeito aos vértices são: (i) quando dois vértices são conectados por uma aresta eles são chamados de vizinhos ou adjacentes, (ii) o grau de um vértice, representado por $d(v)$, é o número de vizinhos desse vértice (Bondy et al., 1976).

Um grafo pode ser representado por uma matriz de adjacência (Vazirani et al., 2018). Dado um grafo $G(V, E)$, $n = |V|$ denota a quantidade de vértices e $m = |E|$ a quantidade de

arestas. A matriz de adjacência de G, denotada por $A = (a_{ij})$ é uma matriz de ordem $|V|$x$|V|$, tal que (Cormen et al., 2009):

$$a_{ij} = \begin{cases} 1, & \text{se } (i, j) \in E; \\ 0, & \text{caso contrário;} \end{cases}$$

Um subgrafo de um grafo $G(V, E)$ é um grafo $S(V', E')$, tal que, V' está contido em V e E' está contido em E e as pontas de cada aresta em E' estão também em V' (McHugh, 1990).

2.3.2 Representação de texto

Textos podem ser modelados como grafos de diversas maneiras, os vértices podem representar parágrafos, frases, palavras, sílabas, entre outros. Já as arestas representam alguma relação entre os vértices (Blanco e Lioma, 2012).

Um documento modelado como um grafo de palavras corresponde a um grafo onde os vértices representam termos únicos (isto é, termos que não se repetem) do documento, e as arestas representam as coocorrências dos termos dentro de uma janela de coocorrência deslizante de tamanho fixo (Rousseau e Vazirgiannis, 2015).

Existem também variações de representação do grafo de palavras. As arestas, por exemplo, podem ser direcionadas ou conter pesos. No escopo deste trabalho serão abordados grafos de palavras com e sem peso nas arestas, contudo, são considerados somente grafos não direcionados.

Mais formalmente, sendo D o conjunto de todos os documentos da base de dados, para um documento $d \in D$, no grafo de palavras $G_d(V_d, E_d)$, os vértices correspondem aos termos t do documento d e as arestas capturam relações de coocorrência entre os termos dentro de uma janela deslizante de tamanho w (Skianis et al., 2018). Essa janela deslizante que calcula coocorrências é também chamada de janela de coocorrência.

A Figura 2.3 apresenta um exemplo de geração do grafo de palavras para a frase: *"A rápida raposa marrom salta sobre o cão preguiçoso"*. Nesse exemplo, são mostrados os primeiros 4 passos de construção do grafo, cada passo mostra um movimento da janela de coocorrência que nesse exemplo tem tamanho 4. A janela de coocorrência está representada pelas palavras dentro dos retângulos de cor verde. Em cada movimento da janela são adicionados no grafo os vértices que representam as palavras dentro da janela e as arestas entre todas as palavras que estão dentro da janela, ou seja, as palavras que coocorrem dentro dessa janela de coocorrência.

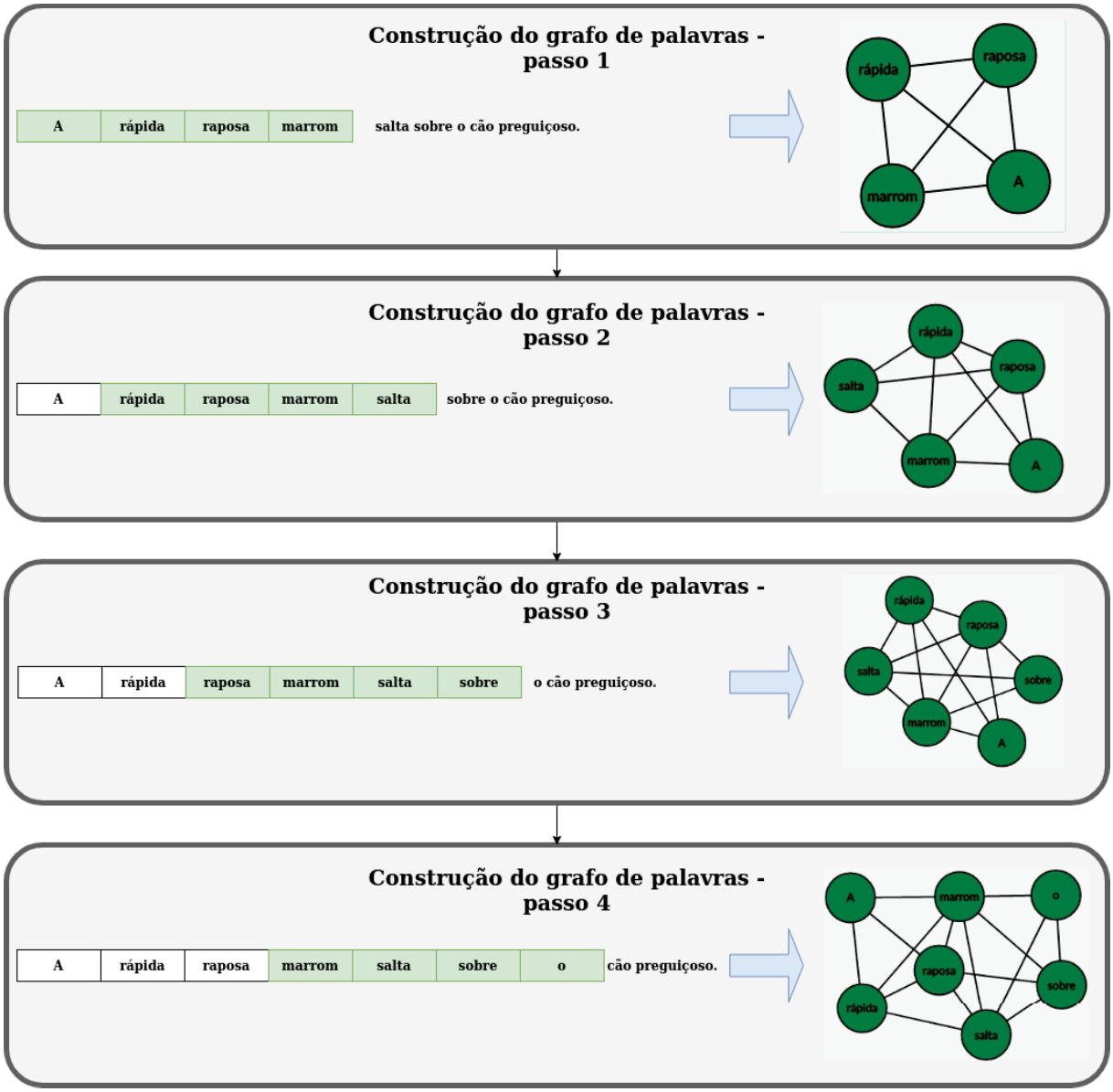


Figura 2.3: Exemplo dos passos inicias de geração de um grafo de palavras a partir de uma frase. A janela de coocorrência utilizada possui tamanho 4, e está representada pelas palavras dentro dos retângulos de cor verde. A cada movimento da janela de coocorrência são adicionados no grafo os vértices que representam as palavras dentro da janela bem como as arestas entre todas as palavras que estão dentro da janela.

2.4 APRENDIZAGEM PROFUNDA

Métodos de aprendizagem de representação são o conjunto de métodos que permitem que uma máquina seja alimentada com dados brutos e descubra automaticamente as representações necessárias para a detecção ou classificação. Aprendizagem profunda, do inglês *deep learning*, são métodos de aprendizagem de representação, com vários níveis de representação, obtidos através da composição de módulos simples, mas não lineares, que transformam a representação em um nível, começando com a entrada bruta, em uma representação em um nível mais alto e um pouco mais abstrata (LeCun et al., 2015). Um modelo profundo é essencialmente uma rede neural artificial que tem uma camada de entrada, muitas camadas escondidas interconectadas no meio e, finalmente, uma camada de saída, um classificador por exemplo (Ganegedara, 2018).

Redes neurais convolucionais (CNN), *Convolutional Neural Networks* em Inglês, são um tipo de rede neural profunda que são amplamente utilizadas pela comunidade de visão computacional. Essas redes são projetadas para processar dados na forma de várias matrizes. Tais dados podem ser por exemplo uma matriz de uma dimensão para sinais e sequências, duas dimensões para imagens e três dimensões para vídeos (LeCun et al., 2015).

A arquitetura de uma CNN é formada tipicamente por uma série de etapas. As primeiras etapas são compostas por dois tipos de camadas: convolução e *poooling*.

Na camada de convolução um filtro deslizante composto de pesos passa na matriz de entrada para produzir a saída dessa camada. A cada posição do filtro sobre a matriz de entrada, os valores da entrada são multiplicados pelas respectivas posições do filtro e somados, produzindo a saída para aquela posição. Após passar o filtro por toda a matriz, obtém-se o mapa de características (*feature map*). O resultado da convolução passa por uma função de ativação, como a ReLU (*Rectified Linear Unit*) (LeCun et al., 2015).

A camada de *pooling* tem o papel de mesclar características parecidas contidas nos mapas de características. Uma unidade de *pooling* pode, por exemplo, calcular o valor máximo de um conjunto de posições de um mapa de características (LeCun et al., 2015).

As saídas das camadas de convolução e *pooling* são geralmente ligadas à entrada de camadas totalmente conectadas (*fully connected layers*). Camadas totalmente conectadas são um conjunto de neurônios totalmente conectados, da entrada à saída. Tais neurônios, são capazes de aprender informações globais combinando características aprendidas pelas camadas de convolução que precedem as camadas totalmente conectadas (Ganegedara, 2018).

2.4.1 Redes neurais convolucionais para texto

Utilizar uma CNN para classificar texto é diferente de a utilizar para classificar imagens, já que as operações de convolução e *pooling* devem ser feitas em uma dimensão (Ganegedara, 2018). Kim (2014) propôs um modelo de CNN específico para texto, onde a convolução é feita nos vetores (*embeddings*) das palavras de uma frase.

O modelo de arquitetura proposto por Kim (2014) é uma variação do proposto anteriormente por Collobert et al. (2011). No modelo de Kim (2014), cada palavra de uma frase é representada por um vetor $x_i \in \mathbb{R}^k$, onde x_i é a i-ésima palavra da frase. E, uma frase de n palavras, que pode ter *padding* quando o tamanho da entrada é menor que n, é representada pela concatenação de todas as suas palavras em sequência (Kim, 2014).

A operação de convolução por um filtro $w \in \mathbb{R}^{hk}$ é aplicada a uma janela de h palavras, para produzir uma nova característica. Essa operação de convolução é aplicada a toda janela de palavras possível em uma sentença para produzir um mapa de características (Kim, 2014).

O *pooling* utilizado é o *max-over-time pooling* (Collobert et al., 2011), que é aplicado sobre cada mapa de características e considera o valor máximo. O propósito desse tipo de *pooling* é realçar a característica mais importante, de maior valor, para cada mapa de características (Kim, 2014).

O modelo de Kim utiliza vários filtros, com tamanhos de janela variados, para obter várias características. Essas características formam a penúltima camada da rede e são passadas para uma camada totalmente conectada, seguida de uma *softmax*, cuja saída é a distribuição de probabilidade sobre os rótulos de cada texto (Kim, 2014).

Este modelo possui duas formas de representação dos vetores das palavras: uma em que os vetores das palavras são inicializados aleatoriamente, e outra em que os vetores das palavras são dados por um word2vec (Mikolov et al., 2013b) treinado previamente.

2.5 APRENDIZAGEM DE REPRESENTAÇÃO DE GRAFOS

O problema central de aprendizagem de máquina em grafos é encontrar uma maneira de incorporar informações sobre a estrutura do grafo em um modelo de aprendizagem de máquina (Hamilton et al., 2017). Pesquisas sobre aprendizagem de representação de grafos tem ganhado cada vez mais atenção nos últimos anos, uma vez que a maioria dos dados do mundo real pode ser representada por grafos (Chen et al., 2020).

Uma abordagem popular para tal tarefa é utilizar *kernels* de grafos, funções que medem a semelhança entre grafos, conectados em um algoritmo que suporte *kernels* como uma máquina de vetores suporte (SVM) (Kriege et al., 2019). Outra maneira, é extrair características escolhidas à mão, o que tem como desvantagens o tempo para para descobrir e projetar tais características e o fato de sua aplicação ser restrita a domínios específicos (Hamilton et al., 2017).

Várias técnicas para *embedding* de grafos (*graph embedding*) têm sido desenvolvidas para converter o grafo em um vetor de alta dimensionalidade, preservando propriedades intrínsecas ao mesmo. Esse processo é também conhecido como aprendizagem de representação de grafos (Chen et al., 2020).

A ideia por trás dos métodos de aprendizado de representação em grafos é aprender um mapeamento que incorpora os vértices, ou todo o grafo, como pontos em um espaço vetorial de baixa dimensão $\mathbb{R}^d$. O objetivo é otimizar esse mapeamento para que as relações geométricas no espaço de *embedding* reflitam a estrutura do grafo original (Hamilton et al., 2017).

Dentre os métodos existentes de aprendizado de representação de grafos existem os métodos de *node embedding*. Algoritmos de incorporação de vértice, em inglês *node embedding*, são aqueles que codificam os vértices do grafo em vetores de baixa dimensionalidade, que sumarizam a posição do vértice no grafo e a estrutura de sua vizinhança local (Hamilton et al., 2017). Em outras palavras, algoritmos de *node embedding* representam cada vértice do grafo como um vetor, onde, os vértices que estão próximos no grafo são incorporados próximos na representação em vetores (Chen et al., 2020).

O Deep Walk (Perozzi et al., 2014), o LINE (Tang et al., 2015) e o node2vec (Grover e Leskovec, 2016) são alguns exemplos de algoritmos de *node embedding* encontrados na literatura. A seguir, detalhamos o funcionamento do node2vec, que é utilizado no desenvolvimento deste trabalho.

2.5.1 Node2vec

O node2vec é um algoritmo semi-supervisionado para extrair características de grafos, que podem ser não direcionados ou direcionados, com ou sem peso. Este algoritmo funciona por meio de passeios aleatórios partindo de cada vértice do grafo (Grover e Leskovec, 2016).

Métodos de passeios aleatórios (*random walks*) geram várias amostras do grafo, através de um grande número de passeios. Esses passeios começam aleatoriamente dos vértices do grafo e indicam o contexto dos vértices conectados. A aleatoriedade dos passeios dá a habilidade de explorar o grafo e capturar as informações estruturais globais e locais ao andar pelos vértices vizinhos (Chen et al., 2020).

Dentre as características principais do node2vec estão: uma noção flexível de vizinhanças de um vértice e um passeio aleatório tendencioso, já que esse passeio é guiado por alguns parâmetros, no qual diversas vizinhanças do vértice são exploradas (Grover e Leskovec, 2016).

A Figura 2.4 ilustra os passos principais do node2vec, onde primeiramente são geradas amostras de passeios aleatórios sobre o grafo de entrada, essas amostras são utilizadas no treinamento da rede neural skip-gram para geração de um embedding para cada vértice do grafo.

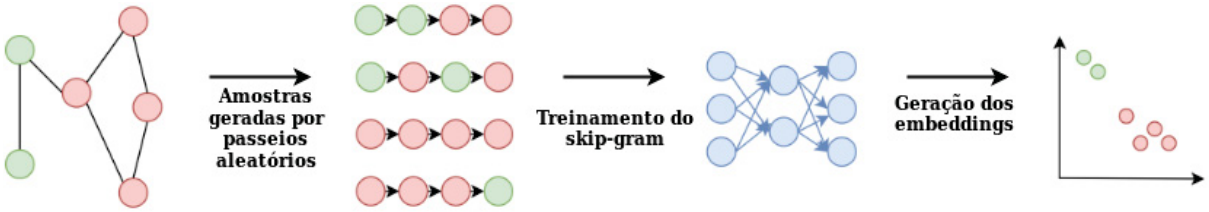


Figura 2.4: Ilustração dos passos principais do algoritmo node2vec. Primeiramente são geradas amostras de passeios aleatórios sobre o grafo de entrada, essas amostras são utilizadas no treinamento da rede neural skip-gram para geração de um embedding para cada vértice do grafo. Figura adaptada de (Godec, 2018).

No node2vec, dado o grafo $G(V, E)$ do qual se deseja extrair a representação, uma função $f : V \mapsto \mathbb{R}^d$ mapeia os vértices para um vetor de características de dimensão d. De forma equivalente, f é uma matriz de parâmetros de tamanho $|V|$xd. Para cada vértice do grafo $u \in V$, $N_s(u) \subset V$ é a vizinhança desse vértice gerada através de uma estratégia de amostragem S (Grover e Leskovec, 2016).

O node2vec propõe uma extensão do modelo skip-gram, proposto no contexto de NLP, para funcionar em grafos. Por isso é necessário gerar a vizinhança de cada vértice. Para isso, dado um vértice fonte u, são propostas diversas amostras de vizinhança desse vértice, onde, dependendo da estratégia de amostragem, a vizinhança do vértice u, $N_s(u)$, não está restrita somente aos vizinhos imediatos de u (Grover e Leskovec, 2016).

A vizinhança de um vértice u é gerada então através de passeios aleatórios de um tamanho fixo l, sendo a quantidade de passeios aleatórios um parâmetro do algoritmo. Este passeio é guiado por dois parâmetros p e q, que permitem à busca intercalar entre largura e profundidade, ou seja, se a vizinhança explorada serão os vértices vizinhos imediatos ou vértices mais profundos. O parâmetro p é também chamado de parâmetro de retorno, e o parâmetro q de parâmetro de entrada e saída (Grover e Leskovec, 2016).

O parâmetro de retorno (p) controla a probabilidade de revisitar um vértice no caminho. Se este parâmetro possuir um valor alto, a probabilidade de incluir na amostra um vértice já visitado é baixa. Em contrapartida, se esse parâmetro tiver um valor baixo, o passeio tem maior probabilidade de revisitar vértices, mantendo o passeio próximo ao vértice inicial (Grover e Leskovec, 2016).

O parâmetro de entrada e saída (q) permite à busca diferenciar entre vértices internos e externos. Se $q > 1$, o passeio aleatório é predisposto em direção aos vértices próximos ao vértice inicial. Nesse sentido, o comportamento do passeio assemelha-se a uma busca em largura, em que as amostras são compostas de vértices em uma localidade pequena. Se $q < 1$, a busca está inclinada a visitar vértices distantes do vértice inicial. Este comportamento reflete a busca em profundidade que encoraja a exploração externa (Grover e Leskovec, 2016).

Considerando que um passeio aleatório acabou de passar pela aresta (t, v) e agora está no vértice v, para decidir qual o próximo vértice a ser visitado são avaliadas as probabilidades de transição π_{vx} a partir de v. A probabilidade de transição para π_{vx} é dada pela Equação 2.2, sendo que α é conhecido como viés da busca (*search bias*) (Grover e Leskovec, 2016).

$$\pi_{vx} = \alpha_{pq}(t, x) \cdot w_{vx} \tag{2.2}$$

Onde w_{vx} é o peso da aresta (v, x) e:

$$\alpha_{pq}(t,x) = \begin{cases} \frac{1}{p} & \text{se } d_{t,x} = 0 \\ 1 & \text{se } d_{t,x} = 1 \\ \frac{1}{q} & \text{se } d_{t,x} = 2 \end{cases}$$

e $d_{t,x}$ denota o menor caminho entre os vértices t e x.

Esse processo está ilustrado na Figura 2.5. Como o passeio saiu do vértice t e foi para o vértice v, a probabilidade de voltar de v para t é $\frac{1}{p}$, a probabilidade de ir de t para x_1 é 1, a probabilidade de ir de t para x_2 é $\frac{1}{q}$ e a probabilidade de ir de t para x_3 é $\frac{1}{q}$.

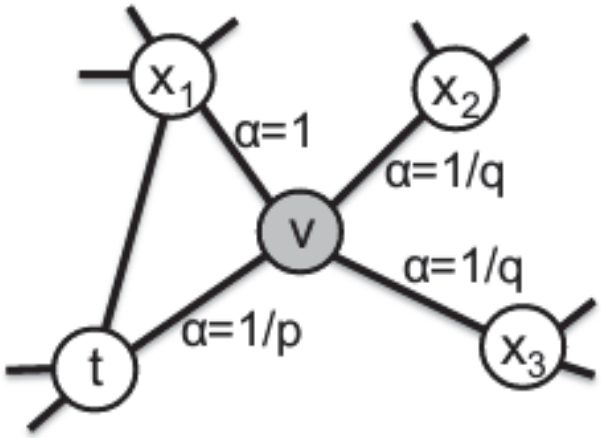


Figura 2.5: Ilustração do procedimento de passeios aleatórios do node2vec. O passeio saiu do vértice t e foi para o vértice v e agora está calculando as probabilidades para decidir qual o próximo vértice a ser visitado no passeio. Os rótulos nas arestas indicam o viés da busca α. Figura extraída de (Grover e Leskovec, 2016)

Utilizando os passeios aleatórios controlados pelos parâmetros p e q, são geradas sequências de vértices. Essas sequências são convertidas para parágrafos de texto que são passados para um algoritmo de aprendizado baseado no word2vec (Abu-El-Haija et al., 2018). Assim sendo, o node2vec é uma extensão do skip-gram para grafos (Grover e Leskovec, 2016).

Dessa forma, o node2vec gera de forma flexível amostras de vizinhança de um vértice, permitindo que essas amostras sirvam para gerar a representação desse vértice com base nas propriedades de sua vizinhança.

Em resumo, o algoritmo do node2vec possui os seguintes parâmetros principais:

- **r** (*num_walks*): número de passeios aleatórios gerados a partir de cada vértice do grafo;
- **l** (*walk_length*): tamanho do passeio, número de vértices que serão visitados;
- **p** : parâmetro de retorno;
- **q**: parâmetro de entrada e saída.

2.6 MEDIDAS DE ASSOCIATIVIDADE DE PALAVRAS

A noção de associatividade de palavras é usada em muitas aplicações de processamento de linguagem e recuperação de informação (Damani e Ghonge, 2013).

Existem três tipos principais de medidas de associatividade de palavras: baseadas em conhecimento (*knowledge based*), baseadas em similaridade de distribuição (*distributional similarity based*) e baseadas em coocorrência léxica (*lexical co-occurrence based*) (Damani e Ghonge, 2013).

Medidas baseadas em similaridade de distribuição caracterizam uma palavra pela distribuição de outras palavras ao redor dela e comparam duas palavras para similaridade de distribuição. Elas também são utilizadas para modelar o significado de uma frase. Medidas baseadas no conhecimento utilizam fontes de conhecimento como enciclopédias e redes semânticas.

Medidas baseadas em coocorrência simplesmente contam frequências de unigramas e bigramas das palavras em um par (Damani e Ghonge, 2013).

Medidas de associatividade de palavras baseadas em coocorrência (*co-occurrence based measures*), estimam a associação entre duas palavras computando alguma função da frequência em unigramas (*unigrams*) e bigramas (*bigrams*) dessas palavras (Damani, 2013). Essas medidas são utilizadas em vários domínios como ecologia, psicologia, medicina e processamento de linguagem (Chaudhari et al., 2011).

Neste trabalho, utilizamos medidas de associatividade de palavras baseadas em coocorrência. Essas medidas são populares porque são computacionalmente eficientes e podem ser aplicadas a qualquer idioma facilmente (Damani e Ghonge, 2013). Nesse contexto, considera-se coocorrência de um par de palavras x e y se essas ocorrem juntas em um intervalo de tamanho fixo. Em outras palavras, x e y coocorrem se aparecem juntas dentro de uma janela de coocorrência deslizante passando pelo texto. Dessa forma, o tamanho da janela de coocorrência é um parâmetro.

Cada uma das medidas e associatividade de palavras baseadas em coocorrência utilizadas neste trabalho estão descritas a seguir.

2.6.1 PMI - *Pointwise Mutual Information*

Proposta por Church e Hanks (1990), a PMI (*Pointwise Mutual Information*) é uma das medidas de coocorrência mais populares (Damani e Ghonge, 2013).

A PMI possui um grande número de aplicações potencialmente importantes, incluindo (Church e Hanks, 1990):

- Restringir o modelo de linguagem para reconhecimento de fala e reconhecimento óptico de caracteres (OCR);
- Fornecer dicas de desambiguação para analisar estruturas sintáticas altamente ambíguas, como compostos de substantivos, conjunções e frases preposicionais;
- Recuperar textos de grandes bancos de dados (por exemplo: periódicos e patentes);
- Aumentar a produtividade de linguistas computacionais na compilação de léxicos de fatos léxico-sintáticos.

Considerando, por exemplo, a aplicação de reconhecimento óptico de caracteres (OCR) e supondo que o OCR atribuiu probabilidade igual de ter reconhecido *"farm"* (fazenda) e *"form"* (forma), onde o contexto é: (1) *"federal ___ credit"* ou (2) *"some ___ of"*. A medida de associatividade PMI pode fazer uso do fato de que *"farm"* é muito mais provável no primeiro contexto e *"form"* é muito mais provável no segundo para resolver a ambiguidade (Church e Hanks, 1990).

Conforme definido por Church e Hanks (1990), a informação mútua (*mutual information*) compara a probabilidade de observar x e y juntos com as probabilidades de observar x e y independentemente. Baseado nisso, o cálculo da PMI é feito conforme a Equação 2.3.

$$log \frac{f(x,y)}{f(x) * f(y)/W} \tag{2.3}$$

Onde:

- W é o número de tokens no corpus;

- $f(x)$, $f(y)$ são as frequências dos unigramas x e y respectivamente;
- $f(x, y)$ é frequência do par com restrição de amplitude (frequência de coocorrência do par de palavras dentro de uma janela de coocorrência de tamanho fixo).

2.6.2 Chi-Square

O teste de chi-square foi investigado por Pearson (1900). Esse teste não assume que as probabilidades sigam uma distribuição normal.

	w_1 = novas	$w_1 \neq$ novas
w_2 = empresas	8 (novas empresas)	4667 (ex: velhas empresas)
$w_2 \neq$ empresas	15820 (ex: novas máquinas)	14287181 (ex: velhas máquinas)

Tabela 2.1: Tabela 2x2 mostrando a dependência das ocorrências das palavras *novas* e *empresas*. Tabela adaptada de Manning e Schutze (1999).

Para explicar a medida chi-square, utilizaremos os valores apresentados na Tabela 2.1, adaptada de (Manning e Schutze, 1999). Essa tabela mostra as dependências de ocorrências das palavras *novas* e *empresas*. Esses valores foram calculados sobre quatro meses do *Newswire* do New York Times: de agosto a novembro de 1990. Este corpus tem cerca de 115 megabytes de texto e cerca de 14 milhões de palavras (Manning e Schutze, 1999).

A Tabela 2.1 mostra a quantidade de vezes, em um cojunto de duas palavras (bigramas), que a palavra *novas* apareceu junto com a palavra *empresas*, quantidade de vezes que a palavra *novas* apareceu sem a palavra *empresas*, quantidade de vezes que a palavra *empresas* aparece sem a palavra *novas* e a quantidade de bigramas em que nem a palavras *novas* e nem a palavra *empresas* aparece.

A medida chi-square soma as diferenças entre os valores observados e esperados em todos os quadrados da Tabela 2.1, escalonados pela magnitude dos valores esperados. Dessa forma, o chi-square é calculado pela Equação 2.4.

$$\sum_{\substack{x' \in \{x, \neg x\} \\ y' \in \{y, \neg y\}}} \frac{(f(x', y') - Ef(x', y'))^2}{Ef(x', y')} \tag{2.4}$$

Onde:

- W é o número de tokens no corpus;
- $f(x)$, $f(y)$ são as frequências dos unigramas x e y respectivamente;
- $f(x, y)$ é frequência do par com restrição de amplitude (frequência de coocorrência do par de palavras dentro de uma janela de coocorrência de tamanho fixo);
- $Ef(x', y')$ é o valor esperado para $f(x', y')$.

As frequências esperadas $Ef(x', y')$ são calculadas a partir das probabilidades marginais, ou seja, dos totais das linhas e colunas convertidas em proporções. Por exemplo, voltando à Tabela 2.1, a frequência esperada para célula $(1; 1)$ (*novas empresas*) é a a probabilidade marginal da palavras *novas* ocorrendo como primeira parte do bigrama multiplicada pela probabilidade marginal da palavra *empresas* ocorrendo como a segunda parte do bigrama (multiplicada pelo número de bigramas no corpus) (Manning e Schutze, 1999).

Esse cálculo para o bigrama (*novas empresas*) é exemplificado na Equação 2.5.

$$\frac{(8+4667)}{N} \times \frac{(8+15820)}{N} \times N \approx 5.2 \tag{2.5}$$

2.6.3 LLR - *Log Likelihood Ratio*

A medida de associatividade LLR (*Log Likelihood Ratio*) foi proposta por Dunning (1993). Conforme aponta o autor da métrica, métodos com base em testes de razão de verossimilhança produzem bons resultados com amostras relativamente pequenas. Esses testes podem ser implementados com eficiência e têm sido usados para a detecção de termos compostos e para a determinação de termos específicos de domínio (Dunning, 1993).

Na pesquisa em que propõe a LLR, Dunning (1993) aponta que, para análise de texto e problemas semelhantes, o uso de razões de probabilidade leva a resultados estatísticos muito melhores. O efeito prático dessa melhoria é que a análise textual estatística pode ser feita de forma eficaz com volumes de texto muito menores do que o necessário para testes convencionais com base em distribuições normais assumidas, e permite que sejam feitas comparações entre o significado das ocorrências de fenômenos raros e comuns (Dunning, 1993).

Além disso, Dunning (1993) afirma que as estatísticas baseadas na suposição de distribuição normal são inválidas na maioria dos casos de análise estatística de texto, a menos que sejam usados textos enormes ou a análise seja restrita apenas às palavras mais comuns (ou seja, aquelas com menor probabilidade de serem de interesse). Usar esses métodos pode superestimar seriamente a importância de eventos relativamente raros. A análise estatística paramétrica baseada na distribuição binomial ou multinomial estende a aplicabilidade dos métodos estatísticos a textos muito menores do que os modelos que usam distribuições normais (Dunning, 1993).

Baseado nisso, o cálculo da medida de associatividade de palavras LLR é dado pela Equação 2.6.

$$\sum_{\substack{x' \in \{x, \neg x\} \\ y' \in \{y, \neg y\}}} p(x', y') log \frac{p(x', y')}{p(x')p(y')} \tag{2.6}$$

Onde:

- W é o número de tokens no corpus;
- $f(x), f(y)$ são as frequências dos unigramas x e y respectivamente;
- $p(x) = f(x)/W$ e $p(y) = f(y)/W$ são as frequências de x e de y divididas pelo número de tokens no corpus;
- $f(x, y)$ é frequência do par com restrição de amplitude (frequência de coocorrência do par de palavras dentro de uma janela de coocorrência de tamanho fixo);
- $p(x, y) = f(x, y)/W$ é a frequência de $f(x, y)$ dividida pelo número de tokens no corpus.

2.7 VALIDAÇÃO CRUZADA ESTRATIFICADA

Dentre os métodos existentes de validação está a validação cruzada estratificada, que é o método utilizado nesse trabalho.

No método de validação cruzada, os dados rotulados são divididos em q partições iguais. Uma das q partições é utilizada para teste e as $(q-1)$ restantes são utilizadas para treinamento. Esse processo é repetido q vezes, para que cada uma das q partições sejam utilizadas para teste. A média da taxa de acerto sobre as q partições de testes diferentes é reportada (Aggarwal, 2018).

Na validação cruzada estratificada, as partições são estratificadas então elas contém aproximadamente as mesmas proporções de rótulos como na base de dados original (Kohavi, 1995).

2.8 CONSIDERAÇÕES FINAIS

Neste capítulo foram apresentados os conceitos principais que serão abordados no decorrer deste trabalho, dentre os quais estão: como modelar um texto na forma de um grafo, uma rede neural convolucional para texto, o algoritmo de aprendizagem de representação de grafos node2vec, as medidas de associatividade de palavras e as métricas de avaliação. No próximo capítulo serão apresentados os trabalhos relacionados.

3 TRABALHOS RELACIONADOS

Neste capítulo serão apresentados os trabalhos relacionados. Estes trabalhos se diferenciam principalmente na forma como são extraídas as características dos grafos de textos e na forma como essas características são classificadas, podendo assim ser agrupados em três categorias. A primeira categoria, abordada na Seção 3.1, contém pesquisas que não utilizam aprendizagem profunda e apresentam diferentes métodos para extrair características de textos modelados como grafos e classificá-los. A segunda categoria, apresentada na Seção 3.2, mostra pesquisas que, apesar de distinguirem-se no método de representação de cada vértice do grafo, utilizam redes neurais convolucionais como classificador. Finalmente, a terceira categoria, descrita na Seção 3.3, contém os trabalhos que utilizam conceitos de convolução de grafos. Na Seção 3.4 apresentamos uma discussão sobre os trabalhos mencionados neste capítulo.

3.1 GRAFOS DE PALAVRAS

Nesta seção apresentamos duas pesquisas (Rousseau et al., 2015; Skianis et al., 2018), que encontram-se no conjunto de trabalhos relacionados em razão de apresentarem abordagens baseadas em grafos de palavras para modelar documentos de texto.

Em Rousseau et al. (2015), cada documento de texto é modelado como um grafo de palavras onde os vértices do grafo representam os termos únicos do documento, e as arestas representam as coocorrências entre termos dentro de uma janela de coocorrência de tamanho fixo. Dessa forma, na abordagem de Rousseau et al. (2015), uma base de dados com n documentos será representada por n grafos.

Como os textos estão na forma de grafos, é possível utilizar técnicas de classificação de grafos, em vez de classificar os textos em si. Classificação de grafos, conforme definido em Rousseau et al. (2015), é a tarefa de predizer de forma automática o rótulo da classe de um grafo.

Para realizar a classificação dos grafos de cada texto, dois métodos convencionais podem set utilizados: subgrafos como características e kernel de grafos. O primeiro método, subgrafos como características, funciona por meio de buscas no conjunto de grafos por subgrafos frequentes, isto é, buscam-se padrões de subgrafos. Esses subgrafos são então utilizados como características para a classificação. Já os kernels de grafos são funções que calculam a similaridade entre grafos.

Na abordagem de Rousseau et al. (2015) foram explorados métodos de subgrafos como características. Para isso, um pacote de software chamado *gSpan* (*Graph-Based Substructure Pattern Minning*) (Yan e Han, 2002), foi utilizado para extrair os subgrafos frequentes. Desse modo, após a mineração de subgrafos que aparecem com frequência nos grafos dos textos, esses subgrafos são utilizados para produzir vetores de características para classificação.

Para avaliar sua abordagem Rousseau et al. (2015) realizaram experimentos em várias bases de dados: duas de classificação multi-classe de documentos, uma de detecção de spam e uma de mineração de opinião. O classificador utilizado pelo autores foi o SVM (*Suporte Vector Machine*, Máquina de Vetores de Suporte). Os resultados se mostraram melhores em taxa de acerto e F1-score com diferença estatística comparados à extração de características utilizando *n-gram* e, como classificadores o KNN (*N-Nearest Neighbors*), o Naive Bayes e o SVM.

Em Skianis et al. (2018), de forma similar a (Rousseau et al., 2015), cada documento é representado por um grafo. Formalmente, sendo D a coleção de todos os documentos da base de dados, cada documento $d \in D$, é representado como um grafo $G_d = (V_d, E_d)$, onde os vértices correspondem aos termos t do documento, e as arestas capturam relações de coocorrência entre

os termos dentro de uma janela de coocorrência de tamanho w. Dessa maneira, são adicionadas arestas conectando os vértices correspondentes aos termos que coocorrem dentro da janela de coocorrência (Skianis et al., 2018).

Além da representação dos documentos por grafos de palavras, Skianis et al. (2018) propõem grafos que representam coleções e grafos que representam rótulos (*labels* das classes).

Um grafo do tipo coleção é formado pela união de todos os grafos de documentos que pertencem a uma coleção. Sendo $\{G_1, G_2, \ldots, G_d\}_{|D|}$ o conjunto de grafos que correspondem a todos os documentos $d \in D$, o grafo do tipo coleção G é o grafo definido pela união dos grafos de documentos $G_1 \cup G_2 \cup \ldots \cup G_d$ dessa coleção. O propósito do grafo do tipo coleção é capturar as dependências globais dos termos da coleção.

Já o grafo de rótulos foi proposto para integrar informações da classe ao grafo. Dessa forma, cada grafo de rótulos representa uma classe. Nesse grafo, as palavras dos documentos que pertencem a uma determinada classe compõem o conjunto dos vértices, e as arestas as coocorrências dessas palavras.

Em sua abordagem, Skianis et al. (2018) mostram como atribuir pesos aos termos do documento, de forma a representar a importância do vértices correspondentes no grafo. Além de pesos nos termos, Skianis et al. (2018) definem pesos para as arestas, que representam a similaridade entre dois termos. Esse peso é dado pela distância cosseno dos *word embeddings* desses termos. Esses *word embeddings* foram extraídos do modelo pré treinado disponibilizado por Mikolov et al. (2013b), porque, segundo os autores os *word embeddings* pré treinados obtiveram melhores resultados que os *word embeddings* treinados pelos autores. Ademais, Skianis et al. (2018) também propõem uma forma de penalizar, no grafo do tipo coleção, o peso de termos que são importantes na coleção de documentos.

A pesquisa de Skianis et al. (2018) foi avaliada em seis bases de dados incluindo classificação multi-classe de documentos, análise de sentimentos e subjetividade. O classificador utilizado foi um SVM, e os resultados obtidos são comparáveis a outros métodos, como a CNN para classificação de texto (Kim, 2014) sem pré treinar os vetores das palavras.

3.2 GRAFOS DE PALAVRAS E APRENDIZAGEM PROFUNDA UTILIZANDO CNN

Nessa seção apresentamos duas pesquisas (Peng et al., 2018; Bijari et al., 2020), que têm em comum o fato de que representarem textos como grafos. Ambos também utilizam *embeddings* que representam os vértices, porém esses *embeddings* diferenciam-se na maneira como são obtidos. Outro ponto de semelhança entre esses trabalhos é que ambos utilizam redes neurais convolucionais.

A pesquisa de Peng et al. (2018), foi desenvolvida aplicada ao problema de classificação hierárquica de texto, no qual os textos devem ser associados aos rótulos da classe com uma hierarquia. Nessa pesquisa os autores desenvolveram uma CNN profunda para grafos, aplicando operações de convolução no grafo de palavras.

Como nos trabalhos anteriormente mencionados neste capítulo, em Peng et al. (2018), cada documento é representado por um grafo, construído capturando coocorrências de palavras em uma janela de coocorrência. Além disso, cada vértice do grafo está associado a um vetor de características, que é o *word embedding* pré-treinado da palavra que o vértice representa. De forma a incorporar informação semântica, Peng et al. (2018) utilizaram o word2vec (Mikolov et al., 2013a,b) treinado na Wikipedia, com o modelo CBOW (*Continuous Bag of Words*) utilizando janela de contexto de tamanho 5 e *embeddings* de dimensão 50.

Nessa abordagem de Peng et al. (2018), após a geração dos grafos, os vértices são ordenados pelos seus graus (número de vizinhos). Então, são selecionados N vértices de maior

grau, sendo N um parâmetro, e uma busca em largura é executada a partir de cada vértice selecionado para expandir um subgrafo. Com isso, são obtidos N subgrafos a partir dos N vértices de maior grau.

O tamanho de cada subgrafo deve ser pelo menos g, que também é um parâmetro do algoritmo. Para subgrafos com mais vértices que g, um filtro é aplicado, e, para subgrafos com menos vértices que g, são adicionados vértices arbitrários desconexos de qualquer outro vértice do grafo.

O próximo passo é então aplicar convolução sobre os *embeddings* dos subgrafos. A primeira camada da rede convolucional tem como entrada um espaço de características de tamanho $N \times g \times D$, onde: N é o número de subgrafos normalizados, g é a quantidade de vértices de cada subgrafo e D é a dimensão dos *word embeddings*. A saída das camadas profundas de convolução (*deep convolutional neural network layers)* alimenta uma camada totalmente conectada, por fim, uma camada *sigmoid* realiza a classificação hierárquica (Peng et al., 2018).

Os experimentos com a *Deep Graph-CNN*, como é chamada essa abordagem de Peng et al. (2018), foram executados em duas bases de dados RCV1 e NYTimes. Os resultados mostraram melhoras para essas duas bases comparados aos resultados de outros modelos do estado da arte.

Outro estudo, conduzido por Bijari et al. (2020), foi proposto com aplicação no problema de análise de sentimentos. A ideia principal desse estudo foi de modelar cada documento como um grafo e então analisar esses grafos utilizando algoritmos de aprendizagem de representação de grafos (Bijari et al., 2020).

A abordagem de Bijari et al. (2020) é composta de três fases principais, nomeadas de: representação do grafo, aprendizado das características e classificação.

A representação do texto no grafo, como nas outras abordagens apresentadas nesse capítulo, é dada pelo modelo de grafos de palavras. Formalmente, a representação proposta por Bijari et al. (2020) é definida como: dado uma frase S e sendo W o conjunto de todas as palavras na frase S. Um grafo $G(W, E)$ é construído de tal forma que qualquer $w_i, w_j \in W$ são conectadas por $e_k \in E$, se $\exists R_l$ tal que $R_l(w_i, w_j) \in R$. A relação R é satisfeita se, por exemplo, as palavras coocorrem em uma janela de coocorrência de tamanho no máximo N(Bijari et al., 2020).

Conforme apontam Bijari et al. (2020), o aprendizado de representação permite que os algoritmos aprendam características diretamente dos dados. Nesse sentido, as características são extraídas explorando técnicas de aprendizagem e fazendo transformação em dados brutos (Bijari et al., 2020).

Por isso, na fase de aprendizado das características, Bijari et al. (2020) utilizaram o algoritmo node2vec para gerar a representação de cada grafo correspondente a cada frase. Desse modo, os vértices dos grafos são representados por *node embeddings* que são dados pelo *node2vec*.

Para a fase de classificação, Bijari et al. (2020) utilizaram uma variação da arquitetura da *ConvNet* (Kim, 2014; Collobert et al., 2011). A operação de *padding* foi aplicada para que todas as sentenças tivessem o mesmo tamanho na entrada da CNN. Para camada de *pooling* foi utilizado o *max-over-time pooling* (Collobert et al., 2011), onde a ideia é capturar e manter as características mais importantes de cada mapa de características. A saída das camadas de convolução alimentam uma camada totalmente conectada e uma camada *softmax* dá a probabilidade de cada classe de sentimento (Bijari et al., 2020).

Os experimentos da abordagem de Bijari et al. (2020) foram executados sobre cinco bases de dados de análise de sentimentos. Os resultados obtidos mostraram-se superiores aos resultados das abordagens comparadas. Além disso, Bijari et al. (2020) executaram experimentos com o intuito de analisar a sensibilidade de parâmetros como grafos direcionados ou não direcionados,

os parâmetros de exploração p e q do node2vec, e o tamanho da janela de coocorrência utilizada para gerar os grafos.

3.3 ABORDAGENS BASEADAS EM CONVOLUÇÃO DE GRAFOS

Nesta seção serão apresentadas três pesquisas (Yu et al., 2018; Yao et al., 2019; Huang et al., 2019). Essas pesquisas têm em comum o fato de trabalharem com redes de convolução propostas para atuarem diretamente nos grafos. Entretanto, todas essas pesquisas diferenciam-se na maneira de construir os grafos a partir dos textos.

A pesquisa de Yu et al. (2018) foi aplicada ao problema de recuperação de informações entre diferentes tipos de dados (*cross-modal information retrieval*). Nesse problema, o objetivo é encontrar dados heterogêneos de vários tipos a partir de uma determinada consulta de um tipo de dado específico, como por exemplo encontrar imagens de pássaros dado um texto que fale sobre pássaros. Nesse trabalho, as duas modalidades abordadas foram textos e imagens, sendo necessário portanto, construir uma representação para os textos.

Em Yu et al. (2018), o processo de construção do grafo tem o objetivo de combinar informação estrutural e semântica. Dado um conjunto de documentos e o vocabulário, primeiramente é construído um conjunto denotado por W= $[w_1, w_2, \ldots, w_N]$, composto pelas palavras mais comuns do vocabulário. Cada palavra de W é representada por um *word embedding* pré treinado. Tendo essas informações é possível construir o que os autores chamam de grafo de k-vizinhos (*k-neighbor graph*).

Um grafo de k-vizinhos, denotado por $G = (V, E)$, é o grafo onde cada vértice $v_i \in V$ é uma palavra, e cada aresta $e_{ij} \in E$ é definida pela similaridade do cosseno do *word2vec* entre as duas palavras:

$$e_{ij} = \begin{cases} 1, & \text{se } w_i \in \mathrm{N}(w_j) \text{ ou } w_j \in \mathrm{N}(w_i) \\ 0, & \text{caso contrário} \end{cases}$$

onde, $N_k(.)$ denota o conjunto de k-vizinhos mais próximos computando a similaridade do cosseno entre os *word embeddings* (Yu et al., 2018).

Dessa forma, na abordagem de Yu et al. (2018), cada grafo representa um documento. Além disso, o *bag-of-words* de cada documento é calculado, permitindo que a frequência de cada palavra w_i seja utilizada como característica do vértice v_i. Uma GCN, sendo a versão proposta por Defferrard et al. (2016), contendo duas camadas de convolução, é utilizada para aprender a representação de cada grafo.

Voltando ao problema de classificação de texto, Yao et al. (2019), diferente de todos os trabalhos anteriormente mencionados, propõem modelar todos os documentos da base de dados em um único grafo. Dessa forma, é possível utilizar a GCN (Kipf e Welling, 2017) como um classificador de vértices.

Yao et al. (2019) constroem a partir de todos os documentos da base de dados um único grafo com dois tipos de vértices: os que representam palavras e os que representam os documentos. Todas as palavras da base de dados são adicionadas como vértices do tipo palavras e também são adicionados um vértice para cada documento da base de dados. As arestas entre vértices de palavras são adicionadas por janelas deslizantes de coocorrência. Um vértice do tipo documento possui arestas para todas as palavras que estão nesse documentos.

Yao et al. (2019) propõem duas formas de atribuir pesos nas arestas do grafo. O peso de uma aresta entre um vértice de documento e um vértice de palavra é dado pelo TF-IDF da palavra no documento, onde a frequência do termo é o número de vezes que a palavra

aparece no documento, e a frequência inversa de documentos é a fração inversa logaritmicamente dimensionada do número de documentos que contêm a palavra (Yao et al., 2019). O peso das arestas entre palavras são calculadas utilizando PMI (*Pointwise Mutual Information*), e, somente são adicionadas ao grafo arestas entre palavras com o PMI positivo.

Formalmente, (Yao et al., 2019) define o peso de uma aresta entre o vértice i e o vértice j como:

$$A_{ij} = \begin{cases} \text{PMI}\,(i,j)\,, & i \text{ e } j \text{ são palavras, PMI}(i,j) > 0 \\ \text{TF-IDF}_{ij}, & i \text{ é um documento, } j \text{ é uma palavra} \\ 1, & i = j \\ 0, & \text{caso contrário (nenhuma das anteriores)} \end{cases}$$

O PMI de um par de palavras i, j é calculado da seguinte maneira:

$$PMI(i,j) = \log\left(\frac{p(i,j)}{p(i)p(j)}\right)$$

$$p(i,j) = \frac{\#W(i,j)}{\#W}$$

$$p(i) = \frac{\#W(i)}{\#W}$$

Onde, $\#W(i)$ é o número de janelas deslizantes de coocorrência em um corpus que contém a palavra i, $\#W(i,j)$ é o número de janelas deslizantes que contêm as palavras i e j, e $\#W$ é o número total de janelas deslizantes (Yao et al., 2019).

Esse grafo com pesos é dado como entrada para uma GCN (Kipf e Welling, 2017) com duas camadas, onde é feita a aprendizagem de representação. No fim da segunda camada de convolução uma camada do tipo *softmax* realiza a classificação dos vértices de documentos.

Conforme apontado pelos autores, uma limitação desse estudo é que o modelo de GCN utilizado é transdutivo, no qual os vértices que representam os documentos da base de teste, sem os rótulos, estão presentes no treinamento da GCN. Assim, a GCN para texto não pode gerar *embeddings* e predizer documentos de teste não vistos durante o treinamento (Yao et al., 2019).

Os experimentos da abordagem proposta por Yao et al. (2019) foram executados primeiramente para verificar se o modelo proposto poderia alcançar resultados satisfatórios na classificação de texto, mesmo com poucos dados rotulados. Posteriormente, foram realizados experimentos para avaliar a capacidade do modelo de aprender *embeddings* para documentos e palavras.

Nos experimentos executados com a GCN para texto, o tamanho dos *embeddings* na primeira camada de convolução foi 20 e o tamanho da janela deslizante que passa pelo texto para contruir o grafo também foi 20. Esses experimentos foram realizados em cinco bases de dados de classificação de texto. Os resultados mostraram que a GCN para texto foi significativamente melhor que outros modelos do estado da arte, existentes até aquele momento, em quatro das cinco bases de dados testadas.

Outro estudo de Huang et al. (2019) propõe um novo modelo baseado em redes neurais de grafos, que diferente de Yao et al. (2019) constrói um grafo para cada texto. Nesse estudo, Huang et al. (2019) propõem utilizar tamanhos pequenos de coocorrência para que características locais sejam extraídas e o número de arestas seja reduzido.

Conforme apontam Huang et al. (2019), os métodos de representação baseados em redes neurais de grafos que constroem um único grafo de todos os documentos apresentam

alguns problemas práticos. O primeiro problema é o alto consumo de memória devido à grande quantidade de arestas. Outro problema é a dificuldade de construir teste online, testar com novos documentos, porque a estrutura e os parâmetros do grafo dependem do vocabulário dos documentos e não podem ser modificados depois do treino.

Nesse trabalho, um texto de l palavras é representado como $T = r_1, ...r_i, ...r_l$, onde r_i denota a representação da i-ésima palavra. r_i é um vetor inicializado por um *word embedding* de dimensão d e pode ser atualizado durante o treinamento.

Para construir o grafo para um dado texto, todas as palavras desse texto são adicionadas como vértices no grafo. As arestas são construídas baseadas em coocorrência (Huang et al., 2019). De forma concreta, o grafo de um texto T é definido como (Huang et al., 2019):

$N = \{r_i | i \in [1, l]\}$

$E = \{e_{i,j} | i \in [i, l], j[i - p, i + p]\}$

onde: N é o conjunto de vértices e E é o conjunto de arestas do grafo e p denota o número de palavras adjacentes conectadas a cada palavra no grafo (Huang et al., 2019).

Diferente das abordagens de Yu et al. (2018) e Yao et al. (2019) que, utilizam convolução espectral, nessa abordagem de Huang et al. (2019), um método não espectral denominado *message passing mechanism* (MPM, mecanismo de passagem de mensagem) (Gilmer et al., 2017) é utilizado para convolução.

No MPM a construção das representações dos vértices é influenciada por seus vizinhos, o que significa que as representações podem trazer informações do contexto. O MPM primeiramente coleta informações sobre os vértices adjacentes e atualiza suas representações baseado nas suas representações originais e na informação coletada.

Após a fase de aprendizado, a classificação das representações agrupadas dos vértices do grafo é feita por uma camada *softmax*. Dessa forma, na proposta de Huang et al. (2019), a representação de um texto não é dependente de todo o corpus, fazendo com que possa ser realizado teste com novos documentos contendo palavras que não estavam presentes no vocabulário de treino.

A dimensão das representações utilizada nos experimentos foi 300 e os vetores foram inicializados de duas maneiras diferentes: ou com valores aleatórios ou com o Glove (Pennington et al., 2014). Os experimentos foram realizados em três bases de dados de classificação de texto e os resultados mostraram-se superiores aos modelos do estado da arte comparados.

3.4 DISCUSSÃO

As pesquisas Rousseau et al. (2015) e Skianis et al. (2018) demonstram a relevância de representações de texto baseadas em grafos. Esses trabalhos mostram como extrair características dos grafos para classificação. Mesmo sem utilizar aprendizagem profunda, esses trabalhos reforçam que a representação de texto utilizando grafos pode alcançar resultados satisfatórios.

Mais próximos da abordagem proposta neste trabalho estão as pesquisas de Peng et al. (2018), Bijari et al. (2020) e Yao et al. (2019).

Peng et al. (2018) mostram uma forma de normalizar os grafos e utilizar a representação de *word embeddings* de cada palavra que os vértices do grafo representam, tornando possível utilizar uma CNN, proposta inicialmente para trabalhar sobre imagens, para aprendizagem e classificação das representações dos vértices dos grafos.

Bijari et al. (2020) utilizam o node2vec para gerar a representação dos grafos e então utilizam uma CNN, específica para texto, para aprender a representação dada pelo node2vec de cada grafo e classificá-los.

Finalmente, Yao et al. (2019) constroem um único grafo contendo vértices para documentos e para palavras e utilizam diferentes métricas para calcular o peso das arestas. Além disso, Yao et al. (2019) mostram que é possível remover do grafo as arestas entre palavras em que o PMI foi negativo. Então, uma GCN consegue gerar a representação de todo o grafo e classificar os vértices que correspondem a documentos.

Porém conforme apontado por Huang et al. (2019), uma representação contendo um único grafo representando toda base de dados apresenta duas desvantagens: é muito custoso em memória devido à grande quantidade de arestas e não é possível obter a representação de textos que não estavam no grafo durante o treinamento.

Na Tabela 3.1 apresentamos um resumo comparativo dos trabalhos relacionados descritos neste capítulo. Essa tabela mapeia algumas características que esses trabalhos apresentam ou não, as quais ajudaram a definir a proposta apresentada por este trabalho.

A primeira coluna da Tabela 3.1 indica a seção deste capítulo na qual o trabalho relacionado foi apresentado. A proposta deste trabalho, que será apresentada em detalhes no próximo capítulo, incorpora detalhes dos trabalhos apresentados em cada seção.

Dos trabalhos apresentados na Seção 3.1, utilizamos o mesmo modelo de representação de texto para grafo, o modelo de grafos de palavras. Dos trabalhos apresentados na Seção 3.2, incorporamos em nosso trabalho o modelo de aprendizagem de representação utilizando o node2vec. E com base em Bijari et al. (2020), utilizamos também o modelo baseado na CNN para texto de Kim (2014).

Finalmente, a Seção 3.3, mostra que há trabalhos recentes utilizando representações baseadas em grafos e que novas redes de convolução que trabalham diretamente no grafo vêm sendo desenvolvidas. A partir do trabalho de Yao et al. (2019), que utiliza um único grafo para representar todos os documentos e utiliza o PMI para calcular os pesos das arestas, decidimos conduzir um estudo avaliando diferentes formas de calcular os pesos das arestas onde cada grafo representa um único documento. E também, propor um método de remoção de arestas que seja baseado em uma porcentagem do total de arestas de cada grafo.

3.5 CONSIDERAÇÕES FINAIS

Neste capítulo apresentamos as pesquisas relacionadas à proposta deste trabalho. Essas pesquisas mostram a relevância da representação de documentos utilizando grafos, tanto quando ainda não existiam algoritmos para aprender a representação de cada vértice do grafo, quanto atualmente em que existem diversos algoritmos de representação de grafos.

As pesquisas aqui apresentadas motivaram o desenvolvimento deste trabalho que foca na utilização de medidas de associatividade de palavras como pesos nos grafos de palavras onde cada documento é representado por um grafo. Além de adicionar mais informações aos grafos e melhor guiar o algoritmo de aprendizagem de representação node2vec, os pesos funcionam como um guia para filtrar do grafo de palavras as arestas que conectam palavras com baixa associatividade. O objetivo dessa filtragem é ter grafos de palavras com menos arestas mas com o mesmo nível de representação dos grafos de palavras com todas as arestas.

O próximo capítulo apresenta em detalhes a abordagem proposta.

Seção	Trabalho	Problema	Modelagem	Pesos	Remove arestas	Método
Seção 3.1	(Rousseau et al., 2015)	Classificação de texto	Um grafo por documento	Não	Não	Sem *deep learning*
	(Skianis et al., 2018)	Classificação de texto	Um grafo por documento	Sim, distância cosseno dos *word embeddings*	Não	Sem *deep learning*
Seção 3.2	(Peng et al., 2018)	Classificação hierárquica de texto	Um grafo por documento	Não	Não	word2vec + CNN
	(Bijari et al., 2020)	Análise de sentimentos	Um grafo por documento	Não	Não	node2vec + CNN
Seção 3.3	(Yu et al., 2018)	Recuperação de informação entre modalidades	Um grafo por documento	Não	Não	GCN + *bag of words*
	(Yao et al., 2019)	Classificação de texto	Todos os documentos em um único grafo	Sim, PMI	Sim (apenas as negativas)	GCN
	(Huang et al., 2019)	Classificação de texto	Um grafo por documento	Sim, PMI	Não	MPM

Tabela 3.1: Resumo comparativo dos trabalhos relacionados apresentados

4 PROPOSTA

Neste capítulo apresentamos a abordagem proposta neste trabalho. Na Seção 4.1, apresentamos uma visão geral da proposta. Na Seção 4.2, explicamos em detalhes como é feita a modelagem dos textos utilizando grafos, como os pesos das arestas são calculados e nossa abordagem para remoção de arestas. Na Seção 4.3, apresentamos a forma pela qual é feita a aprendizagem de representação dos grafos. Na Seção 4.4, apresentamos a etapa de classificação de grafos de textos. Na Seção 4.5, descrevemos o que é avaliado na proposta e as comparações realizadas.

4.1 PROPOSTA

Através da pesquisa de trabalhos relacionados, apresentada no capítulo anterior, identificou-se a possibilidade de utilização de diversas medidas de associatividade de palavras baseadas em coocorrência para atribuir pesos às arestas dos grafos de palavras, que também são construídos utilizando janelas de coocorrência.

Através desses pesos é possível identificar arestas entre palavras com baixa associatividade, que podem ser removidas do grafo de palavras, gerando uma representação mais concisa.

Baseado nisso, este trabalho propõe uma abordagem, aplicada ao problema de classificação de texto, onde os documentos são representados por grafos utilizando o modelo de grafos de palavras com pesos calculados através de diferentes medidas de associatividade de palavras.

A partir disso, propomos avaliar o impacto da remoção de arestas com os menores pesos. Essa remoção é feita sobre uma porcentagem do total de arestas de cada grafo, assim podemos mensurar até qual taxa de remoção de arestas não há perda significativa na representação gerada.

Para avaliar se há ou não perda, utilizamos um algoritmo de aprendizagem de representação de grafos para gerar os *embeddings* de cada grafo de texto. Esses *embeddings* são então classificados e a partir do acerto na classificação avaliamos se houve ou não perda na representação.

A proposta deste trabalho envolve diversos componentes: pré-processamento dos textos, construção dos grafos com pesos, incluindo a ordenação e remoção de arestas, aprendizagem da representação dos grafos, classificação e por fim avaliação. Uma visão geral de nossa proposta, contendo cada um desses componentes é apresentada na Figura 4.1.

Na etapa de pré-processamento, cada texto primeiramente é convertido para *lowercase* e tokenizado. Então, são removidos os ruídos como pontuação e caracteres que não fazem parte do alfabeto. O próximo passo é remover as *stop words* presentes em cada texto. Por fim, os tokens de cada texto passam pelo processo de *stemming*.

Após o pré-processamento, o passo seguinte é a construção do grafo que representa cada texto da base de dados. Para tal, utilizaremos o modelo de grafos de palavras. Utilizamos diferentes medidas de associatividade de palavras para determinar os pesos nas arestas baseados nas janelas coocorrência. Com isso, o grafo incorpora mais informações sobre as palavras conectadas. Esses pesos também são ordenados e uma porcentagem das arestas de menor peso é removida do grafo. Pois, um baixo valor de peso é um indicativo de baixa associatividade entre essas palavras indicando que a remoção da aresta entre essas palavras pode não causar perda significativa na representação. Na Seção 4.2, descrevemos em detalhes o processo de modelagem dos grafos.

Dado o grafo de cada documento, o próximo passo é o aprendizado da representação dos mesmos. Para isso, utilizamos o node2vec. Como a saída de um algoritmo de aprendizagem de representação é um *embedding* (vetor de características) para cada vértice do grafo, a representação de todo o grafo requer uma matriz de *embeddings*. Dado que temos um grafo por documento, consequentemente, teremos uma matriz de *embeddings* por grafo. A Seção 4.3, apresenta mais informações sobre a etapa de aprendizagem da representação.

O passo seguinte, o de classificação, é feito utilizando um modelo de CNN adaptado de Kim (2014), o qual terá como entrada a matriz de *embeddings* de cada texto que será classificado. A principal diferença comparando com a abordagem de Kim (2014) é que, no lugar de ter uma matriz de *embeddings* com o *embedding* de cada palavra do texto, nossa abordagem terá o *embedding* do vértice correspondente a cada palavra do texto. A Seção 4.4 apresenta mais alguns detalhes sobre a fase de classificação.

A etapa final é a de avaliação da classificação, que analisa se houve perda na representação. A partir disso é feita a comparação e análise das diferentes representações. Esse processo está detalhado na Seção 4.5.

4.2 MODELAGEM DOS GRAFOS

Os grafos em nossa abordagem seguem o modelo de grafos de palavras (Rousseau et al., 2015). Cada documento da base de dados é representado por um grafo, os vértices são as palavras do documento e as arestas são dadas pela coocorrência das palavras dentro de um janela deslizante de tamanho fixo, essa janela é também chamada de janela de coocorrência.

Nesse trabalho propomos utilizar diversas medidas de associatividade de palavras baseadas em coocorrência para atribuir pesos às arestas dos grafos de palavras. Através desses pesos identificamos arestas entre palavras com baixa associatividade, e propomos um método para remover essas arestas, gerando uma representação mais concisa.

Assim, iremos comparar a nossa abordagem de grafos de palavras com pesos e remoção de arestas com os grafos de palavras sem pesos contendo todas as arestas.

Para calcular os pesos, como já mencionado anteriormente, utilizaremos medidas de associatividade de palavras baseadas em coocorrência. Como o grafo de palavras gera as arestas utilizando uma janela de coocorrência que passa pelo texto capturando coocorrências das palavras, propomos utilizar medidas de associatividade que são calculadas sobre as coocorrências das palavras para atribuir pesos às arestas.

Um detalhe importante é que utilizamos o mesmo tamanho para a janela de coocorrência que gera as arestas do grafo de palavras e a janela de coocorrência que calcula a medida de associatividade. Assim, por exemplo, se os grafos são gerados com a janela de tamanho 4 a medida de associatividade é calculada com uma janela de tamanho 4 também.

Para uma melhor avaliação, utilizaremos diferentes medidas de associatividade para calcular o peso das arestas, são elas: PMI (*Pointwise Mutual Information*), LLR (*Log Likelihood Ratio*) e *Chi-square*. Baseado nessas medidas, para cada uma delas modelamos dois tipos de grafos com peso : **local** e **global**. Esse dois tipos de grafos com peso, diferenciam-se na forma em que os pesos são calculados. Os **pesos locais** são sempre calculados sobre cada documento separadamente, o que quer dizer que levamos em conta apenas as coocorrências de palavras do documento em questão para calcular a associatividade entre suas palavras. Assim, o mesmo par de palavras pode ter diferentes valores de peso, dependendo do documento. Os **pesos globais**, por sua vez, consideram todo o conjunto de documentos para calcular os pesos, dessa forma, o mesmo par de palavras que aparece em diferentes documentos terá sempre o mesmo valor de peso.

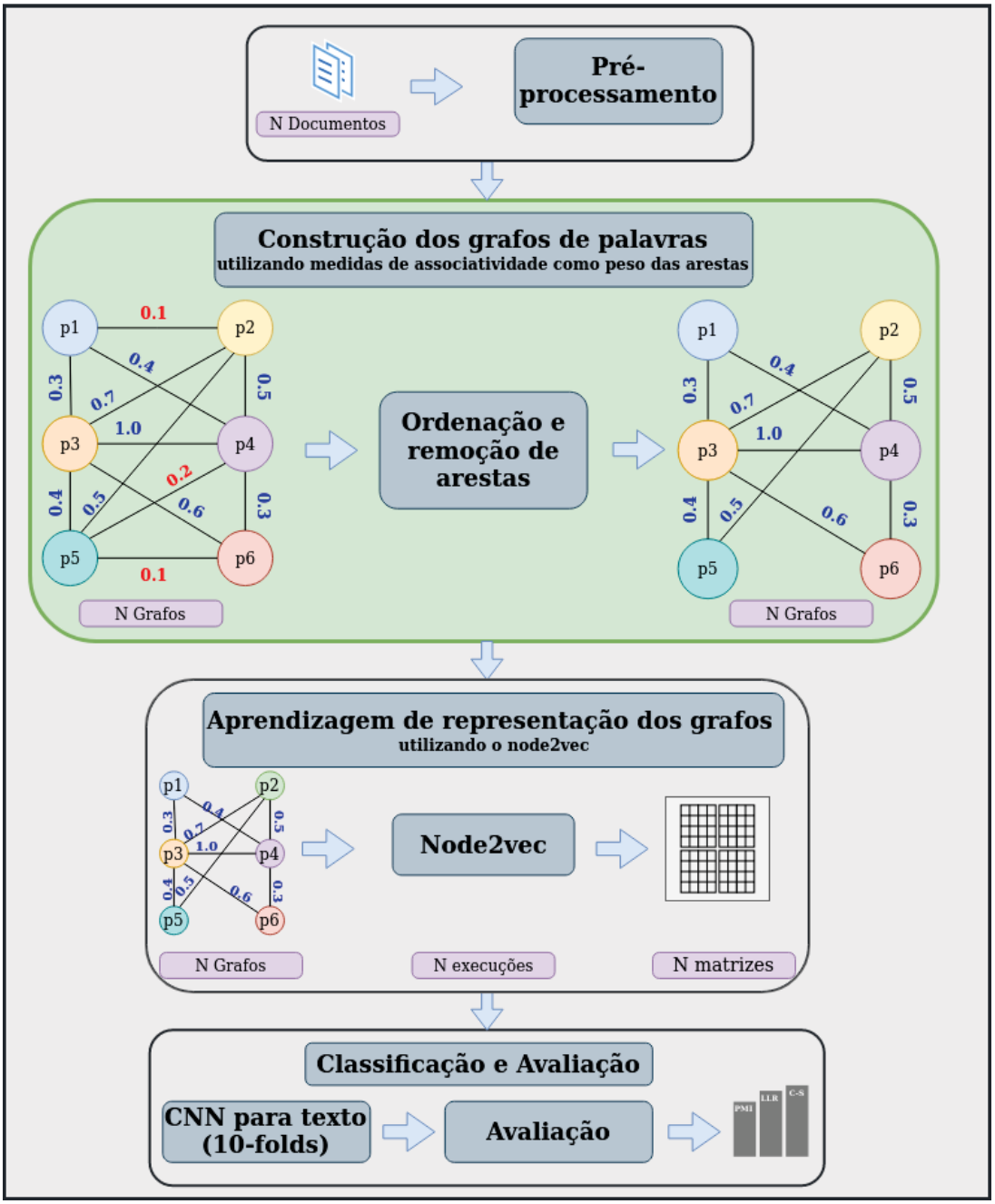


Figura 4.1: Abordagem proposta. Primeiramente, os textos são pré-processados, seguindo para a construção dos grafos com peso que inclui a ordenação das arestas para remoção daquelas com menor peso. O passo seguinte é a aprendizagem da representação dos grafos feita através do node2vec, sucedido pela classificação utilizando uma CNN para texto. Por fim, é realizada a avaliação da classificação.

Desse modo a abordagem incorpora e avalia seis diferentes métodos de atribuir pesos às arestas dos grafos de palavras:

- PMI local;
- PMI global;
- LLR local;
- LLR global;

- Chi-square local;
- Chi-square global.

Tendo os grafos com pesos, podemos então partir para a parte principal da proposta, a filtragem de arestas.

4.2.1 Filtragem de arestas

Após a geração dos grafos com pesos, para cada grafo, as arestas são ordenadas pelo seu peso, do menor para o maior. E removemos uma certa porcentagem p de arestas de cada grafo, começando daquelas com o menores pesos. Essa porcentagem p é um parâmetro da abordagem, e é avaliado considerando-se diferentes valores nos experimentos.

A porcentagem p é sempre um valor inteiro e a remoção das arestas é feita em cada grafo do *dataset* individualmente, então para cada grafo, a partir da quantidade total de suas arestas, são removidas as $p\%$ menores. Considerando por exemplo $p = 5\%$ e uma base de dados contendo 2000 documentos, serão removidas 5% das arestas de cada um dos 2000 grafos. Como os textos na base de dados têm quantidades de palavras diferentes, os grafos gerados a partir cada texto têm consequentemente uma quantidade particular de arestas, dependendo do tamanho do texto. Continuando o exemplo, se existirem dentre os 2000 grafos, grafos com 100 arestas, grafos com 200 arestas e grafos com 500 arestas, de cada um dos grafos com 100 arestas serão removidas as 5 arestas de menor peso, de cada um dos grafos com 200 arestas serão removidas as 10 arestas de menor peso e de cada um dos grafos com 500 arestas serão removidas as 25 arestas de menor peso.

Se no processo de remoção algum vértice ficar desconectado do grafo, isto é, não tiver nenhuma aresta o conectando a qualquer outro vértice, esse vértice é removido do grafo. Isso significa que o grafo que representa um texto pode ter menos palavras que o texto original.

Com esse processo de filtragem de arestas propomos uma representação mais concisa que o grafo de palavras em que se pretende manter a mesma qualidade da representação. Como a porcentagem de arestas a serem removidas é um parâmetro, avaliando diferentes valores, podemos concluir até que porcentagem de arestas é possível remover sem causar perda de qualidade da representação. Assim, queremos mostrar que é possível representar o mesmo texto por um grafo de palavras com menos arestas e que as medidas de associatividade de palavras são boas medidas para ranquear arestas e encontrar aquelas que podem ser removidas.

Grafos com menos arestas levam a um ganho de processamento no algoritmo de aprendizagem de representação pois serão processadas menos arestas. Esse passo de aprendizagem de representação é apresentado detalhadamente a seguir.

4.3 APRENDIZAGEM DE REPRESENTAÇÃO DOS GRAFOS

Para realizar o aprendizado da representação dos grafos, utilizamos o node2vec. Esse algoritmo de aprendizagem de representação de grafos foi escolhido para essa fase de aprendizagem de representação devido a seu funcionamento baseado em passeios aleatórios no grafo. Conforme apresentado na Seção 2.5.1, os passeios aleatórios do node2vec são tendenciosos, uma vez que são guiados por alguns parâmetros. E o cálculo da probabilidade de transição de um vértice para seus vizinhos leva em conta também o peso das arestas.

Como nossa proposta avalia seis diferentes métodos para cálculo de pesos nas arestas utilizando medidas de associatividade de palavras, optamos por utilizar o node2vec para gerar as representações. Com isso, é possível verificar se os diferentes tipos de peso que utilizamos na

abordagem proposta apresentam diferenças nas representações geradas por um algoritmo que leva conta os pesos para fazer a aprendizagem da representação. É possível analisar também se a remoção de arestas tirou dos grafos arestas que eram importantes para a geração de uma boa representação, ou se as arestas removidas podem de fato ser removidas sem causar perda na qualidade da representação.

Além disso, como propomos uma abordagem de filtragem de arestas, podemos avaliar se há redução no tempo de execução e no uso de memória ao realizar a fase de aprendizagem de representação. Pois, como nossos grafos terão menos arestas, consequentemente existem menos possibilidades de passeios aleatórios.

Conforme já apontado anteriormente, a saída de um algoritmo de aprendizagem de representação é um *embedding* (vetor de características) para cada vértice do grafo, e a representação de todo o grafo requer uma matriz de *embeddings*. Assim, como temos um grafo por documento, teremos uma matriz de *embeddings* que representa cada grafo como saída da fase de aprendizagem de representação.

4.4 CNN PARA CLASSIFICAÇÃO

Para a etapa de classificação, utilizamos um modelo de CNN adaptado de Kim (2014), que recebe como entrada a matriz de *embeddings* de cada texto que será classificado. Novamente, a principal diferença comparando com a abordagem de Kim (2014) é que, no lugar de ter uma matriz de *embeddings* com o *embedding* de cada palavra do texto, nossa abordagem terá o *embedding* do vértice correspondente a cada palavra do texto.

4.4.1 *Padding e truncating*

Para garantir que todas as matrizes de *embeddings* tenham o mesmo tamanho antes de submetê-las como entrada à CNN, um valor de limiar é utilizado para garantir sempre o mesmo tamanho das matrizes de embeddings de entrada. Consideramos um valor percentil de 90%, onde tamanho de 90% dos textos do *dataset* estão dentro desse valor. O tamanho de um texto é a quantidade de palavras que ele possui. Os textos com tamanho maior que esse valor de tamanho de 90% dos textos são truncados e os que são menores recebem *padding*. Dessa maneira, completamos com zeros as matrizes de *embeddings* nas quais a quantidade de palavras é menor que o valor em que 90% dos tamanhos dos outros textos da base de dados estão inclusos e removemos as palavras do fim das matrizes nas quais a quantidade de palavras ultrapassa o tamanho de 90% dos outros textos.

Um ponto importante de ressaltar é que o grafo de cada texto é construído sobre o texto completo assim como a execução do node2vec para aprendizagem de representação de cada grafo é feita sobre o grafo do texto completo. Esse passo de cortar ou adicionar linhas na matriz de *embeddings* é feito somente antes da classificação para garantir sempre o mesmo tamanho de entrada na CNN para texto.

4.5 COMPARAÇÃO E ANÁLISES

Para avaliar e comparar a abordagem proposta serão feitas análises sobre o resultado da classificação utilizando o F1-score. Os resultados da classificação dos grafos com pesos e com remoção de arestas serão sempre comparados ao resultado da classificação dos grafos sem peso e sem filtragem de arestas.

Dessa forma, iremos verificar se há diferença entre os resultados das classificações das diferentes representações propostas utilizando pesos para remover arestas e as representações

dadas pelos grafos sem pesos e com todas as arestas. Assim, avaliaremos se é possível obter uma representação mais concisa sem gerar perdas na classificação.

Além disso, um teste estatístico será aplicado para comparar os resultados de nossa abordagem utilizando grafos com peso e remoção de arestas à abordagem utilizando apenas coocorrência para construção do grafo.

Analisaremos o impacto de diferentes porcentagens de corte de arestas e também de diferentes tamanhos de janela de coocorrência para construir os grafos.

Para determinar se na fase de aprendizagem de representação houve algum ganho ao utilizar nossa proposta, analisaremos tempo de execução e uso de memória na execução do node2vec.

4.6 CONSIDERAÇÕES FINAIS

Neste capítulo apresentamos nossa abordagem proposta, que tem o foco principal em utilizar medidas de associatividade de palavras para calcular e atribuir pesos às arestas dos grafos de palavras e utilizar esses pesos para filtrar as arestas de menor peso baseado em um limiar com o objetivo de obter uma representação mais concisa.

O próximo capítulo apresenta os experimentos realizados e os resultados obtidos.

5 EXPERIMENTOS E RESULTADOS

Este capítulo apresenta os experimentos realizados, os resultados obtidos e suas análises. Na Seção 5.1, apresentamos os componentes da implementação proposta que foi desenvolvida. Na Seção 5.2, explicamos a metodologia com que os experimentos foram desenvolvidos como: as bases de dados utilizadas, os grafos de texto propostos, detalhamos todos os parâmetros dos experimentos e quais foram os valores utilizados. Na Seção 5.3, apresentamos todos os resultados e as análises sobre os mesmos. Na Seção 5.4, trazemos uma discussão geral sobre os resultados dos experimentos. Na Seção 5.5, apresentamos as considerações finais.

5.1 IMPLEMENTAÇÃO

Nesta seção apresentamos os componentes da implementação desenvolvida para execução dos experimentos da abordagem proposta.

A implementação desenvolvida possui três componentes principais, conforme apresentados na Figura 5.1, construção da representação, classificação e avaliação. Descrevemos a seguir o que cada um desses componentes implementa e as responsabilidades de cada um deles.

Figura 5.1: Componentes implementados para execução dos experimentos e avaliação

5.1.1 Componente de construção da representação

O componente de construção da representação é responsável pela representação dos textos como grafos. Assim, esse componente recebe como entrada um *dataset* e sua saída são as matrizes de embeddings de cada grafo de texto.

Esse componente implementa os seguintes métodos:

Leitura do *dataset*: lê os arquivos com os textos do *dataset*.

Pré-processamento: realiza o pré-processamento de cada texto do *dataset*, aplicando as seguintes estapas: conversão para letras minúsculas, tokenização, remoção de caracteres não alfabéticos, remoção de *stop words* e *stemming*.

Construção dos grafos: a partir de cada texto, constrói um grafo de palavras que o representa.

Filtragem de arestas: para cada grafo que representa cada texto do *dataset*, calcula e adiciona os pesos às arestas, utilizando a medida de associatividade passada como parâmetro, ordena as arestas a partir do pesos (do menor para o maior) e por fim remove do grafo as arestas conforme a porcentagem de corte p.

Aprendizagem da representação: executa uma instância do node2vec para cada grafo de cada texto da base de dados.

Escrita das matrizes de *embeddings* dos grafos em arquivos: Ao fim da execução de cada instância do node2vec para cada grafo de texto, escreve em um arquivo a matriz de *embeddings* gerada.

Como neste trabalho propomos analisar o tempo de execução e utilização de memória durante a execução do algoritmo node2vec, com o intuito de avaliar se houve algum benefício nesses aspectos com a remoção de arestas, medimos o tempo de execução e a utilização de memória no método que executa o node2vec para cada grafo. Para isso, utilizamos a biblioteca memory_profiler[1]. Essa biblioteca implementa o monitoramento de consumo de memória na execução de programas *python*, que é a linguagem em que a implementação dos experimentos foi desenvolvida. Além disso, essa biblioteca oferece a funcionalidade de monitorar o uso de memória em função do tempo. Assim, medimos a cada 60 segundos o quanto de memória a execução do node2vec está consumindo. Assim, para cada instância dos experimentos, um gráfico mostrando o consumo de memória em função do tempo é reportado nos resultados.

5.1.2 Componente de classificação

O componente de classificação implementa toda a fase de classificação das representações dos grafos como matrizes de *embeddings* utilizando a CNN para texto.

Esse componente implementa então os seguintes métodos:

Leitura das matrizes de *embeddings* dos grafos: Leitura do arquivo com as matrizes de *embeddings* de cada grafo de texto do *dataset*.

Divisão das partições: Dado o *dataset* e uma semente, que é sempre um valor fixo em todos os experimentos (*random_state* = 2), realiza o embaralhamento (*shuffle*) dos exemplos e a divisão do *dataset* em 10 partições.

Classificação: Executa a CNN para texto utilizando validação cruzada estratificada com 10 partições. Assim, sempre 9 partições são utilizadas para treinamento e 1 partição para teste e o processo é repetido 10 vezes para que cada um dos 10 conjuntos seja utilizado para teste.

Escrita dos resultados: Escrita dos 10 valores de taxa de acerto e dos 10 f1-score em um arquivo de resultados para cada instância de teste.

5.1.2.1 Topologia da CNN para texto

A CNN para texto utilizada é composta pela concatenação de três camadas de convolução: a primeira com 32 filtros, a segunda com 64 filtros e a terceira com 128. Cada uma dessas camadas é repetida para dois tamanhos de *kernel*: 2 e 3, resultando em seis camadas de convolução no total. Cada camada de convolução é conectada a uma camada de *dropout* com taxa de 0,2, seguida por uma camada de *pooling*. A saída das camadas de convolução e *pooling* são ligadas a uma camada totalmente conectada com 256 neurônios, seguida por uma camada de *dropout* com taxa de 0,2. No final da rede uma camada *softmax* atribui uma probabilidade a cada classe.

5.1.3 Componente de avaliação

Esse componente implementa a avaliação dos resultados dos experimentos e a comparação através de um teste estatístico dos resultados utilizando os grafos com pesos e com filtragem de arestas com os resultados dos grafos sem pesos e sem remoção de arestas.

Como o componente de classificação tem como saída 10 taxas de acerto e 10 f1-score para cada instância do experimento, na avaliação calculamos a média e o desvio padrão tanto das 10 taxas de acerto quanto dos 10 f1-score.

[1]https://pypi.org/project/memory-profiler/

Tabela 5.1: Informações sobre cada *dataset* utilizado nos experimentos

Dataset	Documentos	Classes
Polarity	2000	2
WebKB	7287	4
R8	7674	8
20 Newsgroups	18846	20

O teste estatístico utilizado foi o *Wilcoxon signed-rank test* [2]. O teste de Wilcoxon é um teste não paramétrico, em que a hipótese nula é de que duas amostras emparelhadas relacionadas vêm da mesma distribuição. A hipótese nula é rejeitada quando o *p-value* é maior que 0,05.

Desse modo o componente de avaliação implementa os seguintes métodos:

Leitura dos arquivos de resultados: Lê o arquivo de resultados para cada instância a ser avaliada.

Cálculo das métricas: Calcula média e desvio padrão das 10 taxas de acerto e 10 dos f1-score para cada instância a ser avaliada.

Testes estatísticos: Executa o teste de Wilcoxon comparando os resultados dos experimentos dos grafos com pesos e filtragem de arestas ao resultado do experimento dos grafos sem pesos.

Gráficos: Plota gráficos para visualização dos resultados dos experimentos.

5.2 METODOLOGIA

Nesta seção apresentamos a metodologia dos experimentos realizados. Primeiramente apresentamos as bases de dados que serão utilizadas, depois novamente mostramos os grafos contidos na proposta com os quais os experimentos serão executados e como eles serão comparados e por fim detalhamos as instâncias dos experimentos e seus parâmetros.

5.2.1 Bases de dados

Os seguintes *datasets* (bases de dados) serão utilizadas nos experimentos:

- **Polarity v2.0:** composto de avaliações de filmes, positivas ou negativas;
- **WebKB:** páginas web coletadas de departamentos de ciência da computação. O *dataset* original contém sete classes, mas utilizamos um subconjunto com quatro classes, que é o subconjunto utilizado em outros trabalhos na literatura;
- **R8:** documentos de notícias das oito classes mais populares do *corpus* Reuters-21578 (coleção de documentos publicados pela agência de notícias Reuters em 1987);
- **20-Newsgroups (20NG):** postagens de notícias separadas por grupos. Utilizamos uma versão deste *dataset* em que as notícias duplicadas foram removidas.

A Tabela 5.1 apresenta a quantidade de documentos e a quantidade de classes de cada *dataset*.

[2]https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.wilcoxon.html

5.2.2 Grafos de texto

Cada texto da base de dados é representado por um grafo. Conforme a proposta, utilizaremos seis medidas para calcular os pesos das arestas dos grafos de palavras: *i*) PMI local; *ii*) PMI global; *iii*) LLR local; *iv*) LLR global; *v*) Chi-square local; e *vi*) Chi-square global.

A base de comparação será sempre os grafos de palavras sem pesos nas arestas e sem filtragem de arestas.

Como os grafos de palavras são gerados utilizando uma janela de coocorrência, o tamanho da janela de coocorrência é um parâmetro dos experimentos. Outro parâmetro dos experimentos é a porcentagem de corte p. Na execução dos experimentos diferentes valores foram considerados para esses parâmetros, como é apresentado na Tabela 5.3 da Seção 5.2.4.

5.2.2.1 Parâmetros do node2vec

Conforme apresentado anteriormente na Seção 2.5.1, o node2vec possui alguns parâmetros que precisam ser configurados. A Tabela 5.2, apresenta esses parâmetros, o que cada um deles significa e os valores atribuídos a cada um deles nos experimentos. Esses valores foram escolhidos após testes com diferentes valores. Esses testes de parâmetros foram realizados apenas com os parâmetros do node2vec, com o objetivo de definir os valores para os parâmetros apenas do node2vec. A configuração final escolhida, conforme a Tabela 5.2, foi utilizada em todas as comparações feitas posteriormente na avaliação experimental.

Parâmetro	**Descrição**	**Valor**
dimensions	dimensão dos *embeddings*	100
walk_length	número de vértices em cada passeio	2
num_walks	número de passeios por vértice	10
p	parâmetro de retorno	2
q	parâmetro de entrada e saída	0.7
workers	número de *workers* para execução paralela	4

Tabela 5.2: Parâmetros do node2vec juntamente com sua descrição e valores utilizados

5.2.3 Avaliação

A avaliação empírica desempenha um papel central na estimativa do desempenho dos sistemas de processamento de linguagem natural (Goutte e Gaussier, 2005). Para avaliar o desempenho de um classificador após a classificação utilizam-se métricas de avaliação.

Dentre as métricas existentes está a taxa de acerto. A taxa de acerto ou acurácia (*accuracy*) é o número de acertos sobre o número total de exemplos.

Como a taxa de acerto (*accuracy*) não considera o número de exemplos por classe ela não é uma boa métrica para avaliar bases de dados desbalanceadas. Para isso, é possível utilizar o *f1-score*, que é calculado com na Equação 5.1.

$$f1 - score = 2 * (precisão * revocação)/(precisão + revocação) \quad (5.1)$$

Onde a precisão é a fração de predições corretas restrita à classe positiva, calculada conforme a Equação 5.2.

$$precisão = (verdadeiros\_positivos/(verdadeiros\_positivos + falsos\_positivos)) \quad (5.2)$$

E a revocação, também conhecida como sensibilidade, dá a porcentagem de positivos detectados, calculada conforme a Equação 5.3.

$$revocação = (verdadeiros\_positivos/(verdadeiros\_positivos + falsos\_negativos)) \tag{5.3}$$

No caso de classificação multi-classe (*dataset* com várias classes), o *f1-score* é a média do *f1-score* para cada classe.

Utilizamos o método de validação cruzada estratificada dividindo os dados sempre em 10 partições, dessa forma, sempre 9 partições são utilizadas para treinamento e 1 partição para teste, e o processo é repetido 10 vezes para que cada um dos 10 conjuntos seja utilizado para teste. Dessa forma, em cada uma das 10 partições calculamos o *f1-score*, e reportamos como resultado final a média dos 10 *f1-scores*.

A seguir, explicamos a execução dos experimentos e os valores utilizados para cada parâmetro.

5.2.4 Execução

A Tabela 5.3 apresenta os parâmetros dos experimentos e seus respectivos valores utilizados. Assim, uma instância de execução é composta da seguinte forma:

Construção da representação: <dataset> <janela de coocorrência> <medida de peso> <corte p>

Classificação: <dataset> <janela de coocorrência> <medida de peso> <corte p>

Avaliação: <dataset> <janela de coocorrência> <corte p>

No componente da avaliação a medida de peso não é um parâmetro porque os resultados de classificação utilizando cada uma das seis medidas para um determinado corte *p* são comparados ao resultado da classificação dos grafos sem pesos e sem remoção de arestas.

Dessa forma, as instâncias de experimentos compõem: 4 *datasets*, 3 tamanhos de janela de coocorrência, 3 valores de corte *p* e 6 medidas de pesos nas arestas. Totalizando assim 216 instâncias de experimentos.

Para a base de comparação que são os grafos sem peso e sem remoção de arestas as instâncias de experimentos compõem: 4 *datasets*, 3 tamanhos de janela de coocorrência. Totalizando 12 instâncias de experimentos como base de comparação.

Parâmetro	**Valores testados**
Tamanho da janela de coocorrência	4, 12, 20
Corte *p*	5%, 10%, 20%
Medida de peso das arestas	PMI local, PMI global, LLR local, LLR global, Chi-square local, Chi-square global

Tabela 5.3: Parâmetros dos experimentos e seus respectivos valores utilizados

A seguir, apresentamos os resultados obtidos nos experimentos e suas análises.

5.3 RESULTADOS

Nesta seção, apresentamos os resultados dos experimentos realizados. Primeiramente, apresentamos todos os resultados analisando apenas o *f1-score* para todas as combinações

de parâmetros, analisando de forma global a variação dos resultados para os diferentes casos testados. Depois, analisamos os resultados olhando separadamente para as diferentes medidas de associatividade, os diferentes tamanhos de janela e para os diferentes percentuais de corte. Por fim, apresentamos os gráficos de tempo de execução por quantidade de memória utilizada para cada instância de teste e analisamos os resultados do ponto de vista do que é mais vantajoso nos fatores tempo de execução e uso de memória.

5.3.1 Resultados de F1-score

A Tabela 5.4 apresenta os resultados do *f1-score* dos experimentos. Estamos sempre comparando os resultados obtidos pelos grafos com peso e com remoção de arestas com os resultados dos grafos sem pesos e sem remoção de arestas (base de comparação). Como executamos os experimentos com três diferentes valores de janela de coocorrência (4, 12 e 20), as comparações no teste estatístico foram feitas da seguinte maneira:

- Comparamos os resultados dos grafos sem pesos gerados com janela de coocorrência de tamanho 4, com os resultados dos grafos com pesos, gerados também com janela de coocorrência de tamanho 4, utilizando cada medida de peso e os diferentes percentuais de corte de arestas.

- Comparamos os resultados dos grafos sem pesos gerados com janela de coocorrência de tamanho 12, com os resultados dos grafos com pesos, gerados também com janela de coocorrência de tamanho 12, utilizando cada medida de peso e os diferentes percentuais de corte de arestas.

- Comparamos os resultados dos grafos sem pesos gerados com janela de coocorrência de tamanho 20, com os resultados dos grafos com pesos, gerados também com janela de coocorrência de tamanho 20, utilizando cada medida de peso e os diferentes percentuais de corte de arestas.

Na Tabela 5.4, os valores em negrito indicam os resultados em que houve diferença estatística significativa com $p<0.05$ no teste de Wilcoxon. Os valores em negrito com o símbolo de − indicam que nesses casos a diferença estatística foi negativa, ou seja, o experimento utilizando a abordagem proposta com aqueles parâmetros obteve um resultado pior que a base de comparação. Os valores em negrito com o símbolo de + indicam que nesses casos a diferença estatística foi positiva, ou seja, o experimento utilizando a abordagem proposta com aqueles parâmetros obteve um resultado melhor que a base de comparação. Novamente, reforçamos que o que procuramos é a estabilidade nos resultados, isto é, que não haja diferença significativa quando filtramos arestas, pois assim verificamos que mesmo com a remoção de um determinado percentual de arestas dos grafos, não houve perda significativa na representação.

As linhas na Tabela 5.4, representam as diferentes medidas de peso, sendo que a primeira linha (Sem peso) mostra os resultados para os experimentos utilizando grafos sem pesos (base de comparação). As linhas da tabela também estão dividindo os resultados por percentual de corte. Como é possível observar, na primeira linha, que apresenta os resultados para os grafos sem pesos, o percentual de corte é 0, porque para a base de comparação não há filtragem de arestas. Já nas outras linhas, para cada medida de associatividade há três linhas, apresentando os resultados para cada percentual de corte (5%, 10% e 20%) para aquela determinada medida de associatividade.

As colunas na Tabela 5.4 representam primeiramente os diferentes *datasets* nos quais os experimentos foram executados. Dentro das colunas de cada *dataset* existem três colunas (4, 12 e 20) que indicam os três tamanhos de janela de coocorrência utilizados para construir os grafos.

Tabela 5.4: Média do f1-score dos 10-folds para cada *dataset* e configuração de parâmetros

		Polarity			**WebKB**			**R8**			**20 Newsgroups**		
		Tamanho da janela			Tamanho da janela			Tamanho da janela			Tamanho da janela		
Grafo	Corte *p*	4	12	20	4	12	20	4	12	20	4	12	20
Sem peso	0	79.83	77.12	79.43	88.48	89.73	90.12	82.23	81.12	84.18	82.90	82.41	83.64
Peso PMI Local	5	79.49	79.42	77.67	89.93	90.13	89.12	83.45	80.56	85.00	**82.15**$^-$	82.61	**82.23**$^-$
	10	79.37	79.19	81.48	89.68	88.95	89.03	83.53	82.44	**80.97**$^-$	82.78	82.81	**82.89**$^-$
	20	80.07	**79.88**$^+$	76.25	89.76	89.34	88.79	79.18	80.40	82.65	**80.87**$^-$	81.97	**81.51**$^-$
Peso PMI Global	5	80.07	78.89	77.97	89.26	89.13	89.60	84.24	82.22	**81.26**$^-$	82.73	82.44	**82.30**$^-$
	10	79.52	77.97	79.28	90.21	89.68	90.03	81.51	82.03	80.01	83.03	82.60	**82.31**$^-$
	20	78.05	79.91	80.19	89.11	90.18	89.03	82.78	80.60	81.70	82.99	82.99	82.99
Peso LLR Local	5	77.89	79.39	78.94	89.83	89.79	89.96	82.71	82.10	**79.79**$^-$	82.68	82.59	**82.45**$^-$
	10	79.00	79.16	79.80	89.67	89.42	88.90	82.26	80.16	**79.25**$^-$	82.83	82.68	82.38
	20	**76.98**$^-$	78.48	78.90	89.84	89.69	89.22	80.43	82.22	**78.96**$^-$	82.35	82.60	**82.21**$^-$
Peso LLR Global	5	79.03	79.65	78.67	**89.99**$^+$	89.58	89.70	84.44	81.50	**79.69**$^-$	82.73	83.22	**82.86**$^-$
	10	78.98	**79.13**$^+$	78.85	90.36	89.73	88.93	81.13	80.74	84.15	82.60	82.65	83.49
	20	79.56	78.80	79.33	89.36	88.72	89.66	82.07	78.93	**81.31**$^-$	83.10	82.84	83.03
Peso CS Local	5	79.77	78.55	77.38	88.87	89.37	89.91	80.90	78.92	**81.15**$^+$	**82.09**$^-$	82.65	**82.46**$^-$
	10	80.85	79.54	79.66	89.29	89.23	89.97	82.93	82.41	81.13	82.34	82.20	**82.64**$^-$
	20	76.85	79.22	78.51	89.05	89.61	89.72	80.69	79.40	85.48	**82.22**$^-$	82.04	**82.45**$^-$
Peso CS Global	5	79.84	79.69	76.36	89.07	89.26	89.66	84.41	83.22	81.98	82.91	**83.16**$^+$	**82.73**$^-$
	10	78.67	78.65	79.35	89.44	89.53	90.52	83.10	81.71	82.33	**82.11**$^-$	82.74	**82.66**$^-$
	20	79.87	79.55	79.02	89.42	89.24	89.63	80.31	81.01	**79.37**$^-$	82.71	**83.29**$^+$	**82.82**$^-$

De maneira geral, podemos observar que os resultados se mantiveram estáveis para a grande maioria das bases de dados, com exceção da 20 Newsgroups, na qual houve muitos resultados negativos estatisticamente. Outra base de dados que apresentou bastante diferença estatística negativa foi a R8 para os grafos gerados com janela de coocorrência 20. A base de dados WebKB foi a base de dados em que a filtragem de arestas conseguiu atingir mais estabilidade, apresentando apenas um resultado com diferença estatística e nesse caso a diferença estatística foi positiva. Na base de dados Polarity os resultados também apresentaram pouca variação, sendo duas delas positivas e uma negativa, mostrando que para esse caso específico de resultado negativo a filtragem causou perda na representação, e que nos demais casos entretanto, utilizando nossa proposta de filtragem, é possível filtrar até 10% das arestas sem causar perdas.

Para todas as bases de dados observamos que, para a janela de coocorrência de tamanho 12, independente da porcentagem de corte e da medida de associatividade utilizada como peso, não houve nenhum resultado significativamente negativo, e obtiveram-se inclusive alguns resultados significativamente melhores que a base de comparação. Observando somente os casos de janela de tamanho 12 podemos ver então que para esse parâmetro de tamanho de janela, independente da variação dos outros parâmetros é possível remover até 20% das arestas dos grafos sem causar perda significativa na representação.

Observando separadamente o parâmetro tamanho da janela de coocorrência, podemos concluir que, para os grafos com pesos, tamanhos de janela maiores não fazem com que o *f1-score* na classificação aumente. E vemos também que tamanhos de janela menores, como 4 e 12, apresentam mais estabilidade nos resultados em termos de *f1-score* na classificação. Contudo, vemos também que a janela de coocorrência de tamanho 12, não apresentou nenhum resultado significativamente pior que a base de comparação, indicando que para essa abordagem de filtragem o tamanho de janela 12 é o ideal para gerar os grafos de palavras.

Observando separadamente o parâmetro de porcentagem de corte de arestas, vemos que não há nenhum caso em que para as variações dos outros parâmetros, esse parâmetro específico se

manteve estável. Dessa forma, observando globalmente, não é possível estimar um valor de corte para o qual, independente da variação dos outros parâmetros, nunca haverá perda significativa. Assim, podemos concluir que a porcentagem de corte é dependente dos parâmetros tamanho da janela de coocorrência e medida de peso.

Observando as medidas de peso, vemos que em todos os casos houve variação estatística tanto positiva como negativa, porém para as medidas locais temos no total 17 valores com diferença estatística negativa e para as medidas globais temos no total 11 valores com diferença estatística negativa. Dessa forma, vemos que, de maneira geral, os pesos calculados globalmente apresentam mais estabilidade nos resultados em comparação aos pesos calculados localmente. Dentre as medidas globais, tanto a PMI global quanto a LLR global obtiveram apenas 3 resultados significativamente negativos, a medida LLR global entretanto obteve também 2 resultados significativamente positivos enquanto que a PMI global não obteve nenhum. Olhando pelo ponto de vista de comparar o número de resultados negativos independente da variação dos outros parâmetros, pode-se afirmar que a LLR global e a PMI global são equivalentes. Se considerarmos porém como critério de desempate o número de resultados significativamente positivos, a medida LLR global comporta-se melhor que a PMI global.

Podemos concluir com essas análises mais gerais que é possível, através da utilização de medidas de associatividade para calcular os pesos das arestas dos grafos de palavras, filtrar as arestas de menores pesos de acordo com uma certa porcentagem. Observamos que construindo os grafos com uma janela coocorrência de tamanho 12, independente a medida de associatividade utilizada para calcular os pesos, é possível utilizar grafos de palavras com 20% menos arestas e ainda obter um resultado de classificação estatisticamente equivalente aos grafos de palavras com todas as arestas.

A seguir, apresentamos análises mais específicas sobre os parâmetros, pontuando comportamentos específicos para cada base de dados.

5.3.2 Análise das medidas de associatividade

Nos experimentos utilizamos três diferentes medidas de associatividade e para cada uma delas calculamos os pesos localmente e globalmente. Conforme apontado anteriormente, não houve uma medida de associatividade na qual os resultados foram sempre estáveis. Observamos porém que nas medidas calculadas globalmente os resultados são mais estáveis que para as medidas calculadas localmente. Isso ocorre provavelmente porque nos grafos com os pesos globais, como os pesos são os mesmos para todas as arestas entre um mesmo par de palavras em todo o grafo, é possível que a informação global tenha feito o node2vec gerar uma representação mais globalmente significativa de cada grafo.

Para a base de dados Polarity, com exceção da medida LLR local, todas as medidas de peso não apresentaram perda significativa nos resultados independente da variação do tamanho da janela de coocorrência que gera os grafos e o percentual de corte. Temos também um resultado estatisticamente melhor que a base de comparação para a medida PMI local e para a medida LLR global. Podemos afirmar portanto que, para a base de dados Polarity, as medidas de associatividade PMI local e global, LLR global e Chi-square local e global são boas métricas para cálculo dos pesos dos grafos de palavras e que até 20% das arestas de menores pesos podem ser removidas de forma que o resultado de classificação seja ainda estatisticamente equivalente aos grafos de palavras sem pesos e com todas as arestas.

Para a base de dados WebKB, todas as medidas de peso não apresentaram perda significativa nos resultados independente da variação do tamanho da janela de coocorrência que gera os grafos e o percentual de corte. Temos também um resultado estatisticamente melhor que a base de comparação para a medida LLR global. Com base nesses resultados, é possível afirmar

que, para a base de dados WebKB, as medidas de associatividade PMI local e global, LLR local e global e Chi-square local e global são boas métricas para cálculo dos pesos dos grafos de palavras e que até 20% das arestas de menores pesos podem ser removidas sem causar perda significativa no resultado da classificação em comparação com os resultados da classificação dos grafos de palavras sem pesos e com todas as arestas.

Para a base de dados R8, com exceção da medida Chi-square local, todas as medidas de peso apresentaram alguma perda significativa nos resultados dependendo da variação do tamanho da janela de coocorrência que gera os grafos e o percentual de corte. Nessa base da dados, a única estabilidade nos resultados em comparação com os resultados dos grafos sem pesos e sem filtragem de arestas foi com o peso Chi-square local, em que todos os resultados foram equivalentes à base de comparação e um resultado foi estatisticamente melhor. Somente para os grafos gerados com janelas de coocorrência 4 e 12, independente do percentual de corte, é que houve estabilidade nos resultados para todas as medidas de peso. Já para a janela de tamanho 20, em todos os casos, com exceção da já apontada anteriormente medida Chi-square local, houve pelo menos um percentual de corte em que os resultados foram significativamente piores que a base de comparação. Em particular, a medida LLR local, para janela de coocorrência de tamanho 20, teve todos os resultados estatisticamente piores independente do percentual de corte.

Para a base de dados 20 Newsgroups, todas as medidas de peso apresentaram alguma perda significativa nos resultados dependendo da variação do tamanho da janela de coocorrência que gera os grafos e o percentual de corte. A medida LLR global, para os cortes 10% e 20%, independente do tamanho da janela de coocorrência, manteve estabilidade nos resultados. Para o corte de 5% porém, houve um caso de resultado estatisticamente negativo para a janela 20. O único caso, nesse *dataset*, em que todas as medidas obtiveram bons resultados foi com a janela de tamanho 12, conforme já apontado anteriormente. Diante desses resultados, para a base de dados 20 Newsgroups, vemos que para os grafos construídos com janela de coocorrência de tamanho 4 e 12, independente do percentual de corte (5%, 10% e 20%), apenas as medidas PMI global e LLR local e global geram resultados estatisticamente equivalentes à base de comparação. Nos demais casos, todas as medidas apresentaram resultados com alguma perda.

5.3.3 Análise do impacto do tamanho da janela

Como já observado anteriormente, para o parâmetro de tamanho da janela de coocorrência que gera os grafos, o tamanho 12, para todas as variações dos demais parâmetros, independente da base de dados, não apresentou nenhum resultado significativamente pior que a base de comparação. Também observamos que tamanhos maiores de janela de coocorrência não implicam em uma melhora nos resultados.

Na base de dados Polarity, com exceção da janela de coocorrência de tamanho 4 para a medida de associatividade LLR local e porcentagem de corte de 20%, todos os resultados foram estaticamente equivalentes à base de comparação. Vemos também dois casos utilizando a janela de coocorrência de tamanho 12 nos quais os resultados foram significativamente melhores que o resultado para os grafos sem pesos e sem remoção de arestas. De maneira geral, para a base de dados Polarity, independente da medida de peso utilizada, para grafos de palavras gerados com janelas de coocorrência 4, 12 e 20, é possível afirmar que pode-se filtrar até 10% das arestas de menores pesos sem causar perda significativa estatisticamente. Pode-se afirmar também, para essa base de dados, que em grafos gerados com janela de coocorrência de tamanho 12 e 20, independente da medida de associatividade utilizada para calcular os pesos das arestas, é possível remover até 20% das arestas de menores pesos sem perda significativa nos resultados da classificação.

Para a base de dados WebKB, em todos os tamanhos de janela de coocorrência utilizados nos experimentos, independente da variação dos outros parâmetros, não houve perda significativa nos resultados. Obteve-se também um resultado estatisticamente melhor que a base de comparação para a janela de tamanho 4. Observa-se que, para essa base de dados, em quaisquer dos tamanhos de janela de coocorrência utilizados nos experimentos para gerar os grafos, é possível remover até 20% das arestas de menores pesos sem perda significativa nos resultados da classificação.

Para a base de dados R8, apenas nos tamanhos de janela de coocorrência de tamanho 4 e 12, é que os resultados, independente da variação dos outros parâmetros, não apresentaram perda significativa em comparação aos resultados dos grafos sem pesos gerados com janelas de 4 e 12. Já para a janela de tamanho 20, na grande maioria dos casos, variando a medida de peso e o percentual de corte, ocorreu alguma perda significativa nos resultados. A única exceção no tamanho de janela 20 para essa base de dados é o peso Chi-square local, que independente da porcentagem de arestas filtradas, não teve nenhum resultado significativamente menor que a base de comparação. A partir desses resultados, podemos concluir que, para a base de dados R8, somente para os grafos gerados com janela de coocorrência de tamanho 4 e 12, independente da medida utilizada para calcular os pesos, é que podemos remover até 20% das arestas de menores pesos sem causar perda significativa na representação.

Para a base de dados 20 Newsgroups, o único tamanho de janela que não teve nenhuma perda significativa nos resultados foi o tamanho 12. Em particular, o tamanho de janela 20, teve resultados significativamente menores que a base de comparação para todas medidas de peso. Já o tamanho de janela 4, apenas utilizando as medidas PMI global e LLR local e global geram resultados estatisticamente equivalentes à base de comparação. Assim, podemos afirmar que para a base de dados 20 Newsgroups, somente utilizando janela de coocorrência de tamanho 12, independente da medida utilizada para calcular os pesos, é que podemos remover até 20% das arestas de menores pesos sem causar perda significativa na representação.

5.3.4 Análise do percentual de corte

Através das análises anteriormente apresentadas, vemos que o percentual de corte é bastante dependente de qual tamanho de janela de coocorrência foi utilizado para construir os grafos e de qual medida de associatividade foi utilizada para calcular os pesos. Dessa forma, não foi possível encontrar um percentual de corte para o qual, em todas as bases de dados, independente da variação do tamanho da janela utilizada para gerar os grafos e da medida de associatividade utilizada para calcular os pesos, os resultados fossem sempre equivalentes à base de comparação.

Para a base de dados Polarity, podemos filtrar até 10% das arestas de menores pesos, independente dos outros parâmetros, sem ter perda na representação. Na base de dados WebKB, podemos remover até 20% das arestas de menores pesos, independente dos outros parâmetros, sem ter perda significativa na representação.

Já para a base de dados R8, vemos que os percentuais de remoção variam os resultados de acordo com a métrica de peso utilizada. Nessa base de dados, nos grafos gerados com janela de coocorrência de tamanho 20, observamos por exemplo que para o peso PMI local temos um resultado estatisticamente pior que a base de comparação removendo 10% das arestas de menores pesos, o que não acontece quando são removidas 20% das arestas de menores pesos em que os resultados são estatisticamente equivalentes à base de comparação.

Para a base de dados 20 Newsgroups, vemos que nos grafos gerados com janela de coocorrência de tamanho 20, para as medidas de pesos PMI local e Chi-square local e global, removendo 5% ou 10% ou 20% das arestas de menores pesos, os resultados são sempre estatisticamente menores que o resultado de classificação da representação do grafo sem pesos e

sem remoção de arestas gerado com janela de coocorrência de tamanho 20. E, como já observado, para grafos gerados com janela de coocorrência de tamanho 12, tanto na 20 Newsgroups como nas outras bases de dados, pode-se filtrar até 20% das arestas de menores pesos sem ter perda significativa na representação.

5.3.5 Análise sobre o tempo de execução e uso de memória

Nesta subseção apresentamos para cada base de dados, medida de peso, tamanho de janela e porcentagem de corte, o tempo de execução e o uso de memória durante a execução do algoritmo node2vec. De forma geral, observamos que o tempo de execução aumenta conforme o tamanho da janela de coocorrência que gera o grafo aumenta. Vemos também que o uso de memória aumenta conforme o tamanho da janela de coocorrência que gera o grafo aumenta. Isso se deve ao fato de que quanto maior a janela de coocorrência, maior o número de arestas para armazenar e, consequentemente, mais possibilidades de passeios aleatórios para o node2vec escolher e processar.

Observamos que, na grande maioria dos casos, o tempo de execução dos grafos com pesos (independente da métrica) é menor que o tempo de execução para os grafos sem pesos, salvo algumas exceções que ocorreram em algumas instâncias para algumas medidas de associatividade. A mesma observação não é verdadeira para o uso de memória, em que na maioria dos gráficos apresentados, vemos que os grafos com pesos utilizam mais memória que os grafos sem pesos. Acreditamos que isso aconteça porque nos grafos com pesos, apesar de ter que armazenar menos arestas, é necessário sempre armazenar os pesos.

5.3.5.1 Medidas de tempo e uso de memória para a base de dados Polarity

Na base de dados Polarity, nos experimentos com os grafos gerados com janela de coocorrência de tamanho 4, observamos pelos gráficos das Figuras 5.2, 5.3 e 5.4, para os três percentuais de corte, que o tempo de execução do node2vec nos grafos com pesos, independente da medida de associatividade, foi sempre menor que o tempo de execução do node2vec nos grafos sem pesos.

No uso de memória, para os grafos gerados com janela de coocorrência de tamanho 4, vemos no gráfico da Figura 5.2, que para o percentual de corte de 5% as curvas de uso de memória dos grafos com pesos, independente de qual medida de associatividade, ficaram superiores à curva do uso de memória da execução do node2vec para os grafos sem pesos.

Já para o percentual de corte de 10% e janela de coocorrência de tamanho 4, vemos no gráfico da Figura 5.3 que as curvas ficaram quase todas sobrepostas, quase não havendo diferença entre o uso de memória pelos grafos com pesos, independente da medida, e os grafos sem pesos. Observamos entretanto, que a curva da medida LLR global nos minutos finais da execução do node2vec disparou no uso de memória, ficando superior a todas as outras curvas.

Para o percentual de corte de 20% e janela de coocorrência de tamanho 4, vemos no gráfico da Figura 5.4, que as curvas de uso de memória dos grafos com pesos, independente de qual medida de associatividade, ficaram inferiores à curva dos grafos sem pesos em quase todo o tempo. Contudo, observamos que a curva da medida PMI local nos minutos finais da execução do node2vec disparou no uso de memória, ficando superior a todas as outras curvas.

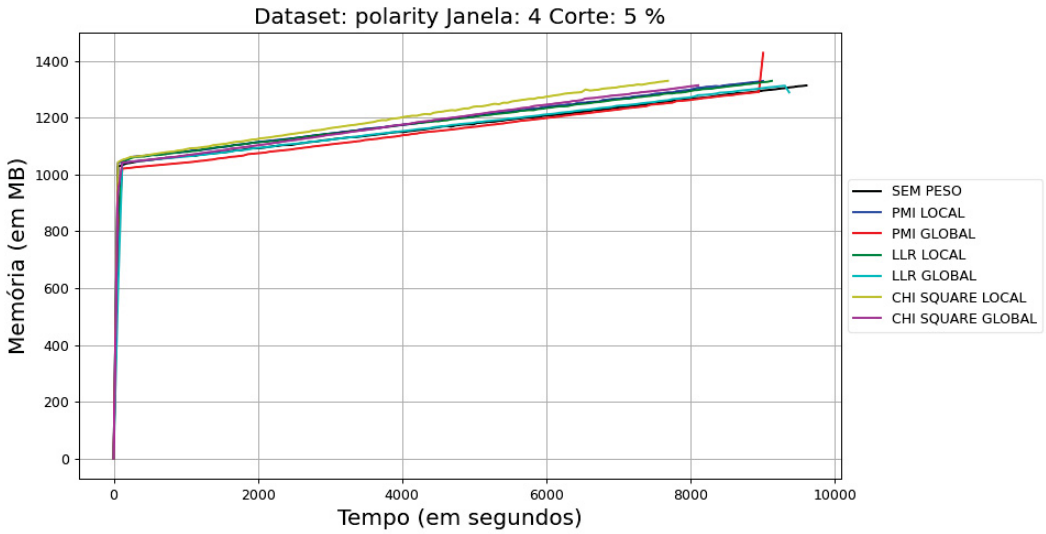


Figura 5.2: Polarity - Janela de coocorrência 4, filtragem de 5% das arestas

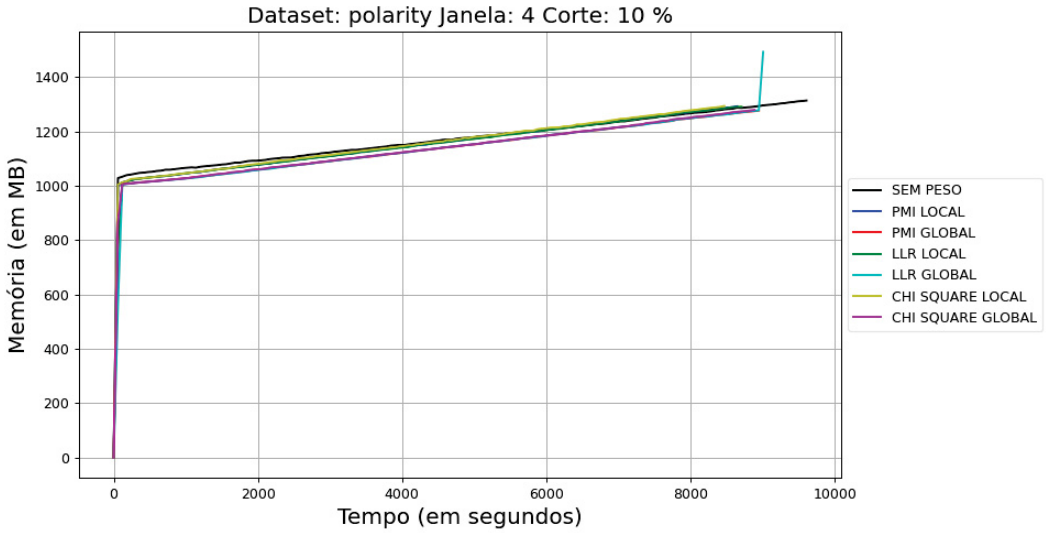


Figura 5.3: Polarity - Janela de coocorrência 4, filtragem de 10% das arestas

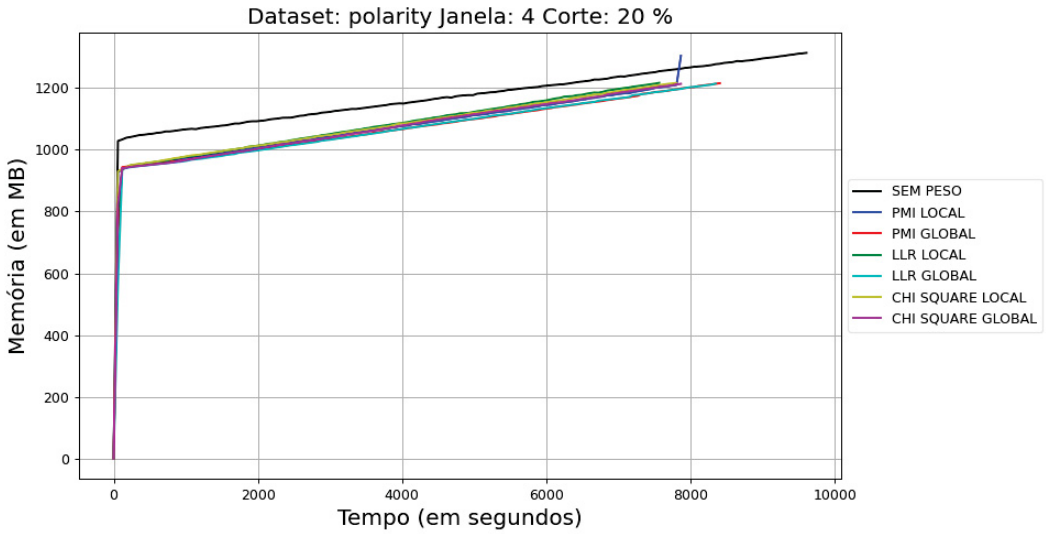


Figura 5.4: Polarity - Janela de coocorrência 4, filtragem de 20% das arestas

Nos experimentos com os grafos gerados com janela de coocorrência de tamanho 12, observamos pelos gráficos da Figuras 5.5, 5.6 e 5.7, que para os três percentuais de corte, o tempo de execução do node2vec nos grafos com pesos, independente da medida de associatividade, foi sempre menor que o tempo de execução do node2vec nos grafos sem pesos.

No uso de memória, para os grafos gerados com janela de coocorrência de tamanho 12, vemos no gráfico da Figura 5.5, que para o percentual de corte de 5% as curvas de uso de memória dos grafos com pesos, com exceção da curva da medida PMI global, ficaram superiores à curva do uso de memória da execução do node2vec para os grafos sem pesos.

Já para o percentual de corte de 10% e janela de coocorrência de tamanho 12, observamos pelo gráfico da Figura 5.6, que as curvas dos grafos com pesos ficaram quase o tempo todo abaixo da curva dos grafos sem pesos, porém, vemos que minutos finais da execução do node2vec, as curvas dos grafos com pesos LLR global e Chi-square disparam no uso de memória e ficam superiores a todas as outras curvas. A curva dos grafos com peso PMI global também dispara nos minutos finais, entretanto não chega a ficar superior à curva dos grafos sem pesos.

Para o percentual de corte de 20% e janela de coocorrência de tamanho 12, vemos no gráfico da Figura 5.7, que as curvas de uso de memória dos grafos com pesos, independente de qual medida de associatividade, ficaram inferiores à curva dos grafos sem pesos em quase todo o tempo. Contudo, observamos que curva da medida PMI global nos minutos finais da execução do node2vec disparou no uso de memória, ficando superior a todas as outras curvas.

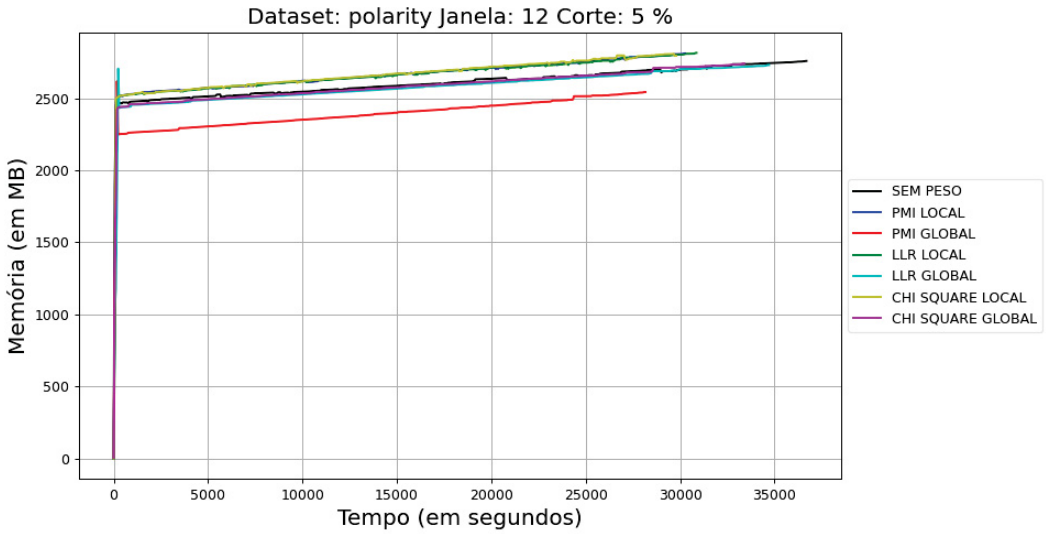


Figura 5.5: Polarity - Janela de coocorrência 12, filtragem de 5% das arestas

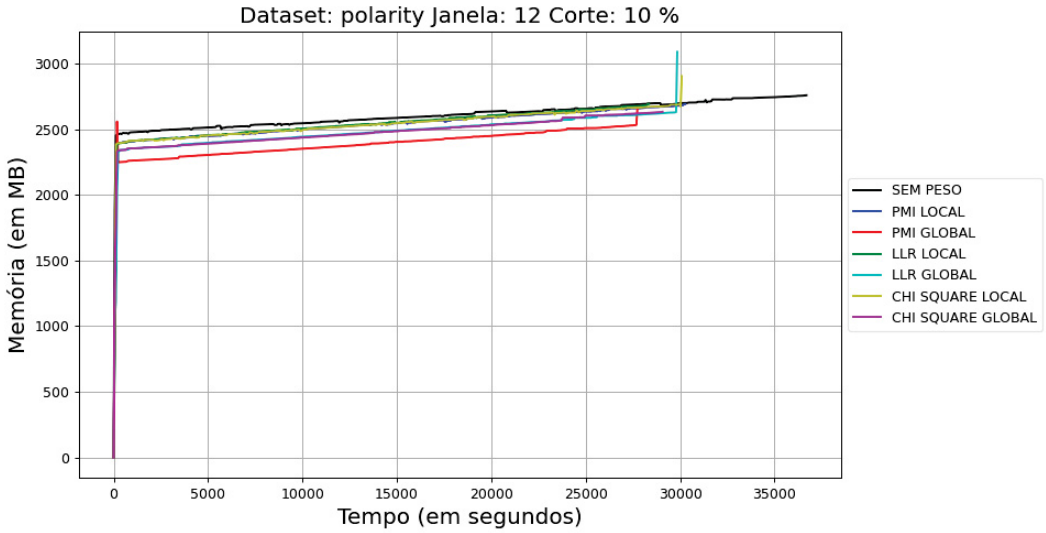


Figura 5.6: Polarity - Janela de coocorrência 12, filtragem de 10% das arestas

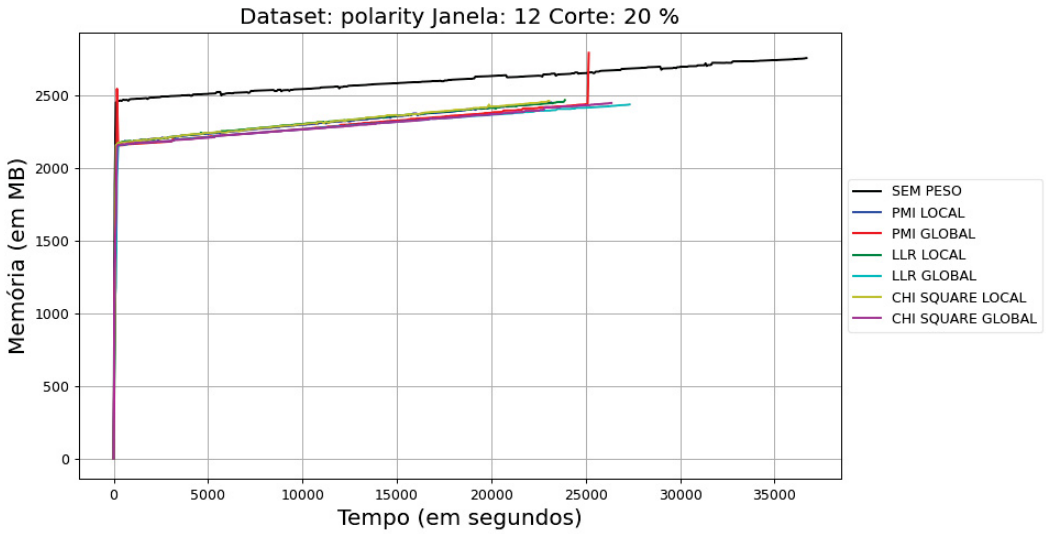


Figura 5.7: Polarity - Janela de coocorrência 12, filtragem de 20% das arestas

Para os grafos gerados com janela de coocorrência de tamanho 20, observamos pelos gráficos da Figuras 5.8, 5.9 e 5.10, para os três percentuais de corte, que temos casos em que o tempo de execução utilizando algumas medidas de pesos não foram inferiores ao tempo de execução do node2vec para os grafos sem pesos e sem filtragem de arestas. Com filtragem de 5% das arestas, isso ocorreu apenas para os grafos com pesos calculados utilizando Chi-square global. Já para a filtragem de 10% das arestas, isso ocorreu para os grafos com pesos calculados utilizando Chi-square local e global, sendo que o tempo de execução do node2vec usando Chi-square global foi maior que utilizando Chi-square local. Para a filtragem de 20% das arestas, isso ocorreu para os mesmos casos que na filtragem de 10% das arestas, sendo novamente o tempo de execução do node2vec usando Chi-square local e global superior ao tempo de execução do node2vec usando todas as outras medidas de pesos e também superior aos grafos sem pesos.

No uso de memória, para os grafos gerados com janela de coocorrência de tamanho 20, vemos no gráfico da Figura 5.8, que para o percentual de corte de 5%, as curvas de uso de memória dos grafos com pesos PMI global e Chi-Square global ficaram inferiores à curva para os grafos sem pesos. As demais curvas ficaram sempre superiores à curva para os grafos sem pesos.

Já para o percentual de corte de 10% e janela de coocorrência de tamanho 20, observamos pelo gráfico da Figura 5.9, que as curvas de uso de memória dos grafos com pesos PMI local e LLR local ficaram superiores à curva para os grafos sem pesos. As demais curvas ficaram sempre inferiores à curva para os grafos sem pesos.

Para o percentual de corte de 20% e janela de coocorrência de tamanho 20, vemos no gráfico da Figura 5.10, que as curvas de uso de memória dos grafos com pesos, independente de qual medida de associatividade, ficaram inferiores à curva dos grafos sem pesos durante todo o tempo.

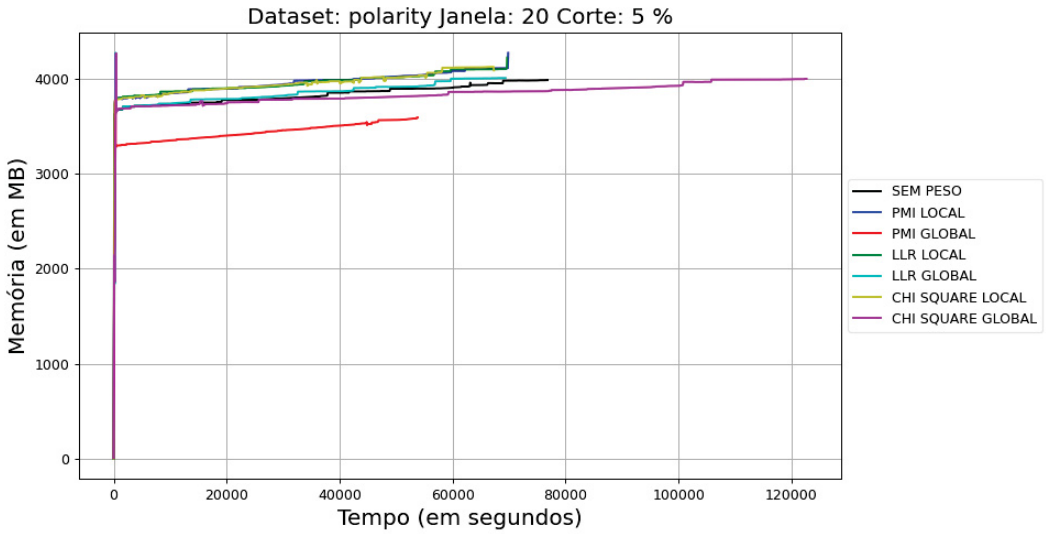


Figura 5.8: Polarity - Janela de coocorrência 20, filtragem de 5% das arestas

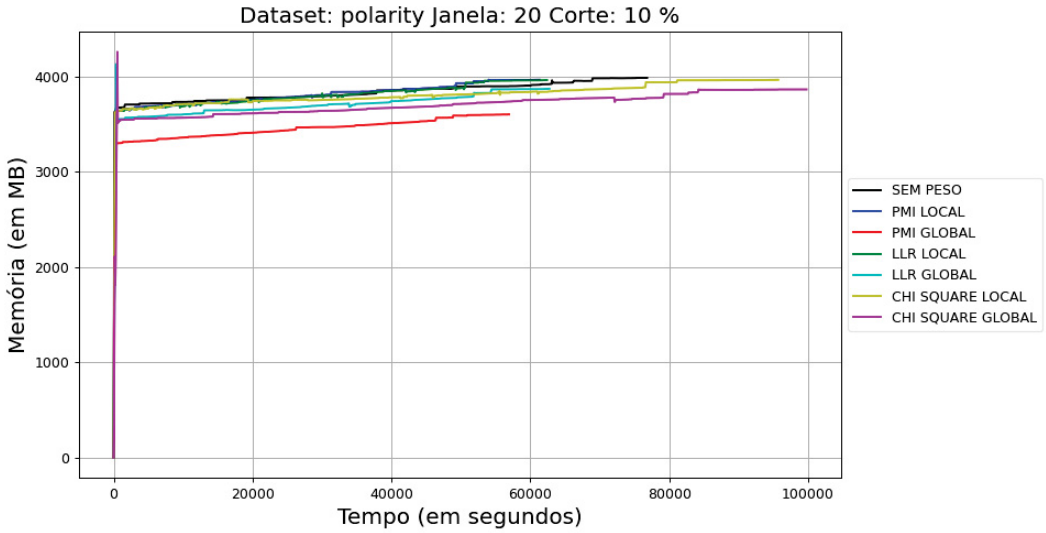


Figura 5.9: Polarity - Janela de coocorrência 20, filtragem de 10% das arestas

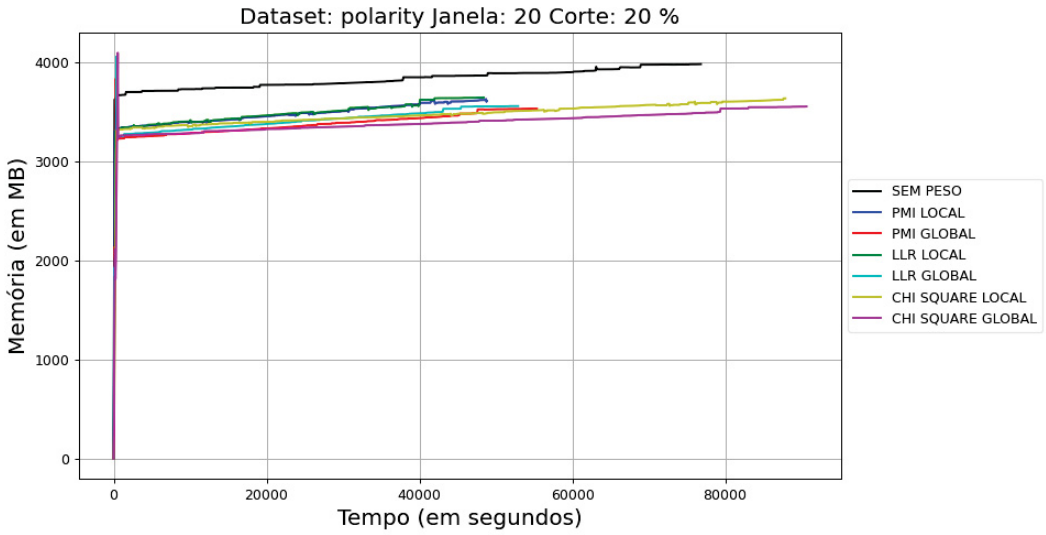


Figura 5.10: Polarity - Janela de coocorrência 20, filtragem de 20% das arestas

Voltando aos resultados de *f1-score*, havíamos observado que para a base de dados Polarity, independente da medida de associatividade utilizada para calcular os pesos e do percentual de corte, construindo os grafos com janelas de coocorrência de tamanho 12 e 20, não havia perda na representação. Olhando para o tempo de execução e o uso de memória sabemos que, dentre essas duas opções, os grafos construídos com janelas de coocorrência de tamanho 12 são mais vantajosos. Observando o uso de memória, para os grafos com pesos construídos com janela 12, nos gráficos das Figuras 5.5, 5.6 e 5.7, vemos que a medida PMI local é a mais vantajosa utilizando as porcentagens de cortes de 5% e 10% e que a medida LLR global é a mais vantajosa utilizando a porcentagem de corte de 20%, essa medida é entretanto, superior as outras em tempo de execução nesse caso específico.

5.3.5.2 Medidas de tempo e uso de memória para a base de dados WebKB

Na base de dados WebKB, nos experimentos com os grafos gerados com janela de coocorrência de tamanho 4, para o percentual de corte de 5% o tempo de execução foi similar na maioria dos casos, conforme apresentamos no gráfico da Figura 5.11, a única exceção foi utilizando a medida PMI global no qual o tempo de execução foi inferior. Para o percentual de corte de 10%, conforme o gráfico da Figura 5.12, o tempo de execução para os pesos calculados com as medidas de associatividade PMI global, LLR local e LLR global foram superiores ao tempo de execução dos grafos sem pesos, nos demais casos o tempo de execução foi inferior. Já para o percentual de corte de 20%, conforme o gráfico da Figura 5.13, em todos os casos o tempo de execução foi menor que o tempo de execução dos grafos sem pesos.

No uso de memória, para os grafos gerados com janela de coocorrência de tamanho 4, vemos no gráfico da Figura 5.11, que para o percentual de corte de 5% as curvas de uso de memória dos grafos com pesos variam bastante e a única medida de associatividade que apresentou vantagem comparada ao uso de memória dos grafos sem pesos foi a PMI global.

Já para o percentual de corte de 10% e janela de coocorrência de tamanho 4, observamos pelo gráfico da Figura 5.12, que as curvas ficaram muito próximas umas das outras, variando muito pouco. Não é possível observar uma curva que seja inferior no uso de memória durante todo o tempo de execução.

Para o percentual de corte de 20% e janela de coocorrência de tamanho 4, vemos no gráfico da Figura 5.13, que as curvas de uso de memória dos grafos com pesos, independente de qual medida de associatividade, ficaram inferiores à curva dos grafos sem pesos em quase todo o tempo. Somente entre os 1000 e 2000 segundos de tempo de execução é que algumas curvas ficam superiores à curva dos grafos sem pesos, mas depois do tempo de execução de 2000 segundos todas as curvas ficam abaixo da curva dos grafos sem pesos.

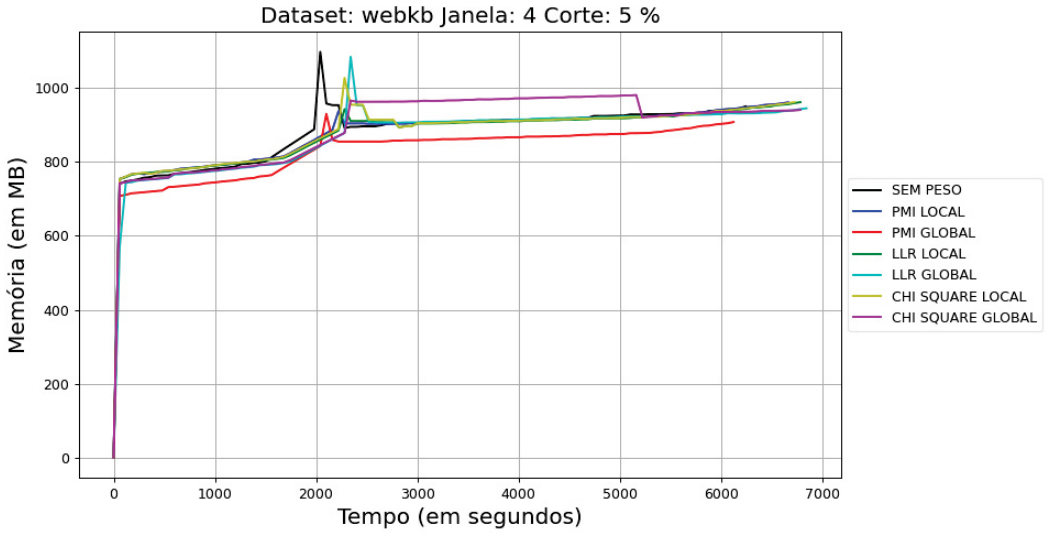


Figura 5.11: WebKB - Janela de coocorrência 4, filtragem de 5% das arestas

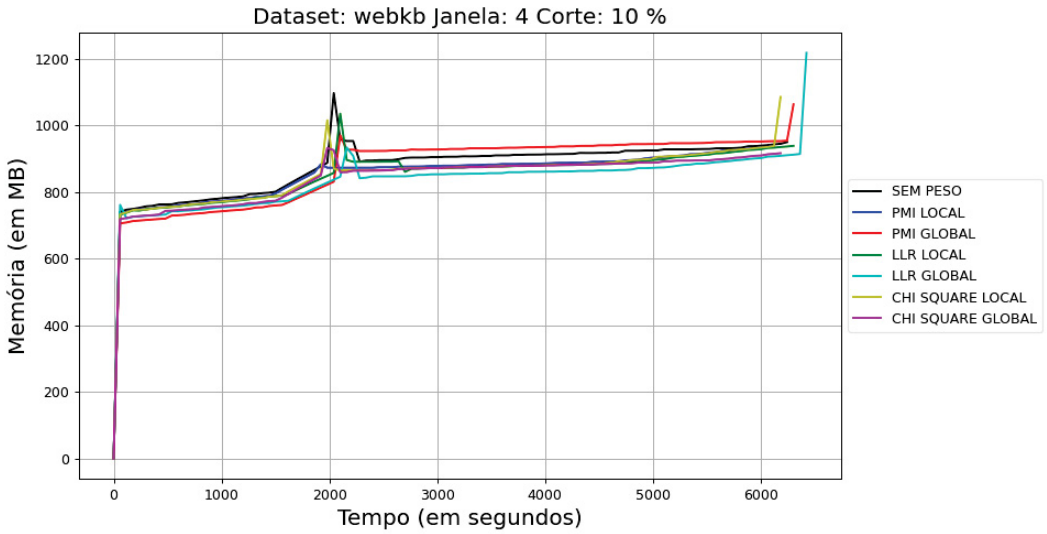


Figura 5.12: WebKB - Janela de coocorrência 4, filtragem de 10% das arestas

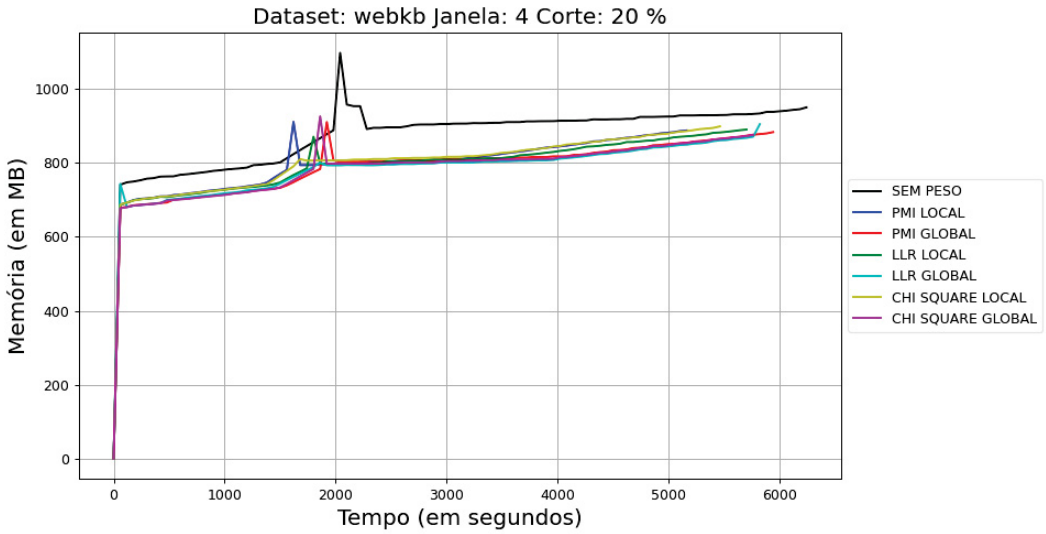


Figura 5.13: WebKB -Janela de coocorrência 4, filtragem de 20% das arestas

Nos experimentos com os grafos gerados com janela de coocorrência de tamanho 12, vemos pelos gráficos da Figuras 5.14, 5.15 e 5.16, que o tempo de execução dos grafos com pesos independente da medida de associatividade e percentual de corte, foi sempre inferior ao tempo de execução no node2vec para os grafos sem pesos.

No uso de memória, para os grafos gerados com janela de coocorrência de tamanho 12, vemos no gráfico da Figura 5.14, que para o percentual de corte de 5% as curvas de uso de memória dos grafos com pesos variam bastante e nenhuma medida de associatividade apresentou vantagem comparada ao uso de memória dos grafos sem pesos, principalmente porque todas as curvas entre 5000 e 1000 segundos são superiores à curva dos grafos sem pesos.

Já para o percentual de corte de 10% e janela de coocorrência de tamanho 12, vemos pelo gráfico da Figura 5.15, que as curvas de uso de memória dos grafos com pesos novamente variam bastante. Observa-se também que a curva da medida de associatividade Chi-square local, continua superior à curva dos grafos sem pesos entre os segundos 15000 e 20000, sendo portanto em uso de memória, dentre todos os métodos de pesos, o mais custoso nesse caso específico.

Para o percentual de corte de 20% e janela de coocorrência de tamanho 12, vemos no gráfico da Figura 5.16, que as curvas de uso de memória dos grafos com pesos, independente de qual medida de associatividade, ficaram superiores à curva dos grafos sem pesos no início do tempo, porém depois dos 10000 segundos de tempo de execução todas as curvas passam a ser inferiores à curva dos grafos sem pesos.

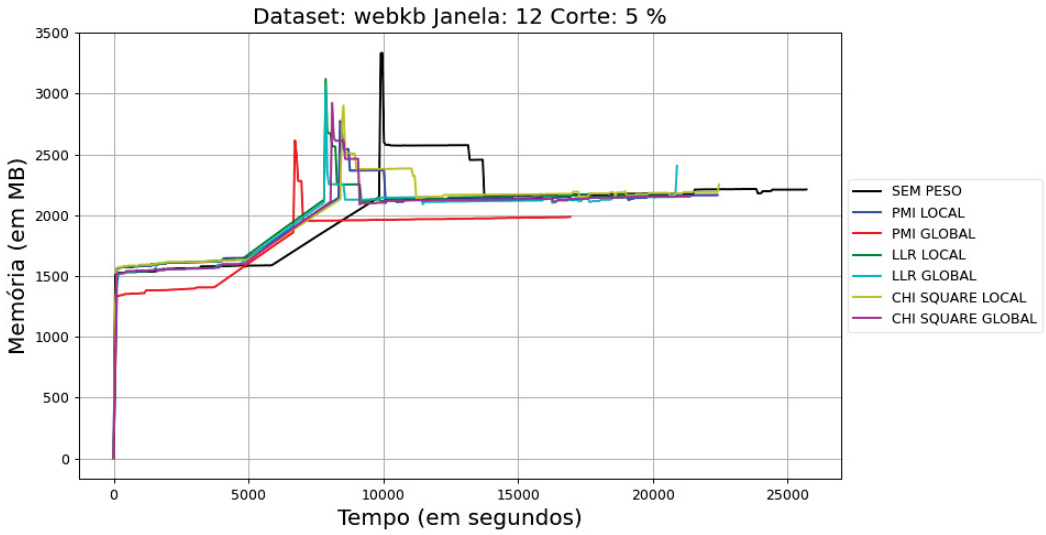


Figura 5.14: WebKB - Janela de coocorrência 12, filtragem de 5% das arestas

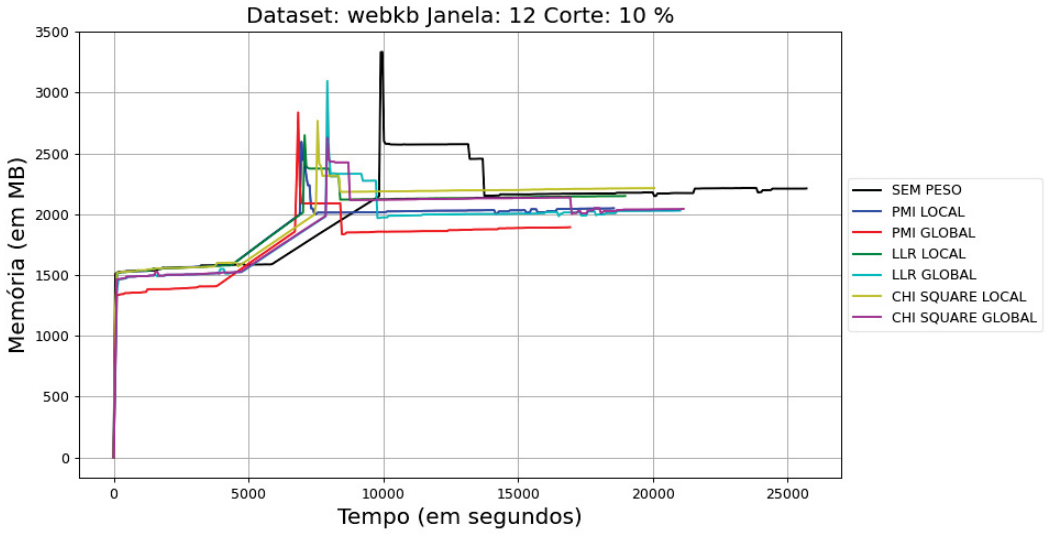


Figura 5.15: WebKB - Janela de coocorrência 12, filtragem de 10% das arestas

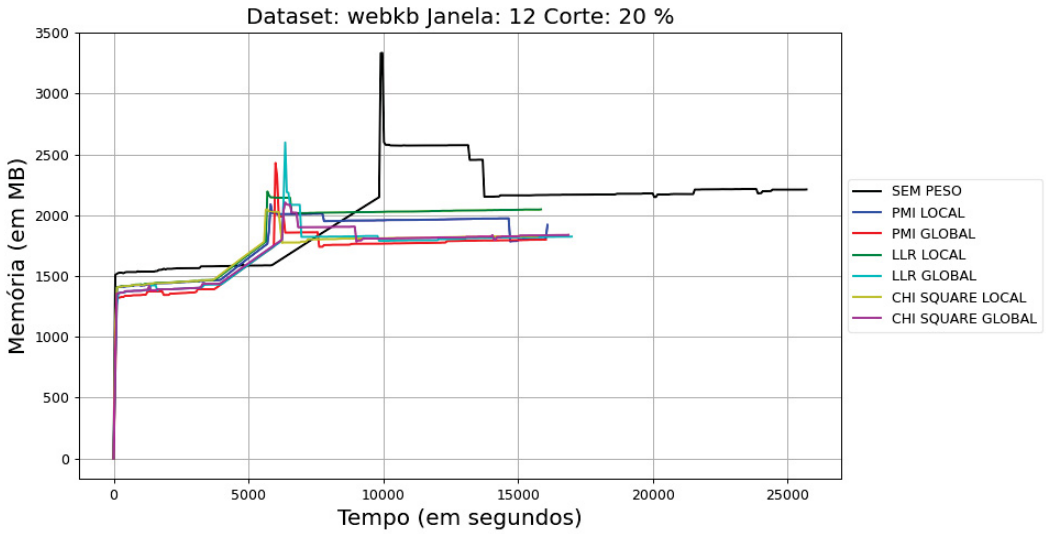


Figura 5.16: WebKB - Janela de coocorrência 12, filtragem de 20% das arestas

Nos experimentos com os grafos gerados com janela de coocorrência de tamanho 20, olhando para o tempo execução pelos gráficos da Figuras 5.17, 5.18 e 5.19, vemos que nos três gráficos, pelo menos em um caso o tempo de execução de alguma medida de peso foi superior ao tempo de execução do node2vec para os grafos sem pesos. Nas três variações de corte, observamos que o node2vec executou mais rápido utilizando como pesos o PMI global.

Olhando para o uso de memória, observamos nos gráficos da Figuras 5.17, 5.18 e 5.19, que as curvas sempre variam bastante para todas as medidas de pesos. É possível notar pelos gráficos que sempre no começo da execução o node2vec para os grafos com pesos utiliza mais memória, mas, conforme já apontado, salvo algumas exceções, os métodos com pesos terminam a execução em menos tempo. Na questão de uso de memória, podemos observar que assim como no tempo de execução, utilizar PMI global se mostra mais vantajoso.

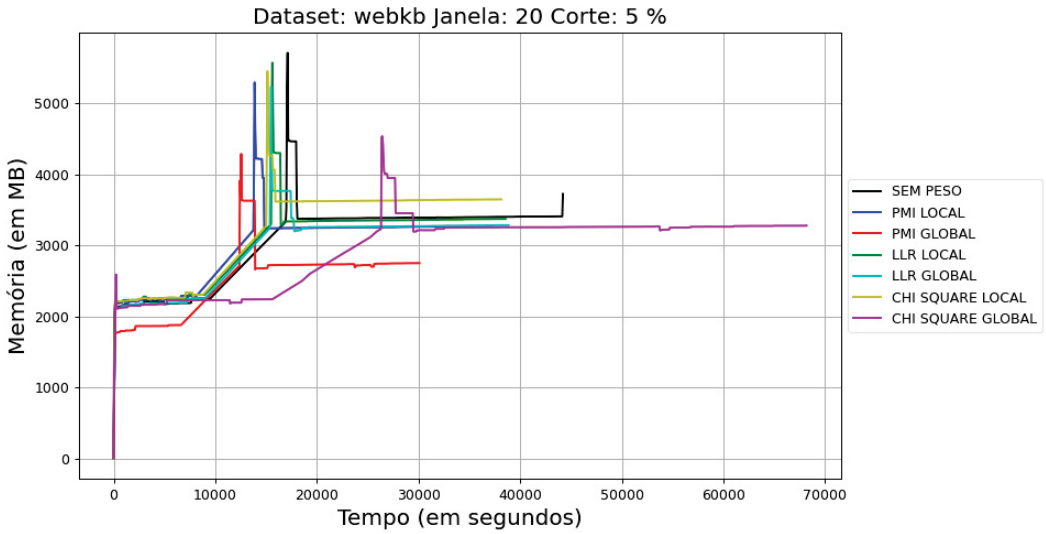


Figura 5.17: WebKB - Janela de coocorrência 20, filtragem de 5% das arestas

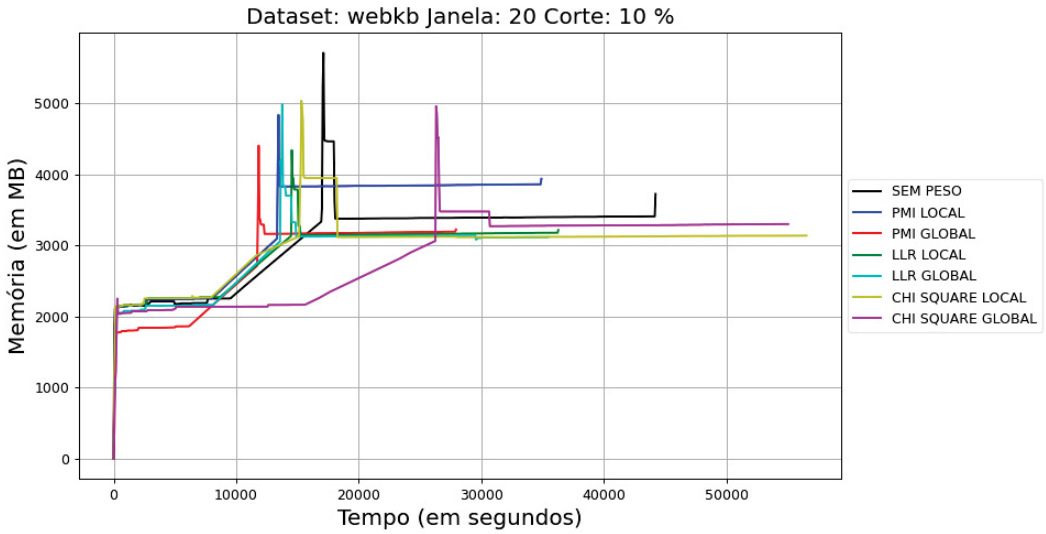


Figura 5.18: WebKB - Janela de coocorrência 20, filtragem de 10% das arestas

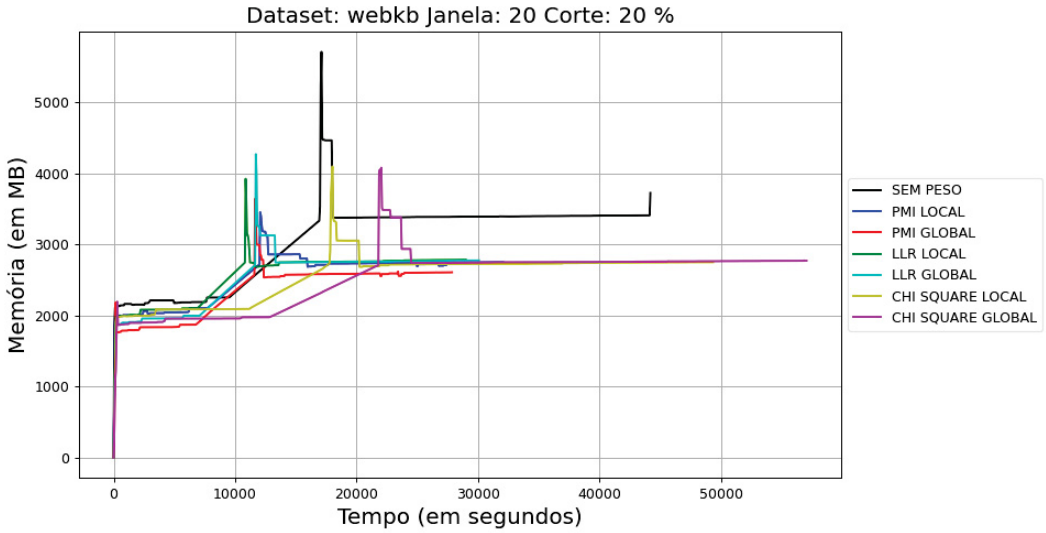


Figura 5.19: WebKB - Janela de coocorrência 20, filtragem de 20% das arestas

Voltando aos resultados de *f1-score*, havíamos observado que para a base de dados WebKB, independente da medida de associatividade utilizada para calcular os pesos, do percentual de corte e do tamanho da janela de coocorrência utilizada para construir os grafos, em nenhum caso havia perda significativa na representação. Olhando para o tempo de execução e o uso de memória sabemos que os grafos construídos com janelas de tamanho 4 e 12 são mais vantajosos. Observando o uso de memória e o tempo de execução, para os grafos com pesos construídos com janela 12, nos gráficos das Figuras 5.14, 5.15 e 5.16, vemos que a medida PMI global é a mais vantajosa utilizando quaisquer porcentagens de cortes 5%, 10% ou 20%.

5.3.5.3 Medidas de tempo e uso de memória para a base de dados R8

Na base de dados R8, nos experimentos com os grafos gerados com janela de coocorrência de tamanho 4, observamos pelos gráficos da Figuras 5.20, 5.21 e 5.22, que para os três percentuais de corte, o tempo de execução do node2vec nos grafos com pesos, independente da medida de associatividade, foi sempre menor que o tempo de execução do node2vec nos grafos sem pesos.

No uso de memória, para os grafos gerados com janela de coocorrência de tamanho 4, observamos pelo gráfico da Figura 5.20, que para o percentual de corte de 5%, as curvas de uso de memória dos grafos com pesos, para a maioria das medidas de associatividade, ficaram iguais ou superiores à curva do uso de memória da execução do node2vec para os grafos sem pesos. A única exceção foi a curva PMI global que ficou sempre inferior as outras curvas.

Já para o percentual de corte de 10% e janela de coocorrência de tamanho 4, vemos no gráfico da Figura 5.21, que as curvas ficaram muito próximas umas das outras, quase não havendo diferença entre o uso de memória pelos grafos com pesos, independente da medida, e os grafos sem pesos.

Para o percentual de corte de 20% e janela de coocorrência de tamanho 4, vemos no gráfico da Figura 5.22, que as curvas de uso de memória dos grafos com pesos, independente de qual medida de associatividade, ficaram inferiores à curva dos grafos sem pesos em todos os casos.

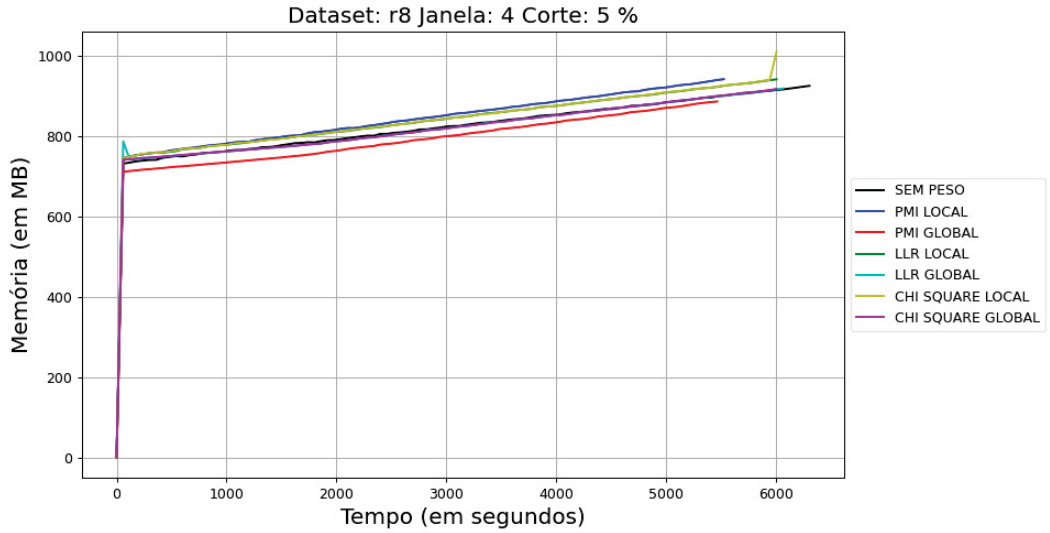


Figura 5.20: R8 - Janela de coocorrência 4, filtragem de 5% das arestas

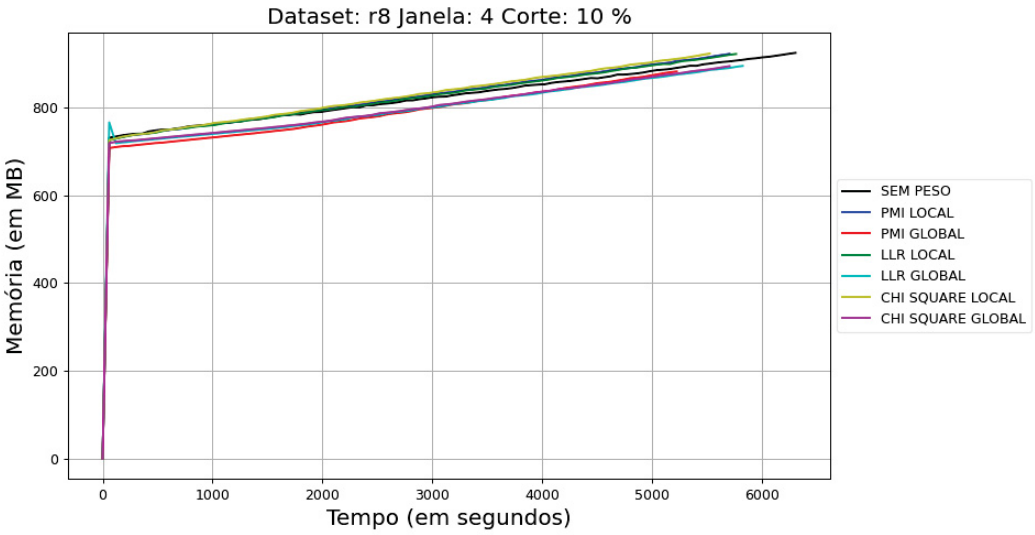


Figura 5.21: R8 - Janela de coocorrência 4, filtragem de 10% das arestas

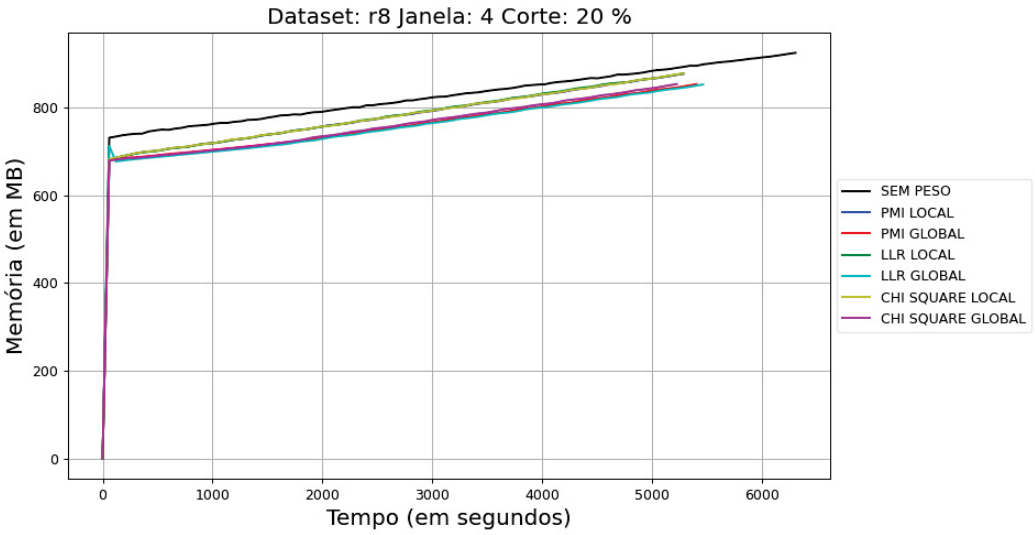


Figura 5.22: R8 - Janela de coocorrência 4, filtragem de 20% das arestas

Nos experimentos com os grafos gerados com janela de coocorrência de tamanho 12, observamos pelos gráficos da Figuras 5.23, 5.24 e 5.25, que para os três percentuais de corte, o tempo de execução do node2vec nos grafos com pesos, independente da medida de associatividade, foi sempre menor que o tempo de execução do node2vec nos grafos sem pesos.

No uso de memória, para os grafos gerados com janela de coocorrência de tamanho 12, vemos no gráfico da Figura 5.23, que para o percentual de corte de 5%, as curvas de uso de memória dos grafos com pesos, para a maioria das medidas de associatividade, ficaram iguais ou superiores à curva do uso de memória da execução do node2vec para os grafos sem pesos. A única exceção foi a curva PMI global que ficou sempre inferior as outras curvas.

Já para o percentual de corte de 10% e janela de coocorrência de tamanho 12, vemos no gráfico da Figura 5.24, que as curvas ficaram muito próximas umas das outras, durante quase todo o tempo, vemos porém que as curvas PMI local e global crescem muito em uso de memória

no final da execução. Para esse caso, é possível notar que utilizar os grafos com peso Chi-square global é mais vantajoso em tempo de execução e uso de memória.

Para o percentual de corte de 20% e janela de coocorrência de tamanho 12, vemos no gráfico da Figura 5.25, que as curvas de uso de memória dos grafos com pesos, independente de qual medida de associatividade, ficaram inferiores à curva dos grafos sem pesos em todos os casos.

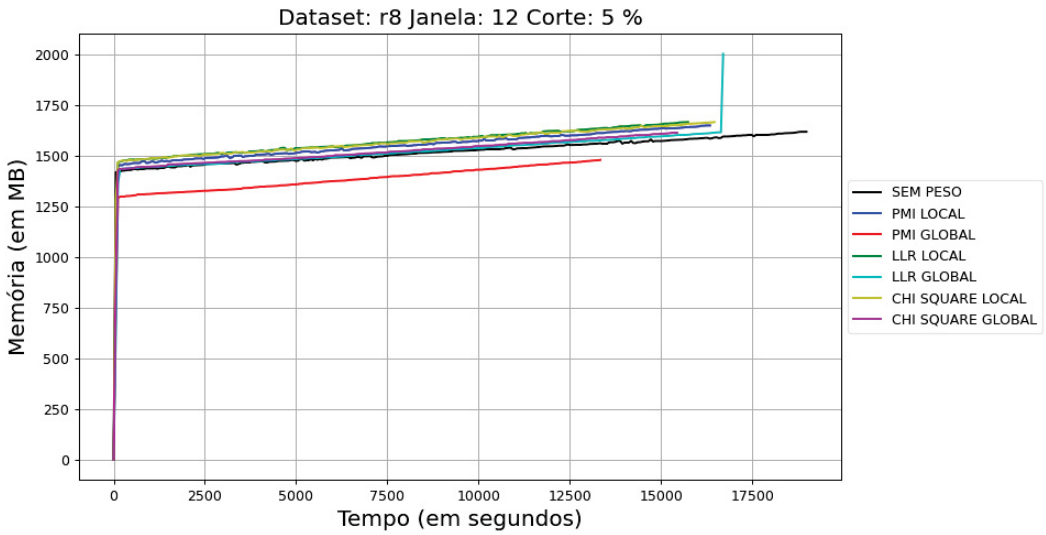


Figura 5.23: R8 - Janela de coocorrência 12, filtragem de 5% das arestas

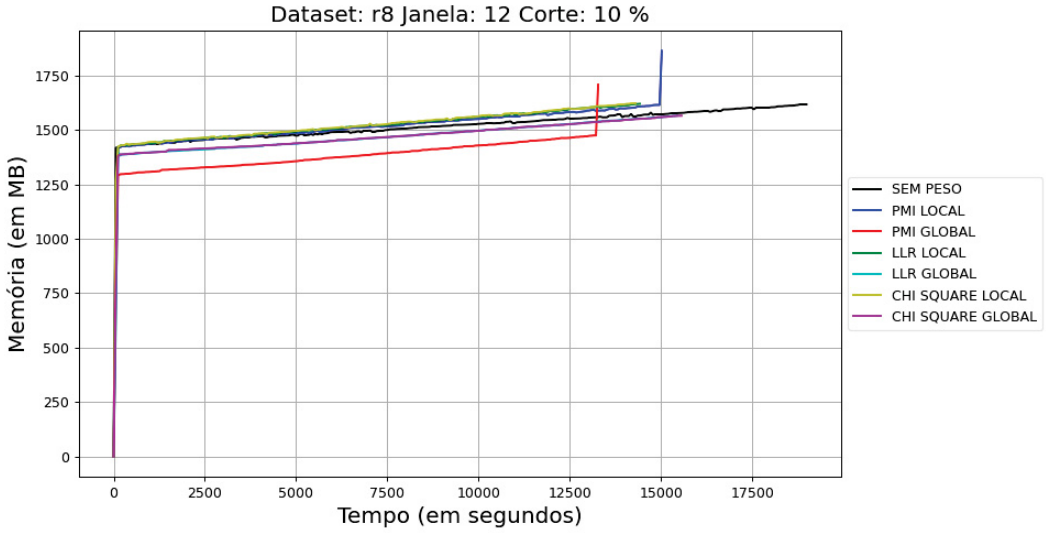


Figura 5.24: R8 - Janela de coocorrência 12, filtragem de 10% das arestas

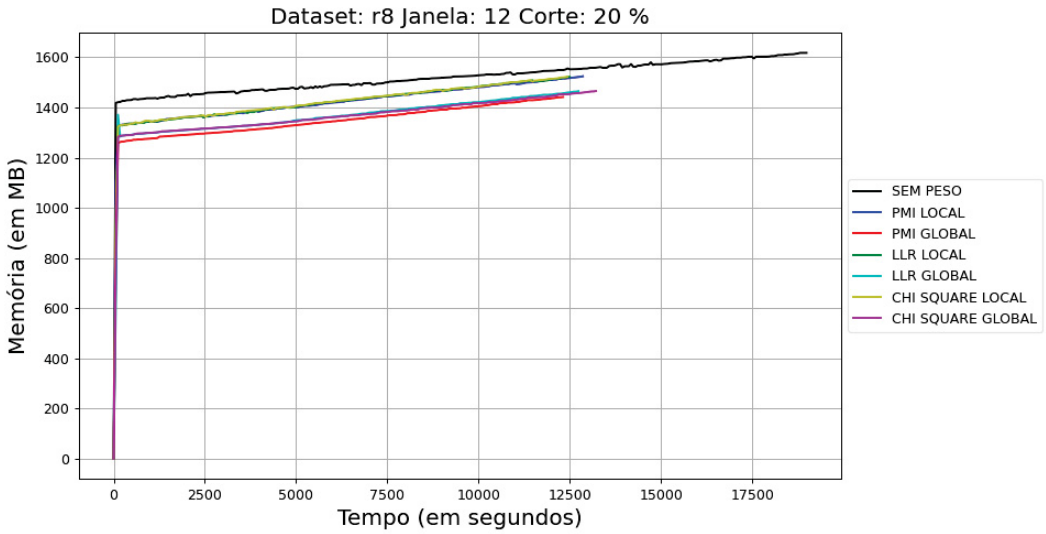


Figura 5.25: R8 - Janela de coocorrência 12, filtragem de 20% das arestas

Nos experimentos com os grafos gerados com janela de coocorrência de tamanho 20, olhando para o tempo execução pelos gráficos da Figuras 5.26, 5.27 e 5.28, vemos que nos três gráficos, pelo menos em um caso o tempo de execução de alguma medida de peso foi superior ao tempo de execução do node2vec para os grafos sem pesos. Nas três variações de corte, observamos que o node2vec executou mais rápido utilizando como pesos o PMI global.

Olhando para o uso de memória, observamos nos gráficos da Figuras 5.26, 5.27 e 5.28, que as curvas não variam muito. Com a filtragem do 20% das arestas, vemos que todas as execuções utilizando grafos com pesos utilizam menos memória que a execução utilizando grafos sem pesos. É possível notar entretanto que as curvas de execução utilizando PMI global nos três casos foi mais vantajosa levando em conta o tempo de execução e o uso de memória.

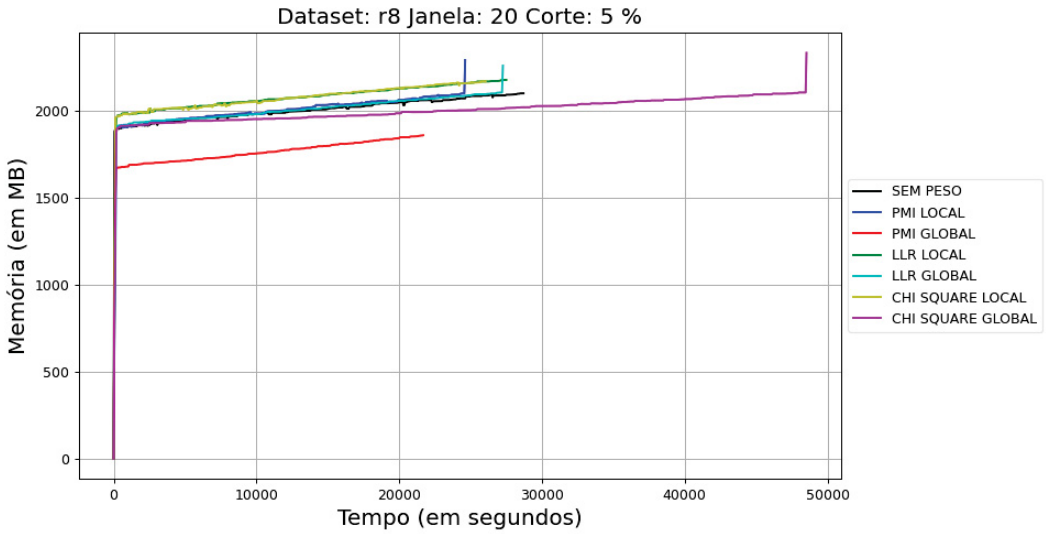


Figura 5.26: R8 - Janela de coocorrência 20, filtragem de 5% das arestas

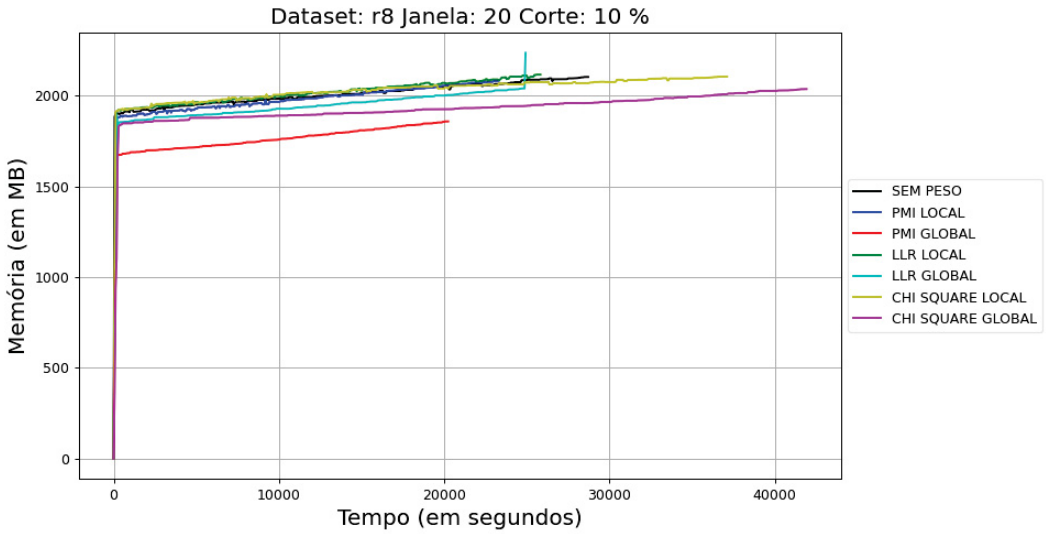


Figura 5.27: R8 - Janela de coocorrência 20, filtragem de 10% das arestas

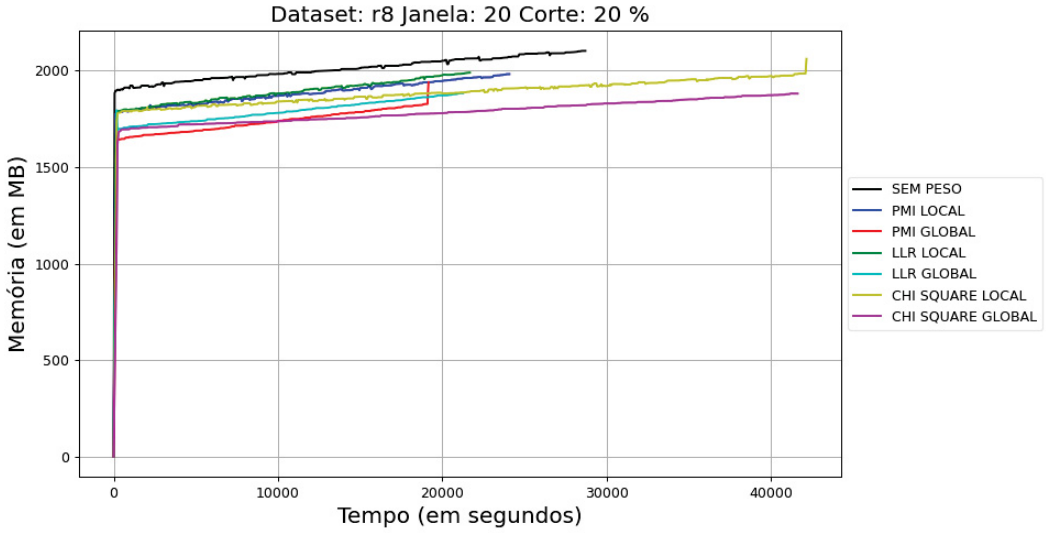


Figura 5.28: R8 - Janela de coocorrência 20, filtragem de 20% das arestas

Voltando aos resultados de *f1-score*, havíamos observado que para a base de dados R8, independente da medida de associatividade utilizada para calcular os pesos e do percentual de corte, nos grafos gerados utilizando a janela de coocorrência de tamanho 4 ou 12, não havia perda significativa na representação. Olhando para o tempo de execução e o uso de memória sabemos que os grafos construídos com janelas de tamanho 4 e 12 e porcentagem de corte de 20% são mais vantajosos. Sabemos também que um tamanho de janela menor implica em tempo de execução menor e menos uso de memória, portanto podemos afirmar que utilizando janela de coocorrência de tamanho 4 e filtrando 20% das menores arestas, independente da medida utilizada para calcular os pesos, temos uma boa configuração de parâmetros para a base de dados R8. A segunda melhor configuração seria então utilizar janela de coocorrência de tamanho 12 para gerar os grafos e filtrar 20% das menores arestas. Podemos observar também que a medida PMI global se mostrou a mais vantajosa para a base de dados R8.

5.3.5.4 *Medidas de tempo e uso de memória para a base de dados 20 Newsgroups*

Na base de dados 20 Newsgroups, nos experimentos com os grafos gerados com janela de coocorrência de tamanho 4, observamos pelos gráficos da Figuras 5.29, 5.30, que o tempo de execução do node2vec nos grafos com pesos, filtrando 5% ou 10% das arestas com os menores pesos, em vários casos foi superior a base de comparação. Somente quando são filtradas 20% das arestas com os menores pesos, conforme a Figura 5.31, é que o tempo de execução de todos os experimentos utilizando grafos com pesos fica inferior a base de comparação.

No uso de memória, para os grafos gerados com janela de coocorrência de tamanho 4, vemos no gráfico da Figura 5.29, que para o percentual de corte de 5% a maioria curvas de uso de memória dos grafos com pesos, em algum ponto fica superior à curva dos grafos sem pesos. A única exceção é a curva PMI global que sempre está abaixo da curva dos grafos sem pesos. Porém, podemos ver que isso não é uma total vantagem dado que o tempo de execução do PMI global foi muito superior ao tempo de execução da base de comparação.

Já para o percentual de corte de 10% e janela de coocorrência de tamanho 4, vemos no gráfico da Figura 5.30, que as curvas dos grafos com pesos ficaram durante quase toda a execução abaixo da curva dos grafos sem pesos, mas que no final da execução a maioria delas passa a usar muita memória. Nesse caso, não houve vantagem em utilizar grafos com pesos.

Para o percentual de corte de 20% e janela de coocorrência de tamanho 4, vemos no gráfico da Figura 5.31, que as curvas de uso de memória dos grafos com pesos, com exceção da curva PMI local e Chi-square local, ficaram inferiores à curva dos grafos sem pesos em todo o tempo. E nesses casos o tempo de execução também foi inferior ao tempo de execução dos grafos sem pesos. Observamos então que somente os grafos com pesos utilizando PMI local e Chi-square local foram desvantajosos em termos de utilização de memória nesse caso.

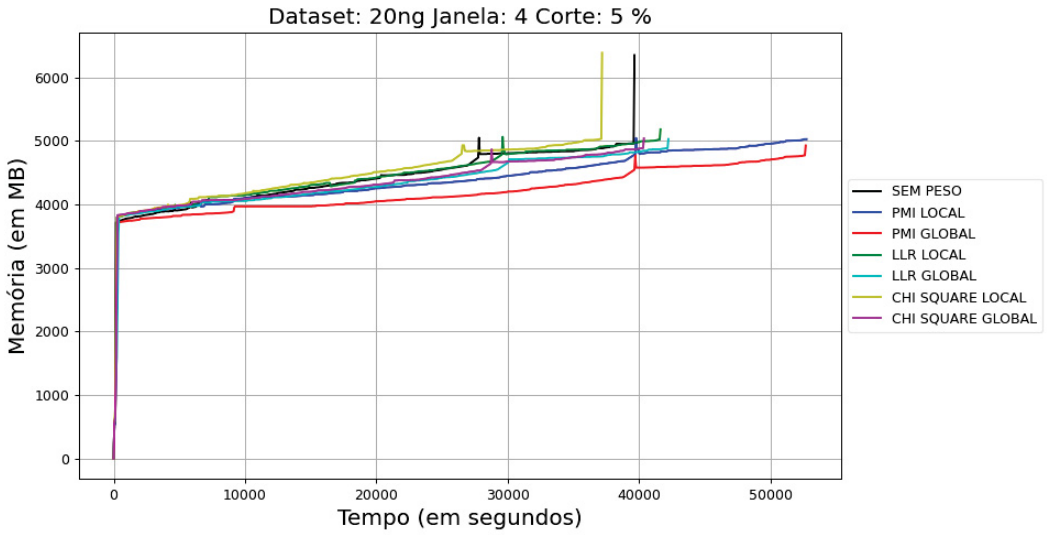


Figura 5.29: 20 Newsgroups - Janela de coocorrência 4, filtragem de 5% das arestas

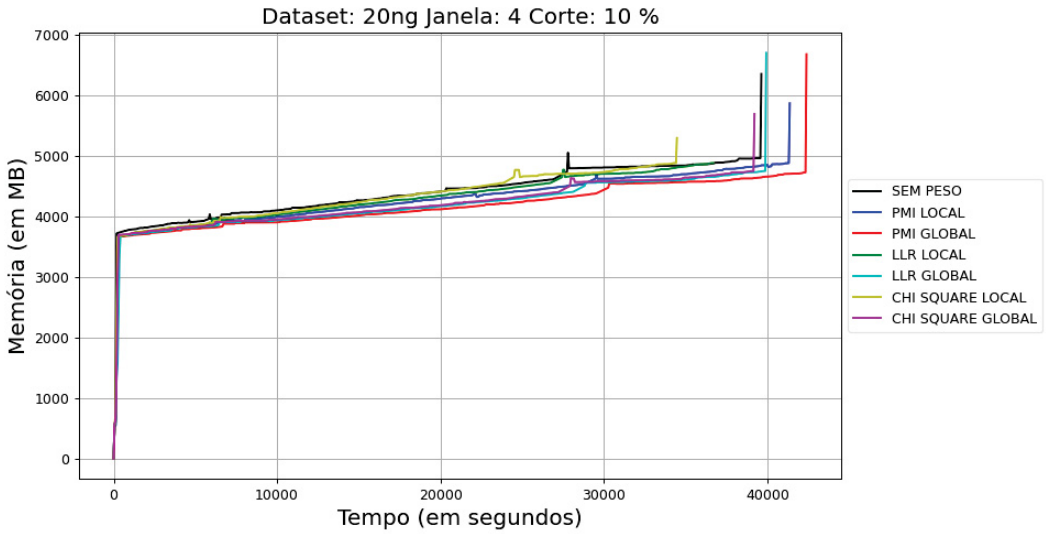


Figura 5.30: 20 Newsgroups - Janela de coocorrência 4, filtragem de 10% das arestas

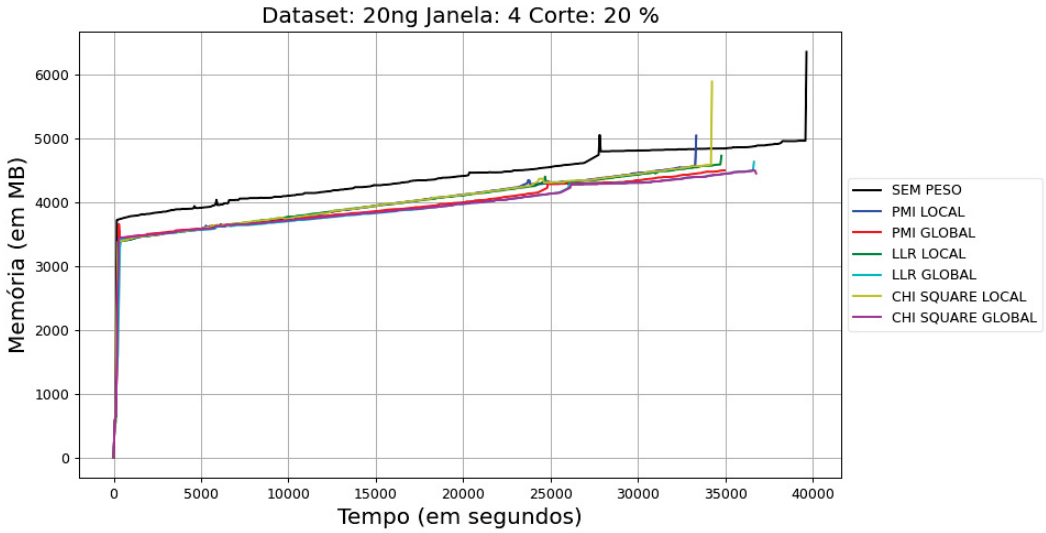


Figura 5.31: 20 Newsgroups - Janela de coocorrência 4, filtragem de 20% das arestas

Nos experimentos com os grafos gerados com janela de coocorrência de tamanho 12, observamos pelo gráfico da Figura 5.32, que o tempo de execução do node2vec nos grafos com pesos, filtrando 5% das arestas com os menores pesos, em vários casos foi superior à base de comparação. E, nesse caso específico a medida PMI global foi a que terminou a execução em menos tempo que todos as outras. Já quando filtramos 10% ou 20% das arestas com os menores pesos, conforme os gráficos das Figuras 5.33 e 5.34, o tempo de execução de todos os experimentos utilizando grafos com pesos fica inferior a base de comparação.

No uso de memória, para os grafos gerados com janela de coocorrência de tamanho 12, observamos pelo gráfico da Figura 5.32, que para o percentual de corte de 5% a maioria curvas apresentam pequenos picos em certos intervalos. Nesse caso, a única medida de associatividade que se mostrou um pouco melhor foi a PMI global, mas podemos ver que ela não é vantajosa

durante toda a execução já que em um momento entre os segundos 80000 e 100000 ela apresenta um pico no uso de memória.

Para o percentual de corte de 10% e janela de coocorrência de tamanho 12, vemos pelo gráfico da Figura 5.33, que as curvas também apresentam pequenos picos em certos intervalos. Novamente, a medida de associatividade que se mostrou mais vantajosa foi a PMI global, mas é possível notar que ela não é vantajosa durante toda a execução devido aos dois picos no uso de memória.

Para o percentual de corte de 20% e janela de coocorrência de tamanho 12, observamos no gráfico da Figura 5.34, que as curvas de uso de memória dos grafos com pesos, mantém-se em quase todo o tempo, com exceção de alguns picos, inferiores à curva dos grafos sem pesos. Nesse caso, se olhamos apenas para o uso de memória a medida mais vantajosa é a PMI global, porém levando em conta o tempo de execução a medida LLR local se mostra mais vantajosa.

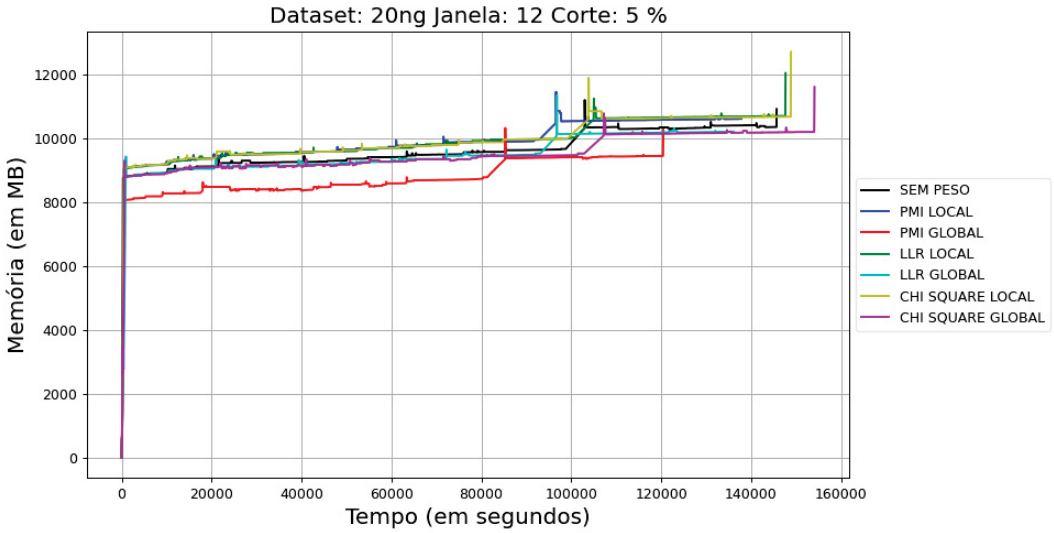


Figura 5.32: 20 Newsgroups - Janela de coocorrência 12, filtragem de 5% das arestas

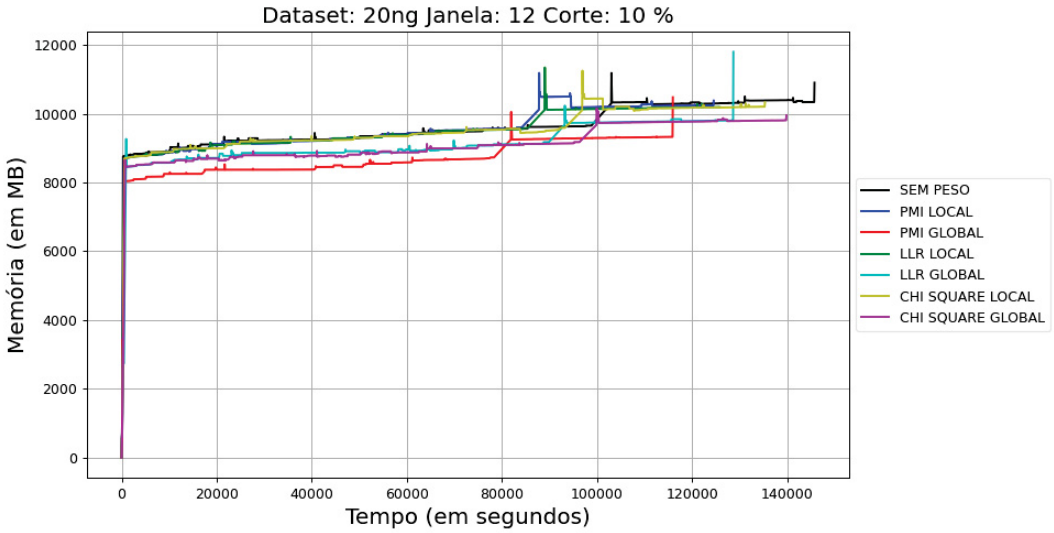


Figura 5.33: 20 Newsgroups - Janela de coocorrência 12, filtragem de 10% das arestas

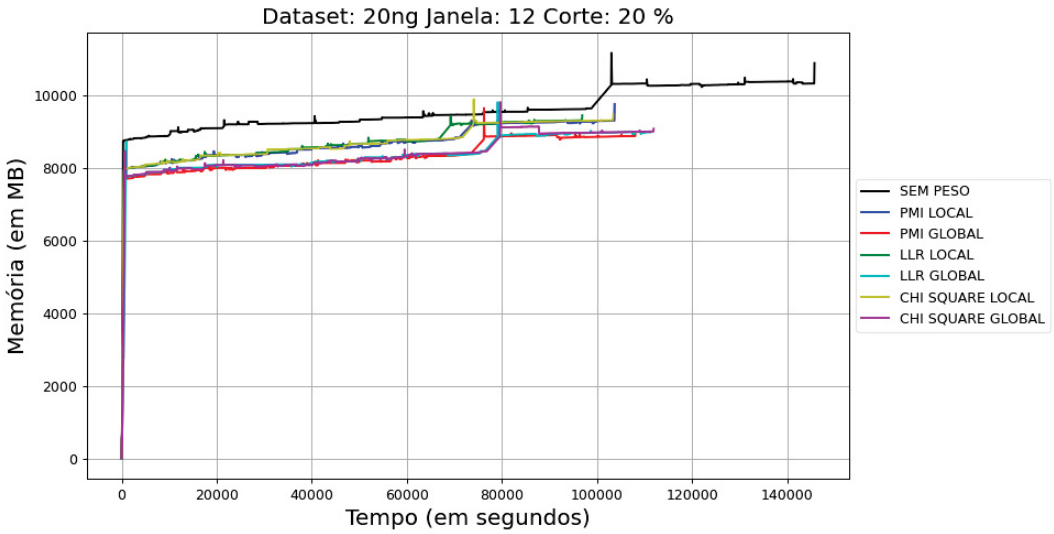


Figura 5.34: 20 Newsgroups - Janela de coocorrência 12, filtragem de 20% das arestas

Nos experimentos com os grafos gerados com janela de coocorrência de tamanho 20, observamos pelo gráfico da Figura 5.35, que o tempo de execução do node2vec nos grafos com pesos, filtrando 5% das arestas com os menores pesos, utilizando PMI local foi superior à base de comparação e nos demais casos inferior. Já quando filtramos 10% ou 20% das arestas com os menores pesos, conforme os gráficos das Figuras 5.36 e 5.37, o tempo de execução de todos os experimentos utilizando grafos com pesos fica inferior a base de comparação.

No uso de memória, para os grafos gerados com janela de coocorrência de tamanho 20, vemos pelo gráfico da Figura 5.35, que para o percentual de corte de 5% a maioria curvas apresentam pequenos picos em certos intervalos e um grande pico sempre superior à curva dos grafos sem pesos. Nesse caso, a única medida de associatividade que se mostrou um pouco melhor foi a PMI global, mas podemos ver que ela não é vantajosa durante toda a execução já que em um momento entre os segundos 100000 e 150000 ela apresenta um pico no uso de memória.

Para o percentual de corte de 10% e janela de coocorrência de tamanho 20, vemos no gráfico da Figura 5.36, que as curvas do uso de memória também apresentam pequenos picos em certos intervalos e um pico maior em algum momento da execução. Novamente, a medida de associatividade que se mostrou mais vantajosa foi a PMI global, mas é possível notar que ela não é vantajosa durante toda a execução devido ao pico no uso de memória entre os segundos 100000 e 150000.

Para o percentual de corte de 20% e janela de coocorrência de tamanho 20, observamos no gráfico da Figura 5.37, que as curvas de uso de memória dos grafos com pesos, mantém-se em quase todo o tempo, com exceção de alguns picos, inferiores à curva dos grafos sem pesos. Nesse caso, se olhamos apenas para o uso de memória a medida mais vantajosa é novamente a PMI global.

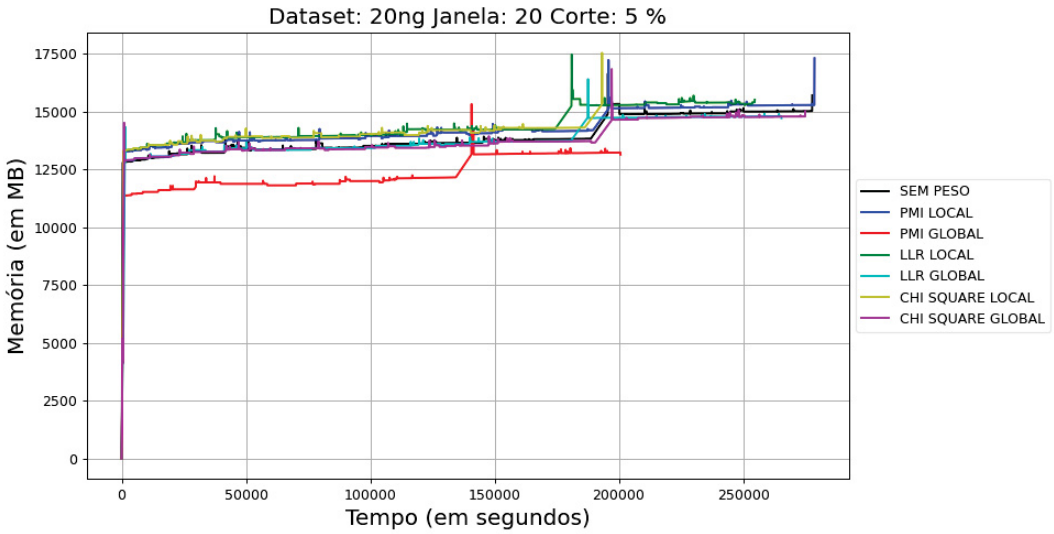


Figura 5.35: 20 Newsgroups - Janela de coocorrência 20, filtragem de 5% das arestas

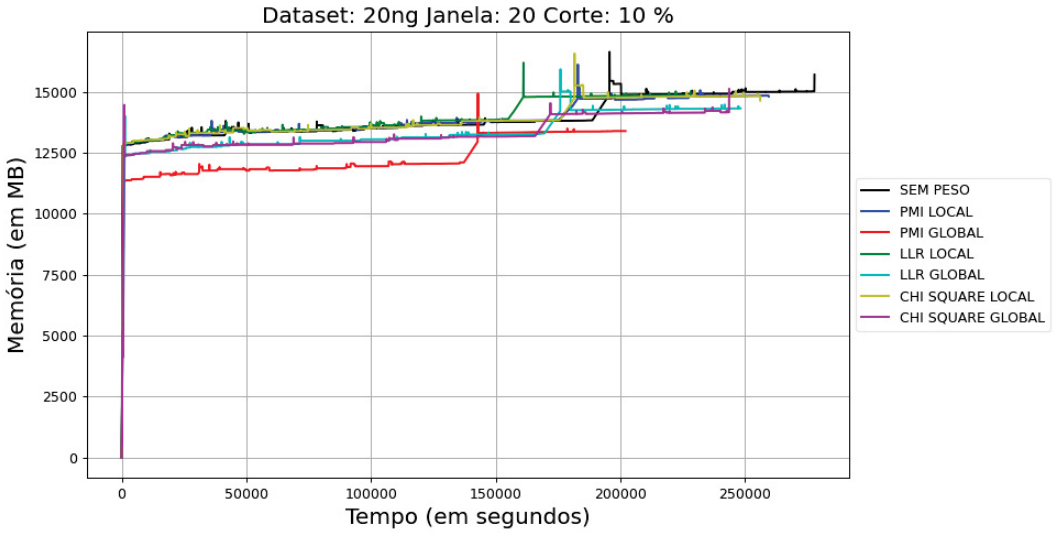


Figura 5.36: 20 Newsgroups - Janela de coocorrência 20, filtragem de 10% das arestas

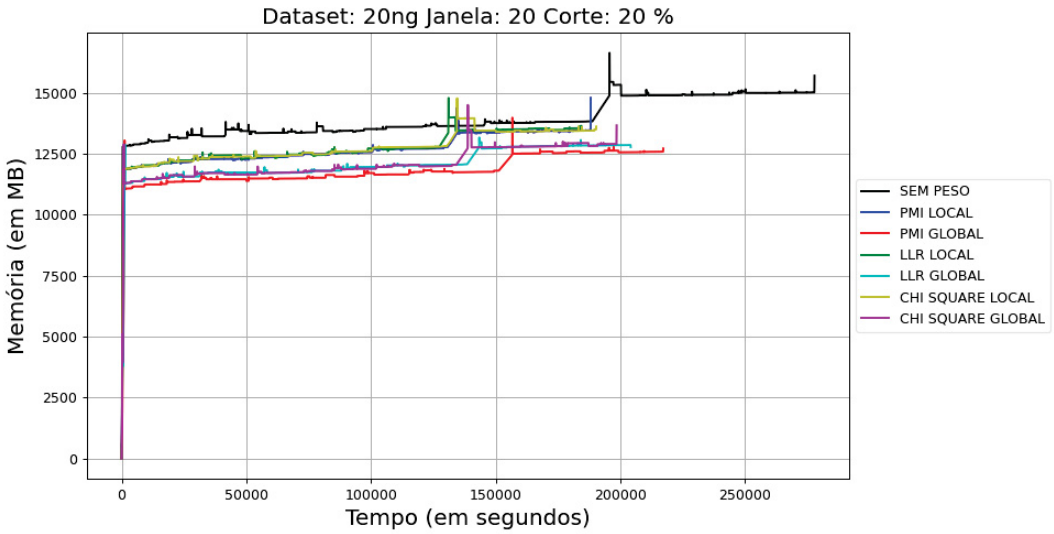


Figura 5.37: 20 Newsgroups - Janela de coocorrência 20, filtragem de 20% das arestas

Voltando aos resultados de *f1-score*, havíamos observado que para a base de dados 20 Newsgroups, independente da medida de associatividade utilizada para calcular os pesos e do percentual de corte, apenas nos grafos gerados utilizando a janela de coocorrência de tamanho 12, não havia perda na representação. Olhando para o tempo de execução e o uso de memória para nos experimentos com janela de coocorrência de tamanho 12, vemos que filtrando 20% das arestas de menores pesos, em todas as medidas de pesos, salvo alguns picos, existe certa vantagem sobre a base de comparação. E conforme pontuado anteriormente se olhamos apenas para o uso de memória a medida mais vantajosa é a PMI global, porém levando em conta o tempo de execução a medida LLR local se mostra mais vantajosa.

5.4 DISCUSSÕES

Através dos resultados dos experimentos, podemos observar que é possível através da utilização de medidas de associatividade para calcular os pesos das arestas dos grafos de palavras, filtrar as arestas de menores pesos de acordo com uma certa porcentagem. Observamos que construindo os grafos com uma janela coocorrência de tamanho 12, independente a medida de associatividade utilizada para calcular os pesos é possível utilizar grafos de palavras com 20% menos arestas e ainda obter um resultado de classificação estatisticamente equivalente aos grafos de palavras com todas as arestas.

Entretanto, quando olhamos para o tempo de execução e utilização de memória, vemos que grafos construídos com tamanhos de janelas menores levam menos tempo para serem processados no node2vec e utilizam menos memória. Observa-se também que, na maioria dos resultados o tempo de execução utilizando grafos com pesos diminui, porém, apenas quando removemos 20% das arestas é que se observa uma diminuição vantajosa no uso de memória.

Observando todos os parâmetros analisados, é difícil dizer uma combinação que funcione bem para todas as bases de dados. Concluímos que a janela de coocorrência de tamanho 12 é a que, independente da medida de peso utilizada, não causa perda significativa na representação. No caso do percentual de filtragem de arestas, para que haja diminuição de tempo de execução e uso de memória, é necessário filtrar sempre 20% das arestas de menores pesos. Fixando esses valores para esses dois parâmetros, podemos observar então a medida de associatividade para

calcular os pesos. No caso da Polarity, a medida LLR global é a mais vantajosa. Já para o WebKB e o R8 a medida mais vantajosa é a PMI global. Já para o 20 Newsgroups, a medida PMI global é mais vantajosa se levamos em conta somente o uso de memória, porém levando em conta o tempo de execução a medida LLR local se mostra mais vantajosa.

Com esses experimentos concluímos que com as medidas de associatividade utilizadas é possível atribuir pesos às arestas dos grafos de palavras e remover até 20% das arestas de menores pesos sem causar perda estatisticamente significativa na representação. Porém, quando comparamos o tempo de execução e o uso de memória do algoritmo node2vec para gerar a representação dos grafos utilizando cada medida de associatividade para calcular os pesos, concluímos que somente quando removemos 20% das arestas é que existe uma vantagem observável.

5.5 CONSIDERAÇÕES FINAIS

Neste capítulo apresentamos os experimentos realizados de forma a avaliar a abordagem proposta bem como os resultados alcançados e suas análises. Observamos que em alguns casos específicos, é possível remover até 20% das arestas dos grafos de palavras e que na maioria das vezes há ganho no tempo de execução do algoritmo node2vec, porém, observamos também que esse ganho não está sempre presente no uso de memória.

O próximo capítulo apresenta a conclusão desse trabalho e as possibilidades de trabalhos futuros identificados.

6 CONCLUSÃO

O problema de classificação de texto tem o objetivo de atribuir de forma automática para um documento de texto uma classe dentre um conjunto de classes previamente definidas. Esse problema possui diversas aplicações e trata-se de um problema clássico e amplamente estudado na área de Processamento de Linguagem Natural. Uma tarefa muito importante na classificação de texto é a construção da representação, que deve prover informações relevantes ao classificador. Recentemente, modelos de representação de textos baseados em grafos vêm sendo muito utilizados, principalmente porque muitos algoritmos de aprendizagem que processam grafos como entrada têm sido desenvolvidos.

Motivada pelo crescente uso de grafos na representação de texto, esta pesquisa concentra-se em um modelo existente de representação de textos utilizando grafos conhecido como grafos de palavras, e propõe-se a trabalhar em uma representação mais concisa para esse modelo. Para isso, propomos um método de filtragem de arestas dos grafos de palavras em que os grafos com menos arestas possuem capacidade de representação estatisticamente equivalente a dos grafos de palavras originais. Nosso método de filtragem utiliza diversas medidas de associatividade de palavras para calcular e atribuir pesos às arestas dos grafos de palavras, então, esses pesos são ordenados e uma porcentagem fixa das arestas com os menores pesos são removidas.

Foram conduzidos experimentos com o intuito de validar a abordagem proposta. Para isso, primeiramente foi realizada a implementação de todo o processo, incluindo pré-processamento dos textos, geração dos grafos e filtragem das arestas, incorporação do algoritmo de aprendizagem de representação, neste trabalho o node2vec, e classificação utilizando uma CNN para texto. Depois, os experimentos foram executados com diversas variações de parâmetros.

Os resultados mostraram que quando utilizamos uma janela de coocorrência de tamanho doze para construir os grafos de palavras, é possível filtrar até 20% das arestas de menores pesos utilizando quaisquer das medidas de associatividade aplicadas para cálculo dos pesos. Analisando o tempo de execução do algoritmo node2vec para gerar a representação dos grafos com pesos e filtragem de arestas em comparação ao tempo de execução do mesmo processo para os grafos sem pesos com todas as arestas, observou-se na maioria dos casos há vantagens utilizando a abordagem proposta. Em contrapartida, quando compara-se o uso de memória, essa vantagem não está sempre presente.

Para trabalhos futuros na mesma abordagem identificamos as seguintes possibilidades:

- Desenvolvimento de um estudo avaliando, para cada medida de associatividade de palavras, quais arestas são filtradas e se há alguma relação entre as arestas removidas por cada medida de associatividade.
- Expansão da abordagem proposta utilizando outras medidas de associatividade.
- Aplicar um teste estatístico para avalizar se há diferença nos resultados comparando todas as diferentes medidas de associatividade entre si.
- Avaliar o impacto de outros percentuais de corte.
- Utilizar outros algoritmos de aprendizagem de representação e comparar os resultados obtidos com os resultados obtidos utilizando node2vec.

REFERÊNCIAS

Abu-El-Haija, S., Kapoor, A., Perozzi, B. e Lee, J. (2018). N-gcn: Multi-scale graph convolution for semi-supervised node classification. *arXiv preprint arXiv:1802.08888*.

Aggarwal, C. C. (2018). *Machine learning for text*. Springer.

Bijari, K., Zare, H., Kebriaei, E. e Veisi, H. (2020). Leveraging deep graph-based text representation for sentiment polarity applications. *Expert Systems with Applications*, 144:113090.

Blanco, R. e Lioma, C. (2012). Graph-based term weighting for information retrieval. *Information retrieval*, 15(1):54–92.

Bondy, J. A., Murty, U. S. R. et al. (1976). *Graph theory with applications*, volume 290. Macmillan London.

Cambria, E. e White, B. (2014). Jumping nlp curves: A review of natural language processing research. *IEEE Computational intelligence magazine*, 9(2):48–57.

Chaudhari, D. L., Damani, O. P. e Laxman, S. (2011). Lexical co-occurrence, statistical significance, and word association. Em *Proceedings of the conference on empirical methods in natural language processing*, páginas 1058–1068. Association for Computational Linguistics.

Chen, F., Wang, Y.-C., Wang, B. e Kuo, C.-C. J. (2020). Graph representation learning: a survey. *APSIPA Transactions on Signal and Information Processing*, 9.

Chowdhury, G. G. (2003). Natural language processing. *Annual review of information science and technology*, 37(1):51–89.

Church, K. W. e Hanks, P. (1990). Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K. e Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(Aug):2493–2537.

Cormen, T. H., Leiserson, C. E., Rivest, R. L. e Stein, C. (2009). *Introduction to algorithms*. MIT press.

Damani, O. (2013). Improving pointwise mutual information (PMI) by incorporating significant co-occurrence. Em *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, páginas 20–28, Sofia, Bulgaria. Association for Computational Linguistics.

Damani, O. P. e Ghonge, S. (2013). Appropriately incorporating statistical significance in pmi. Em *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, páginas 163–169.

Defferrard, M., Bresson, X. e Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. Em *Advances in neural information processing systems*, páginas 3844–3852.

Dunning, T. (1993). Accurate methods for the statistics of surprise and coincidence. *Computational linguistics*, 19(1):61–74.

Gaigole, P. C., Patil, L. e Chaudhari, P. (2013). Preprocessing techniques in text categorization. Em *National Conference on Innovative Paradigms in Engineering & Technology (NVIPET-2013), Proceedings published by International Journal of Computer Applications (IJCA)*.

Ganegedara, T. (2018). *Natural Language Processing with TensorFlow: Teach language to machines using Python's deep learning library*. Packt Publishing Ltd.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O. e Dahl, G. E. (2017). Neural message passing for quantum chemistry. Em *International Conference on Machine Learning*, páginas 1263–1272.

Godec, P. (2018). Graph embeddings — the summary. *Towards Data Science*.

Goutte, C. e Gaussier, E. (2005). A probabilistic interpretation of precision, recall and f-score, with implication for evaluation. Em *European conference on information retrieval*, páginas 345–359. Springer.

Grover, A. e Leskovec, J. (2016). node2vec: Scalable feature learning for networks. Em *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, páginas 855–864. ACM.

Hamilton, W. L., Ying, R. e Leskovec, J. (2017). Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*.

Huang, L., Ma, D., Li, S., Zhang, X. e Wang, H. (2019). Text level graph neural network for text classification. Em *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, páginas 3444–3450, Hong Kong, China. Association for Computational Linguistics.

Ikonomakis, M., Kotsiantis, S. e Tampakas, V. (2005). Text classification using machine learning techniques. *WSEAS transactions on computers*, 4(8):966–974.

Kim, Y. (2014). Convolutional neural networks for sentence classification. Em *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, páginas 1746–1751.

Kipf, T. N. e Welling, M. (2017). Semi-supervised classification with graph convolutional networks. Em *International Conference on Learning Representations (ICLR)*.

Kleinberg, J. e Tardos, E. (2006). *Algorithm design*. Pearson Education India.

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. Em *Ijcai*, volume 14, páginas 1137–1145. Montreal, Canada.

Kriege, N. M., Johansson, F. D. e Morris, C. (2019). A survey on graph kernels. *arXiv preprint arXiv:1903.11835*.

LeCun, Y., Bengio, Y. e Hinton, G. (2015). Deep learning. *nature*, 521(7553):436.

Liu, Y., Liu, Z., Chua, T.-S. e Sun, M. (2015). Topical word embeddings. Em *Twenty-ninth AAAI conference on artificial intelligence*. Citeseer.

Manning, C. e Schutze, H. (1999). *Foundations of statistical natural language processing*. MIT press.

McCormick, C. (2016). Word2vec tutorial - the skip-gram model.

McHugh, J. A. (1990). *Algorithmic graph theory*, volume 68056. Prentice Hall Englewood Cliffs, NJ.

Mikolov, T., Chen, K., Corrado, G. e Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S. e Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. Em *Advances in neural information processing systems*, páginas 3111–3119.

Pearson, K. (1900). X. on the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 50(302):157–175.

Peng, H., Li, J., He, Y., Liu, Y., Bao, M., Wang, L., Song, Y. e Yang, Q. (2018). Large-scale hierarchical text classification with recursively regularized deep graph-cnn. Em *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, páginas 1063–1072. International World Wide Web Conferences Steering Committee.

Pennington, J., Socher, R. e Manning, C. D. (2014). Glove: Global vectors for word representation. Em *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, páginas 1532–1543.

Perozzi, B., Al-Rfou, R. e Skiena, S. (2014). Deepwalk: Online learning of social representations. Em *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, páginas 701–710. ACM.

Ramos, J. et al. (2003). Using tf-idf to determine word relevance in document queries. Em *Proceedings of the first instructional conference on machine learning*, volume 242, páginas 133–142.

Rosen, K. H. (1999). *Discrete Mathematics & Applications*. McGraw-Hill.

Rousseau, F., Kiagias, E. e Vazirgiannis, M. (2015). Text categorization as a graph classification problem. Em *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, páginas 1702–1712.

Rousseau, F. e Vazirgiannis, M. (2015). Main core retention on graph-of-words for single-document keyword extraction. Em *European Conference on Information Retrieval*, páginas 382–393. Springer.

Russell, S. J. e Norvig, P. (2016). *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47.

Shanavas, N., Wang, H., Lin, Z. e Hawe, G. (2019). Structure-based supervised term weighting and regularization for text classification. Em *International Conference on Applications of Natural Language to Information Systems*, páginas 105–117. Springer.

Skianis, K. (2019). *Novel Representations, Regularization & Distances for Text Classification*. Tese de doutorado, Ecole Polytechnique, Paris-Saclay University.

Skianis, K., Malliaros, F. e Vazirgiannis, M. (2018). Fusing document, collection and label graph-based representations with word embeddings for text classification. Em *NAACL-HLT Workshop on Graph-Based Natural Language Processing (TextGraphs)*.

Srividhya, V. e Anitha, R. (2010). Evaluating preprocessing techniques in text categorization. *International journal of computer science and application*, 47(11):49–51.

Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J. e Mei, Q. (2015). Line: Large-scale information network embedding. Em *Proceedings of the 24th international conference on world wide web*, páginas 1067–1077. International World Wide Web Conferences Steering Committee.

Uysal, A. K. e Gunal, S. (2014). The impact of preprocessing on text classification. *Information Processing & Management*, 50(1):104–112.

Vazirani, U., Papadimitriou, C. e Dasgupta, S. (2018). Algorithms.

Yan, X. e Han, J. (2002). gspan: Graph-based substructure pattern mining. Em *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, páginas 721–724. IEEE.

Yao, L., Mao, C. e Luo, Y. (2019). Graph convolutional networks for text classification. Em *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, páginas 7370–7377.

Yu, J., Lu, Y., Qin, Z., Zhang, W., Liu, Y., Tan, J. e Guo, L. (2018). Modeling text with graph convolutional network for cross-modal information retrieval. Em Hong, R., Cheng, W.-H., Yamasaki, T., Wang, M. e Ngo, C.-W., editores, *Advances in Multimedia Information Processing – PCM 2018*, Cham. Springer International Publishing.

Zhang, X., Zhao, J. e LeCun, Y. (2015). Character-level convolutional networks for text classification. Em *Advances in neural information processing systems*, páginas 649–657.

APÊNDICE A – GRÁFICOS COM OS RESULTADOS DA CLASSIFICAÇÃO

Neste apêndice apresentamos os gráficos com os resultados da classificação de cada *dataset* utilizando a abordagem proposta. Cada gráfico apresenta, para a base de comparação, que são os grafos sem pesos nas arestas, e para cada medida de associatividade utilizada como peso das arestas:

- a média da taxa de acerto das 10 partições com desvio padrão;
- e a média do F1-score das 10 partições com desvio padrão.

Apresentamos um gráfico por tamanho de janela e porcentagem de corte. Cada subseção apresenta os resultados para uma base de dados.

A.1 BASE DE DADOS: POLARITY

A.1.1 Polarity - Janela de coocorrência de tamanho 4

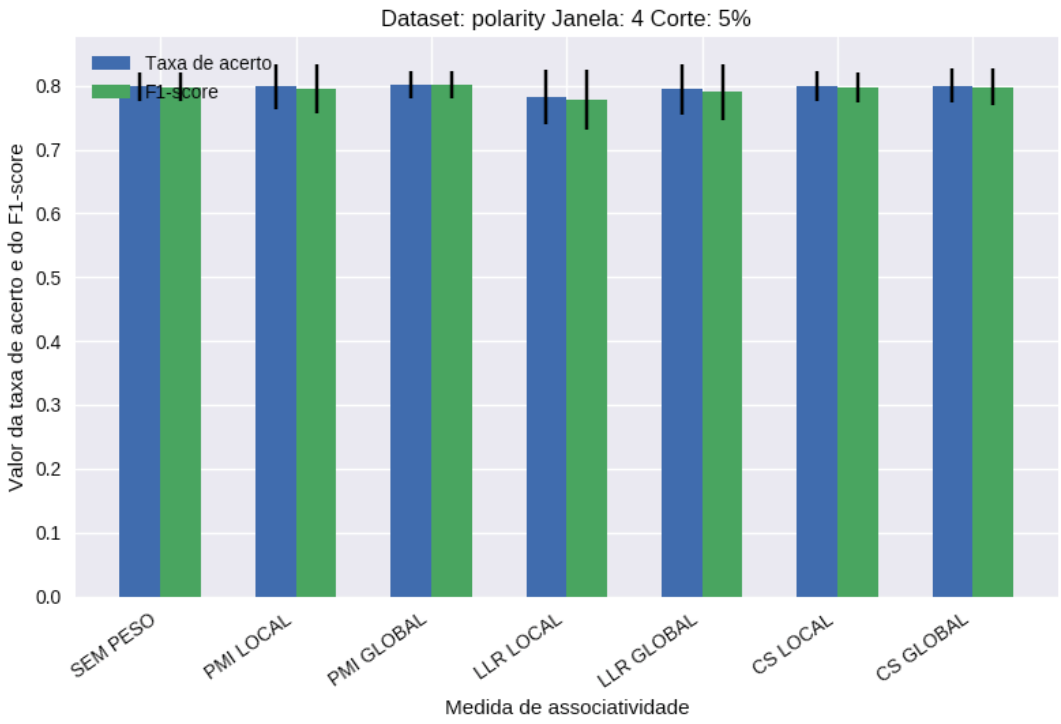


Figura A.1: Polarity - Janela de coocorrência 4, filtragem de 5% das arestas

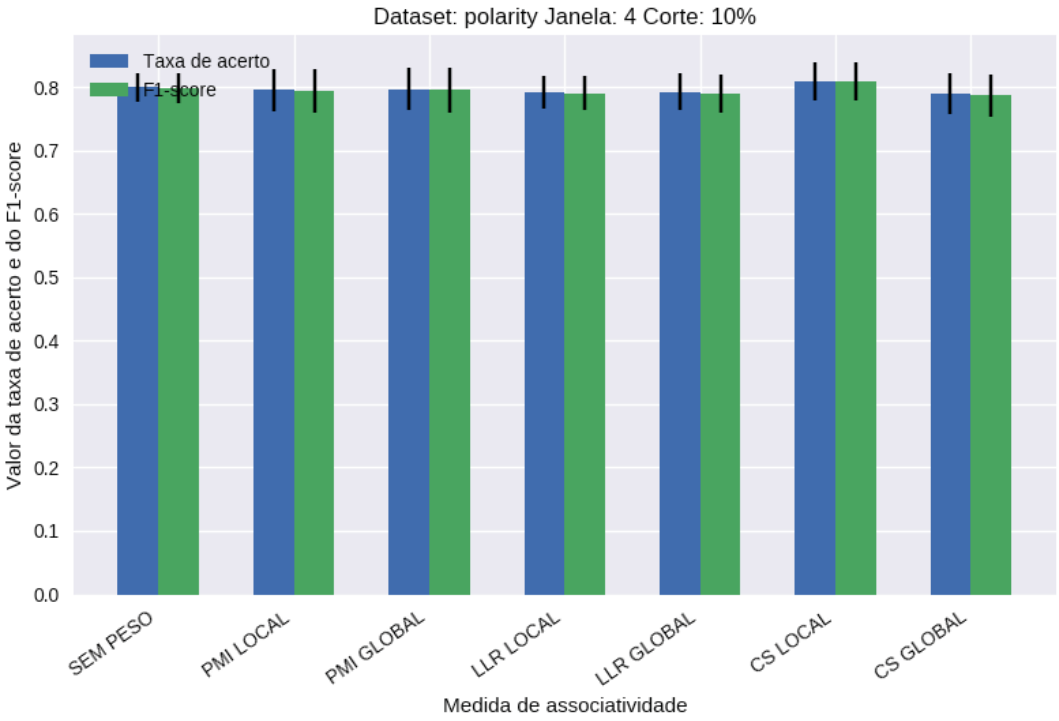


Figura A.2: Polarity - Janela de coocorrência 4, filtragem de 10% das arestas

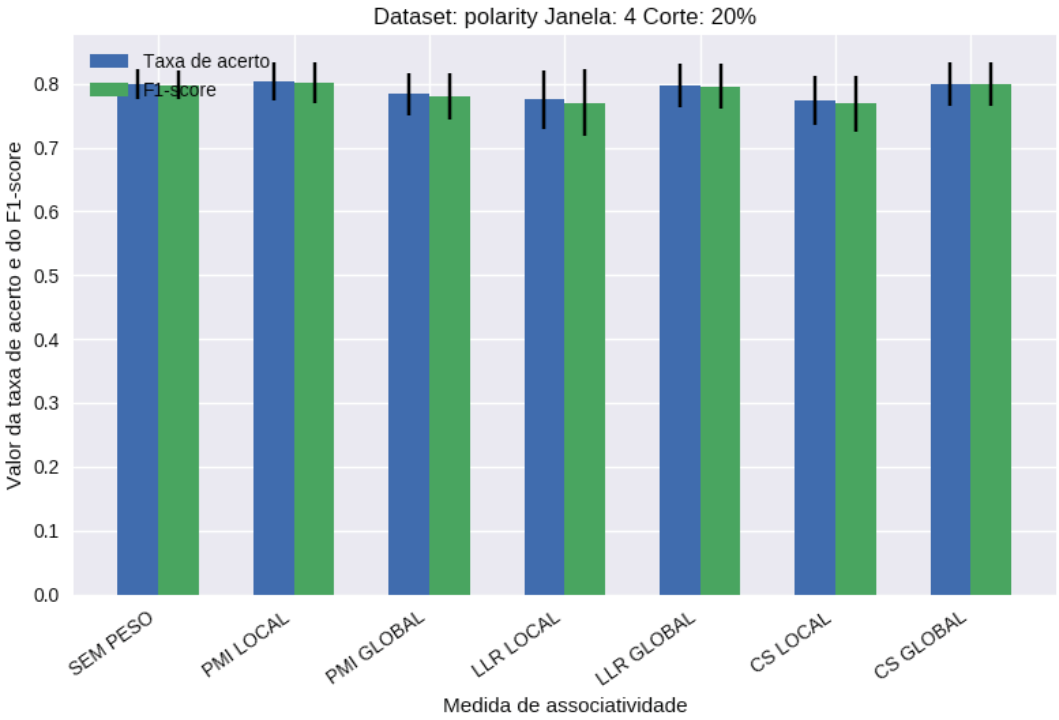


Figura A.3: Polarity - Janela de coocorrência 4, filtragem de 20% das arestas

A.1.2 Polarity - Janela de coocorrência de tamanho 12

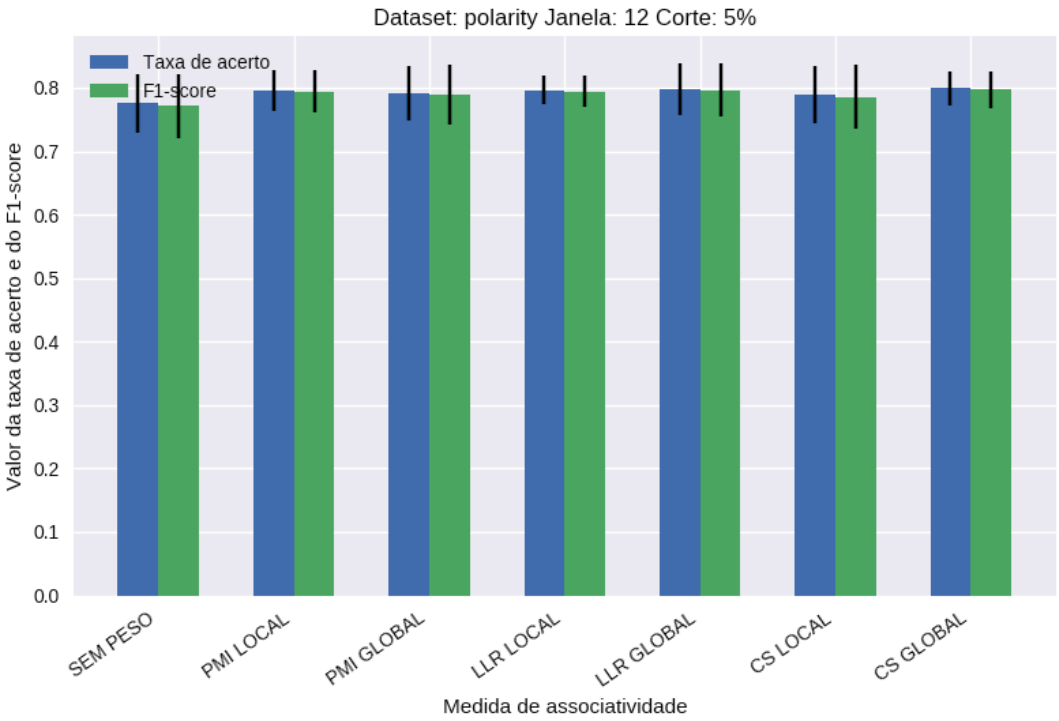


Figura A.4: Polarity - Janela de coocorrência 12, filtragem de 5% das arestas

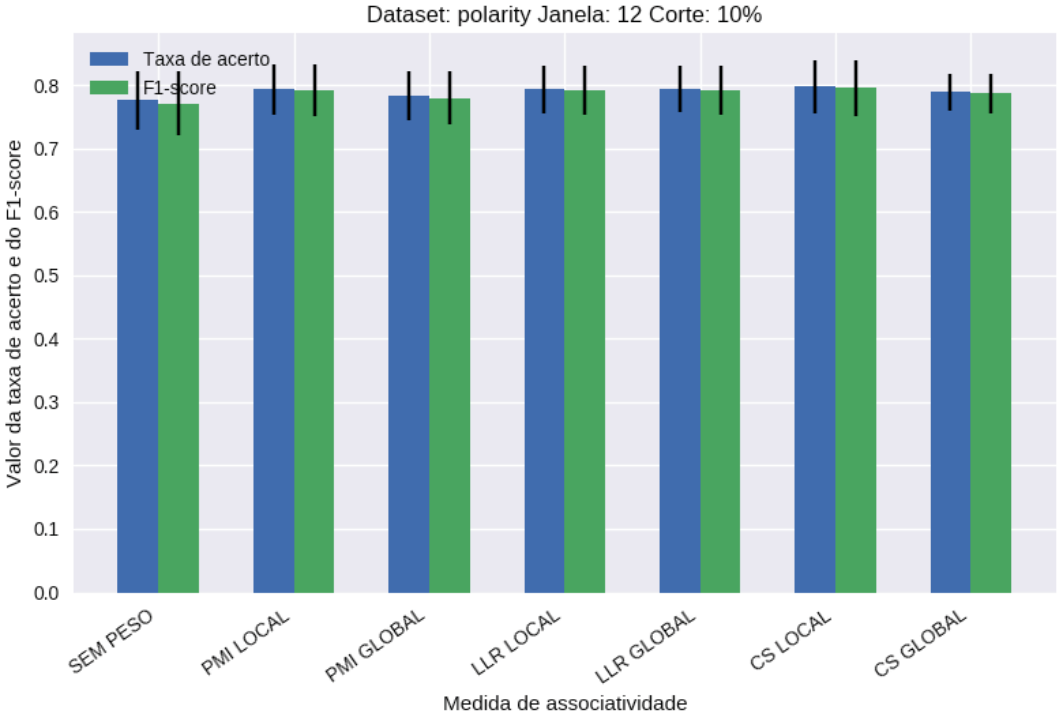


Figura A.5: Polarity - Janela de coocorrência 12, filtragem de 10% das arestas

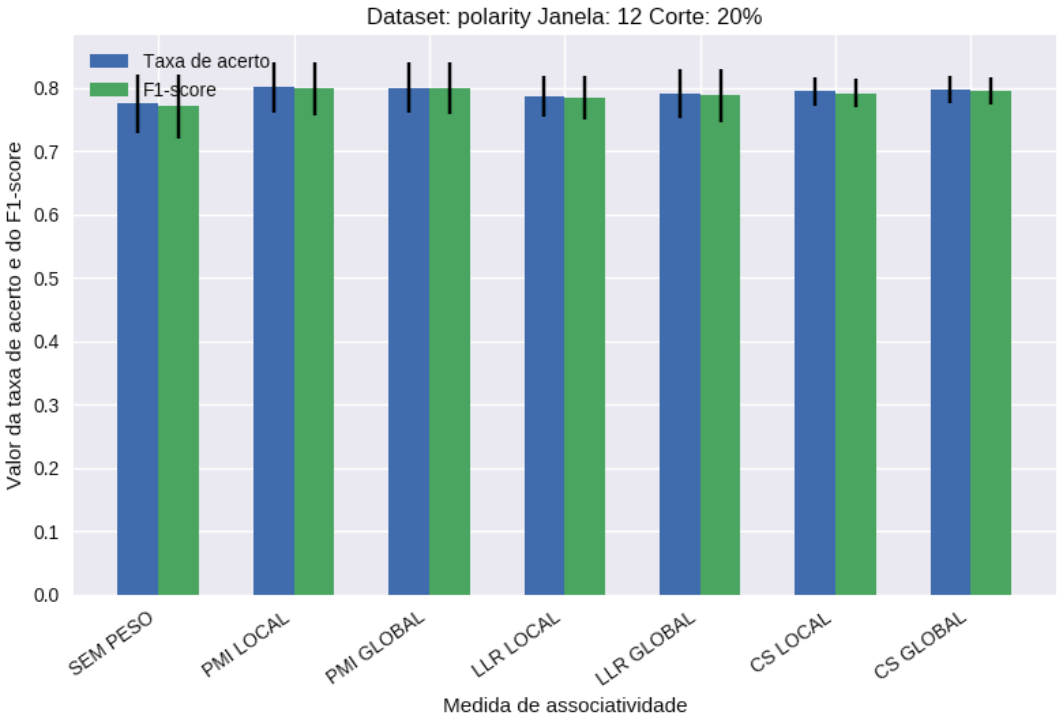


Figura A.6: Polarity - Janela de coocorrência 12, filtragem de 20% das arestas

A.1.3 Polarity - Janela de coocorrência de tamanho 20

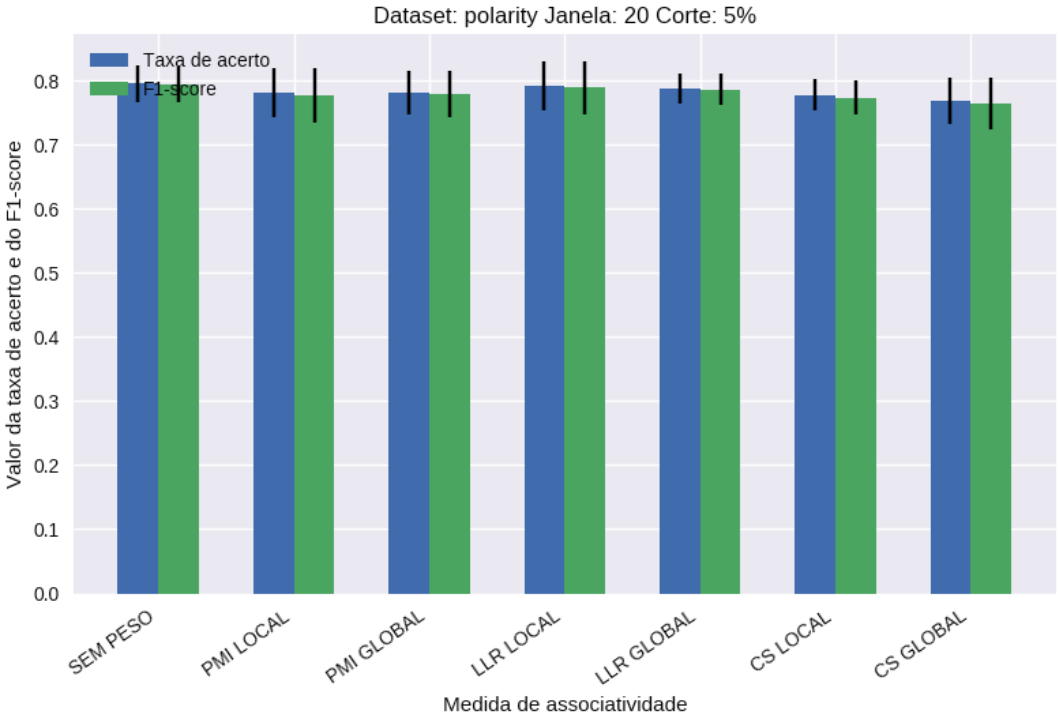


Figura A.7: Polarity - Janela de coocorrência 20, filtragem de 5% das arestas

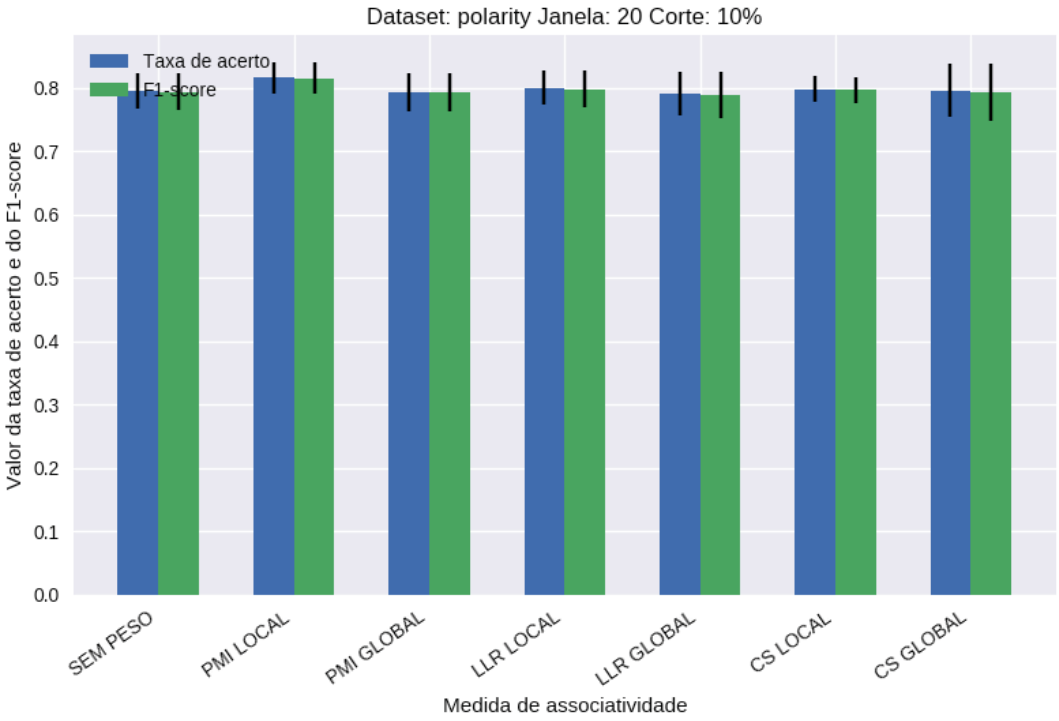


Figura A.8: Polarity - Janela de coocorrência 20, filtragem de 10% das arestas

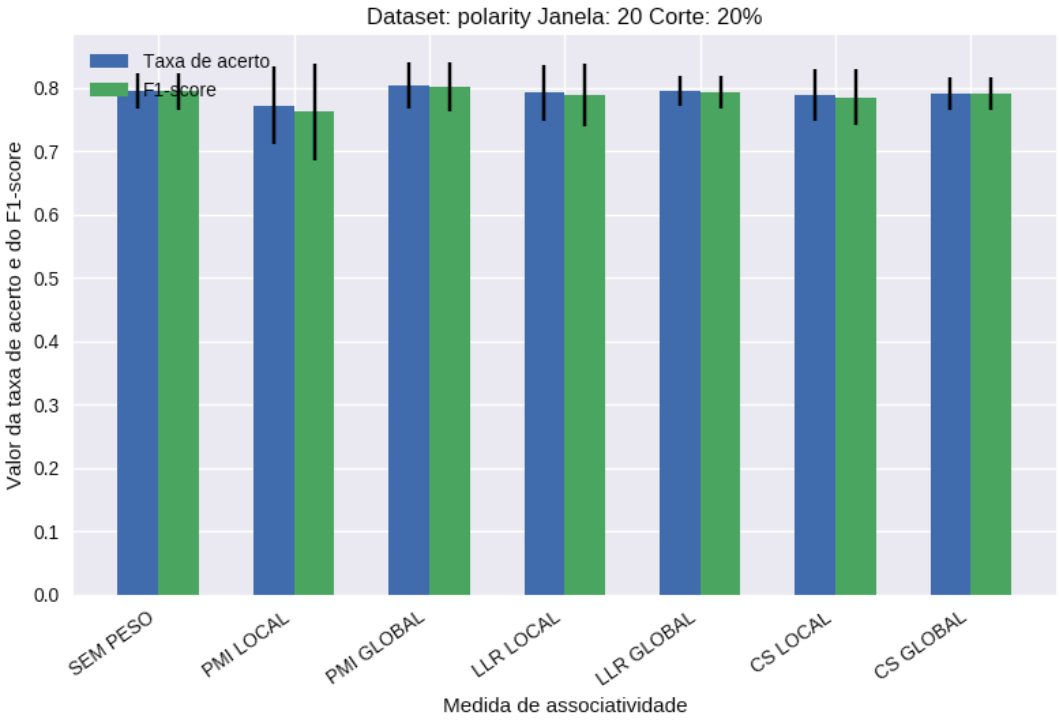


Figura A.9: Polarity -Janela de coocorrência 20, filtragem de 20% das arestas

A.2 BASE DE DADOS: WEBKB

A.2.1 WebKB - Janela de coocorrência de tamanho 4

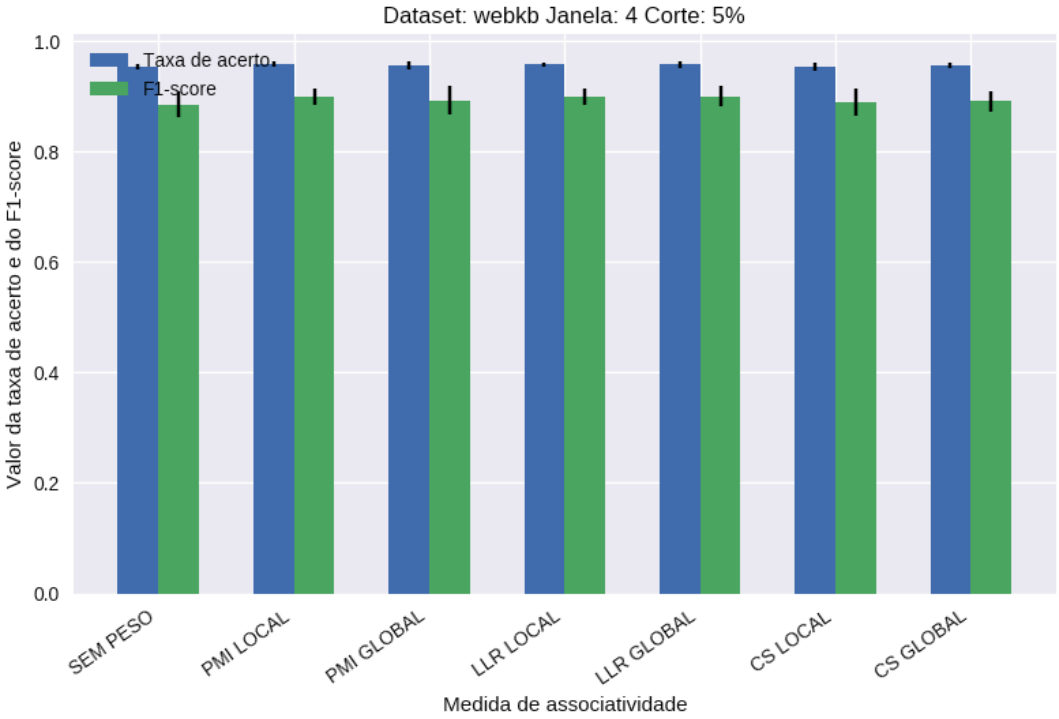


Figura A.10: WebKB - Janela de coocorrência 4, filtragem de 5% das arestas

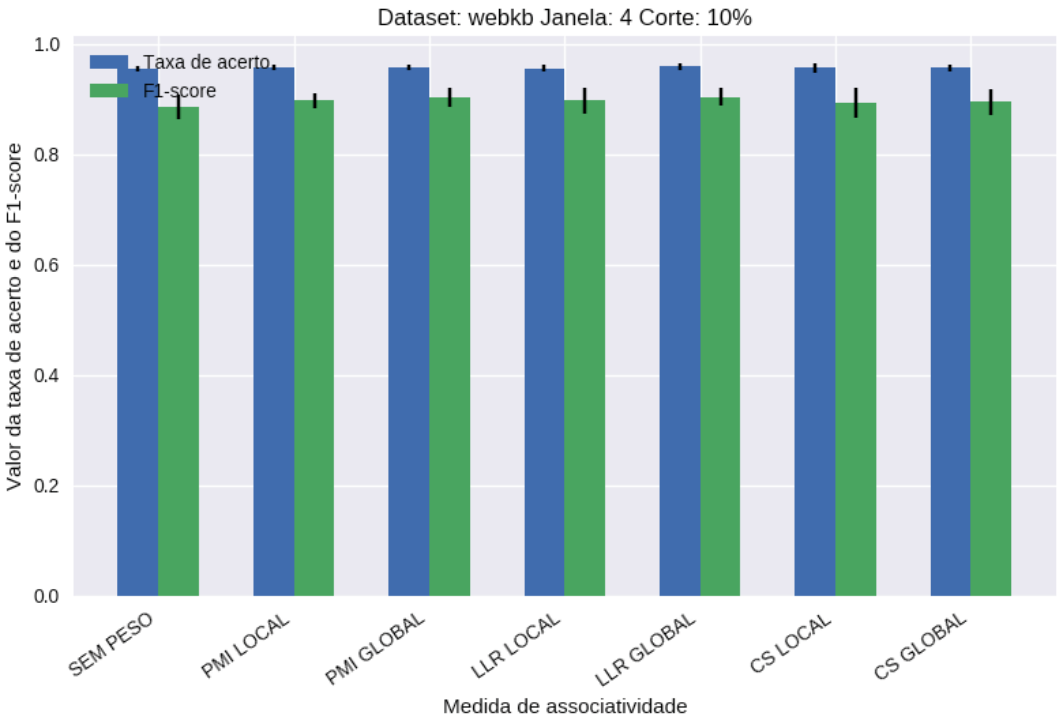


Figura A.11: WebKB - Janela de coocorrência 4, filtragem de 10% das arestas

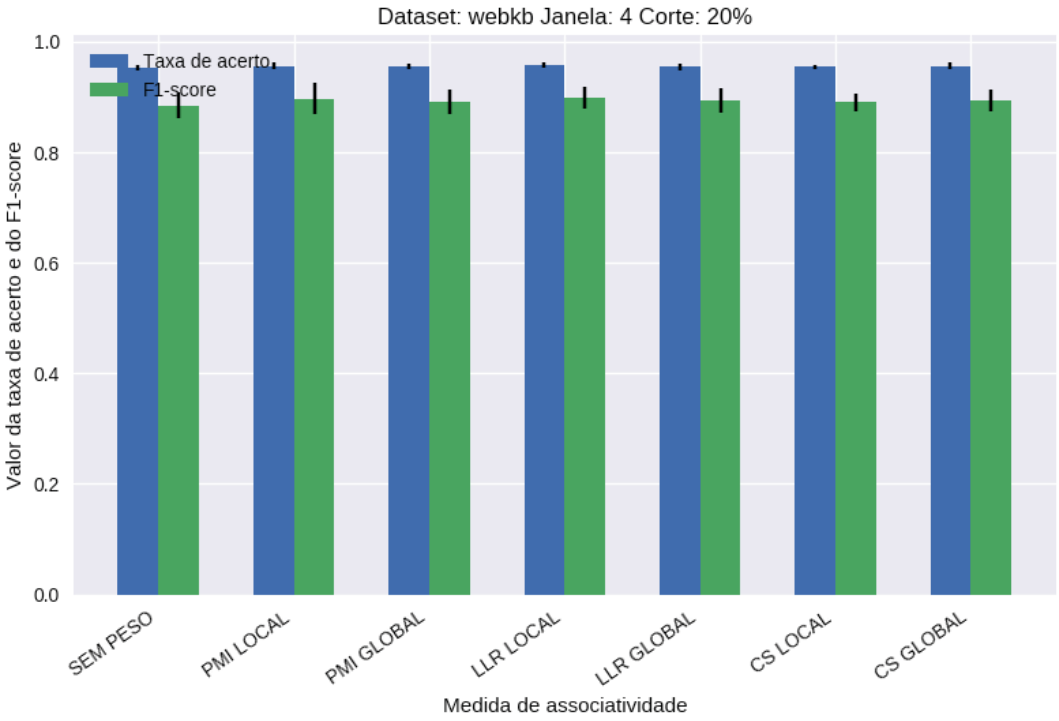


Figura A.12: WebKB - Janela de coocorrência 4, filtragem de 20% das arestas

A.2.2 WebKB - Janela de coocorrência de tamanho 12

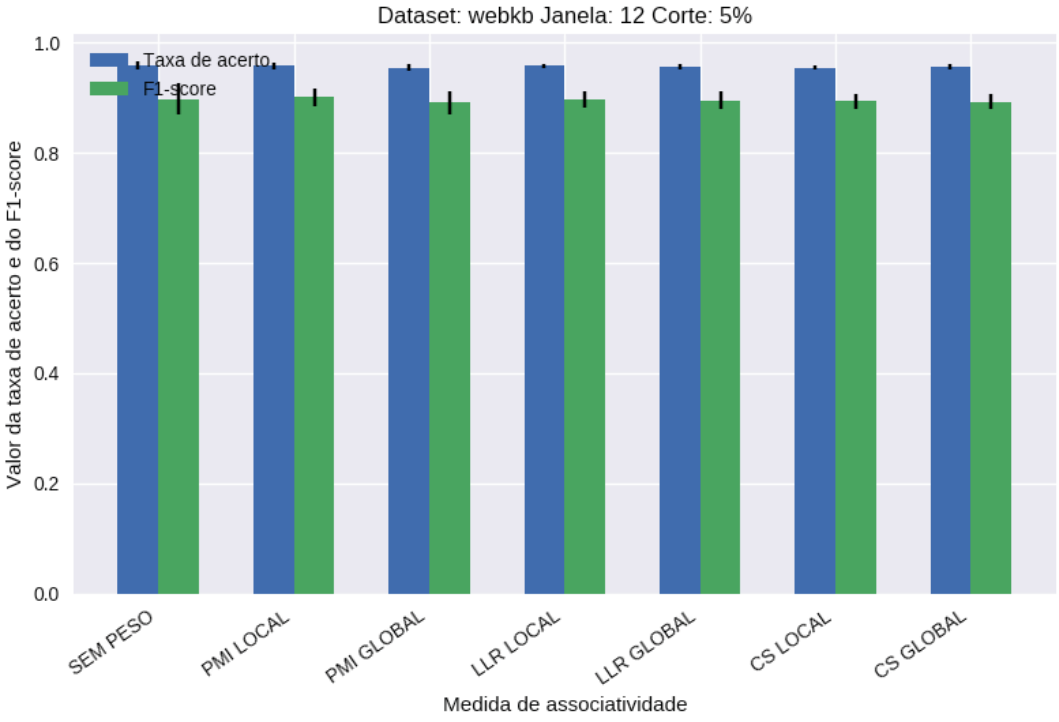


Figura A.13: WebKB - Janela de coocorrência 12, filtragem de 5% das arestas

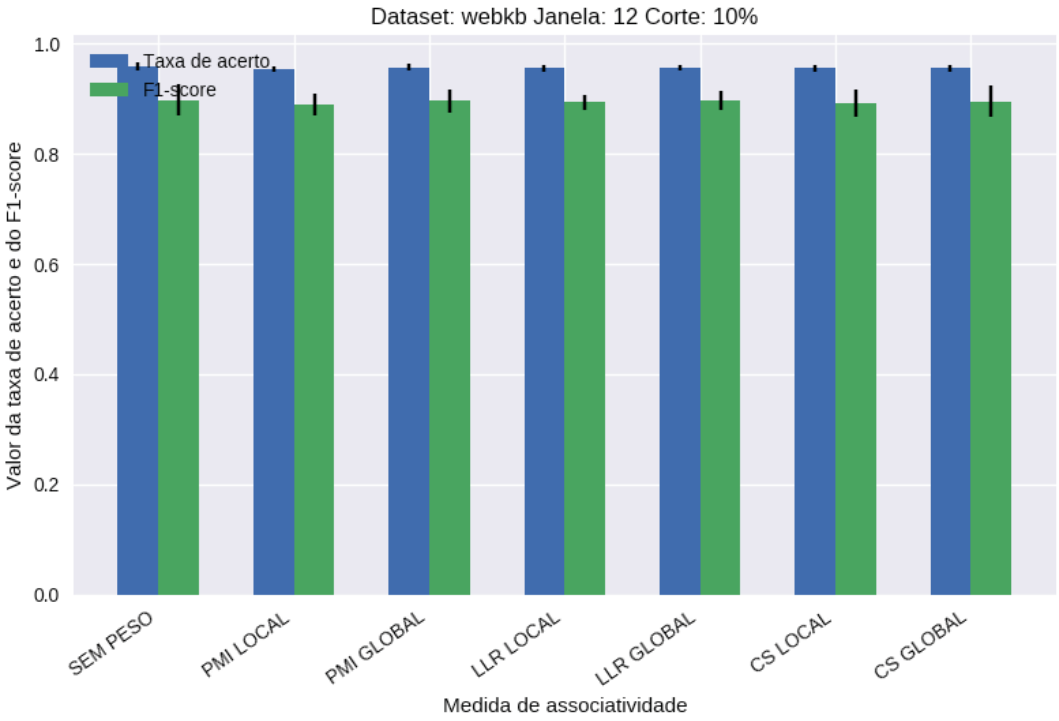


Figura A.14: WebKB - Janela de coocorrência 12, filtragem de 10% das arestas

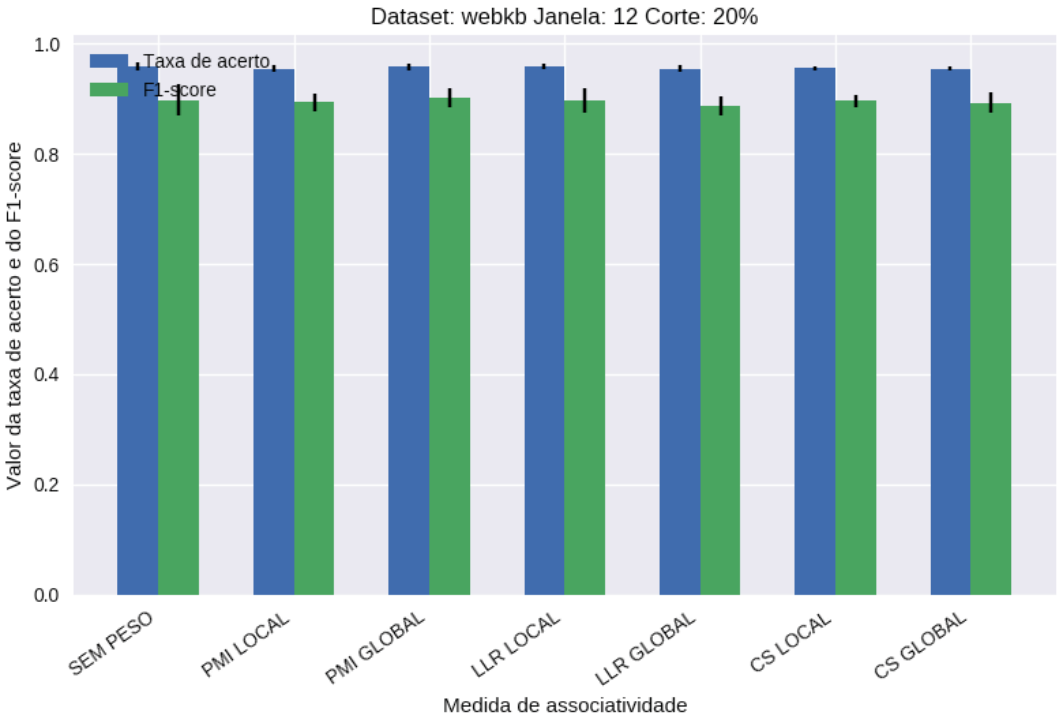


Figura A.15: WebKB - Janela de coocorrência 12, filtragem de 20% das arestas

A.2.3 WebKB - Janela de coocorrência de tamanho 20

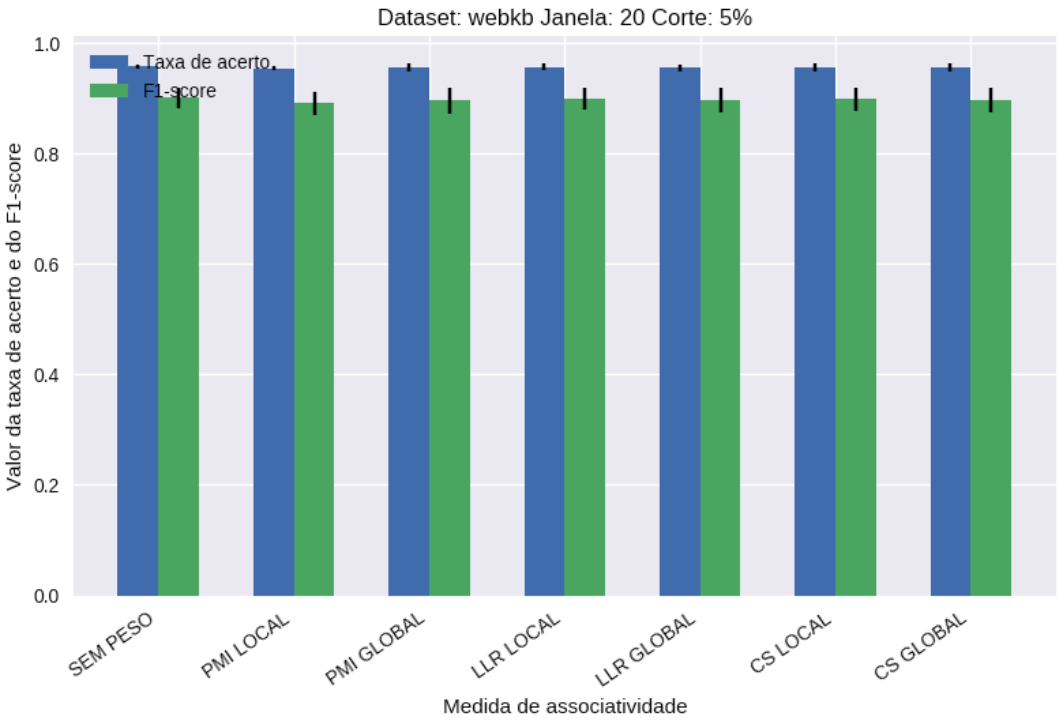


Figura A.16: WebKB - Janela de coocorrência 20, filtragem de 5% das arestas

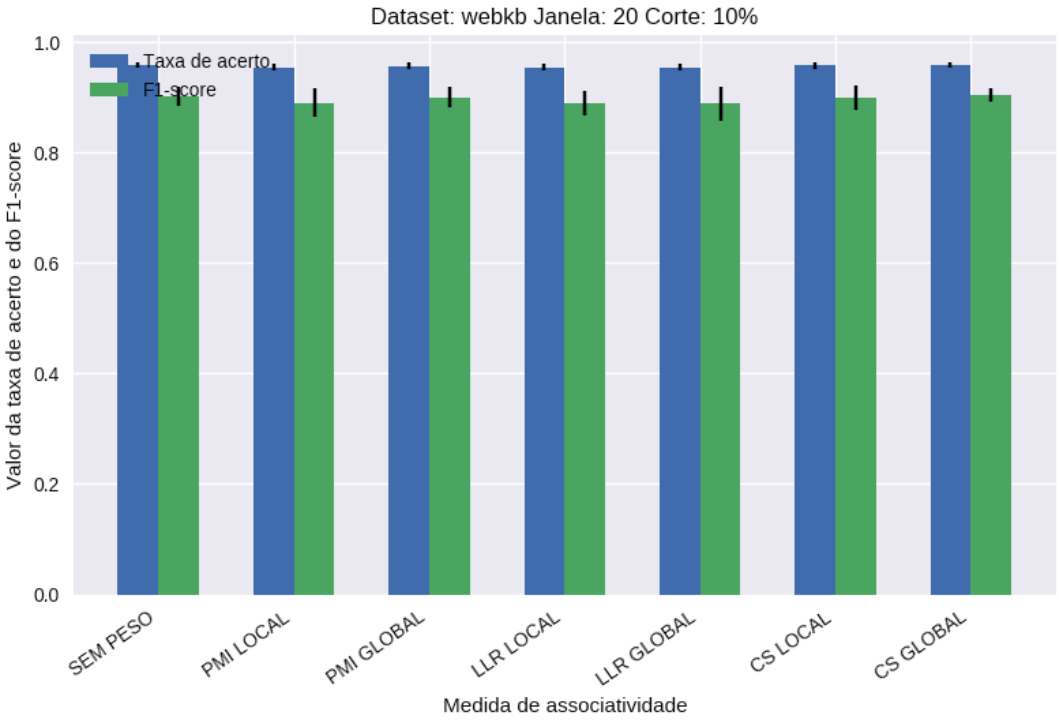


Figura A.17: WebKB - Janela de coocorrência 20, filtragem de 10% das arestas

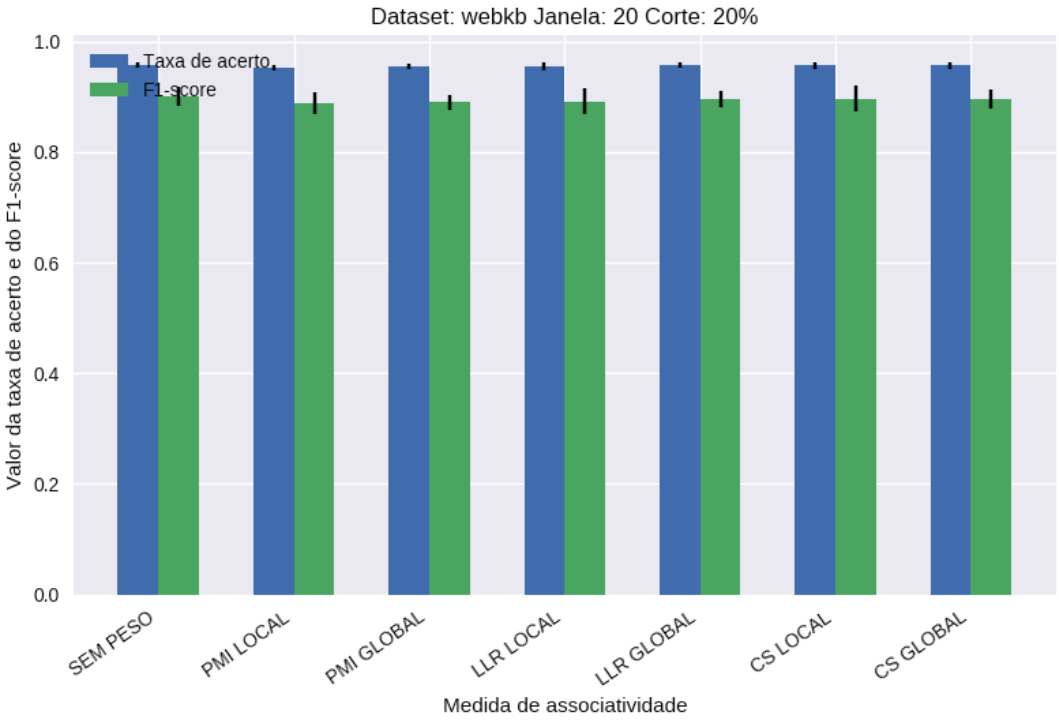


Figura A.18: WebKB - Janela de coocorrência 20, filtragem de 20% das arestas

A.3 BASE DE DADOS: R8

A.3.1 R8 - Janela de coocorrência de tamanho 4

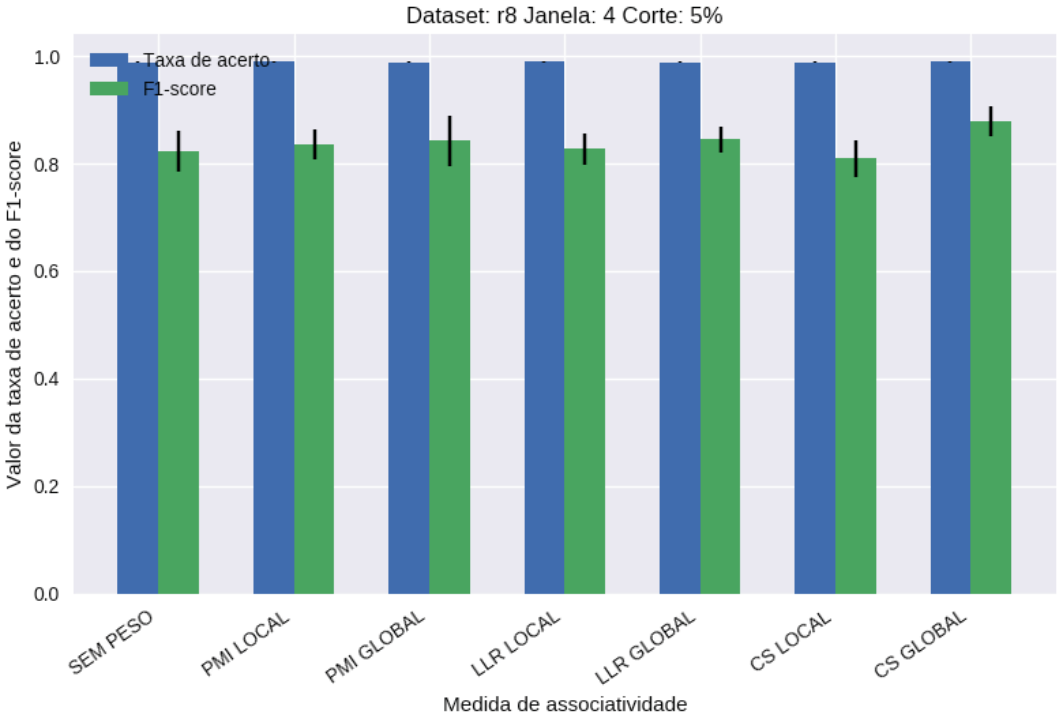


Figura A.19: R8 - Janela de coocorrência 4, filtragem de 5% das arestas

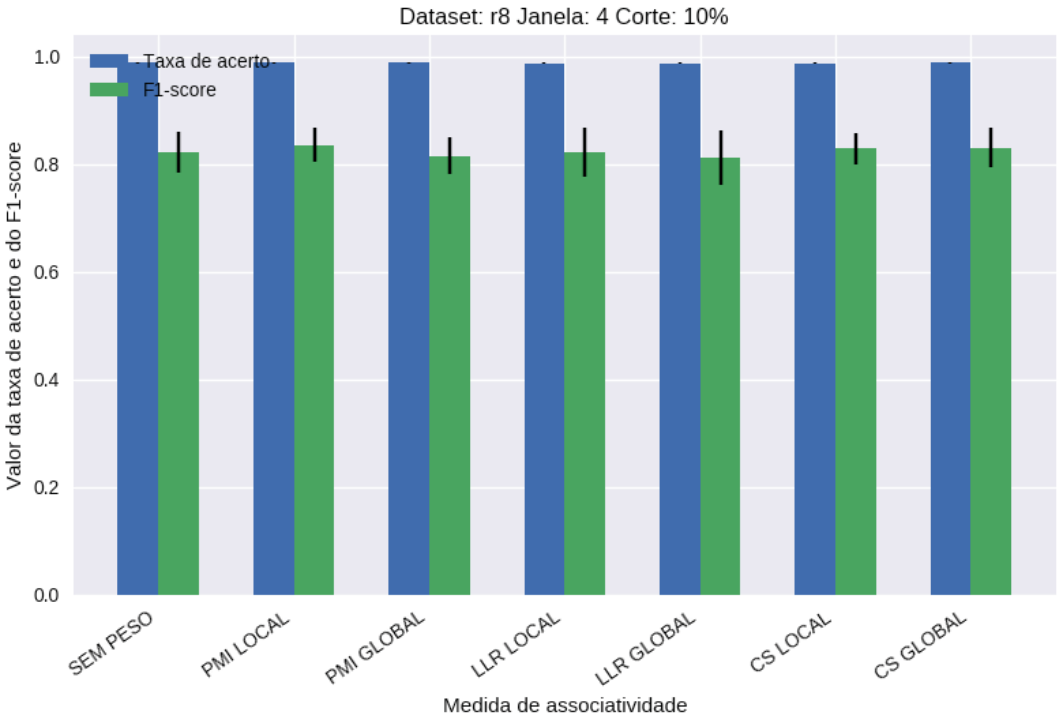


Figura A.20: R8 - Janela de coocorrência 4, filtragem de 10% das arestas

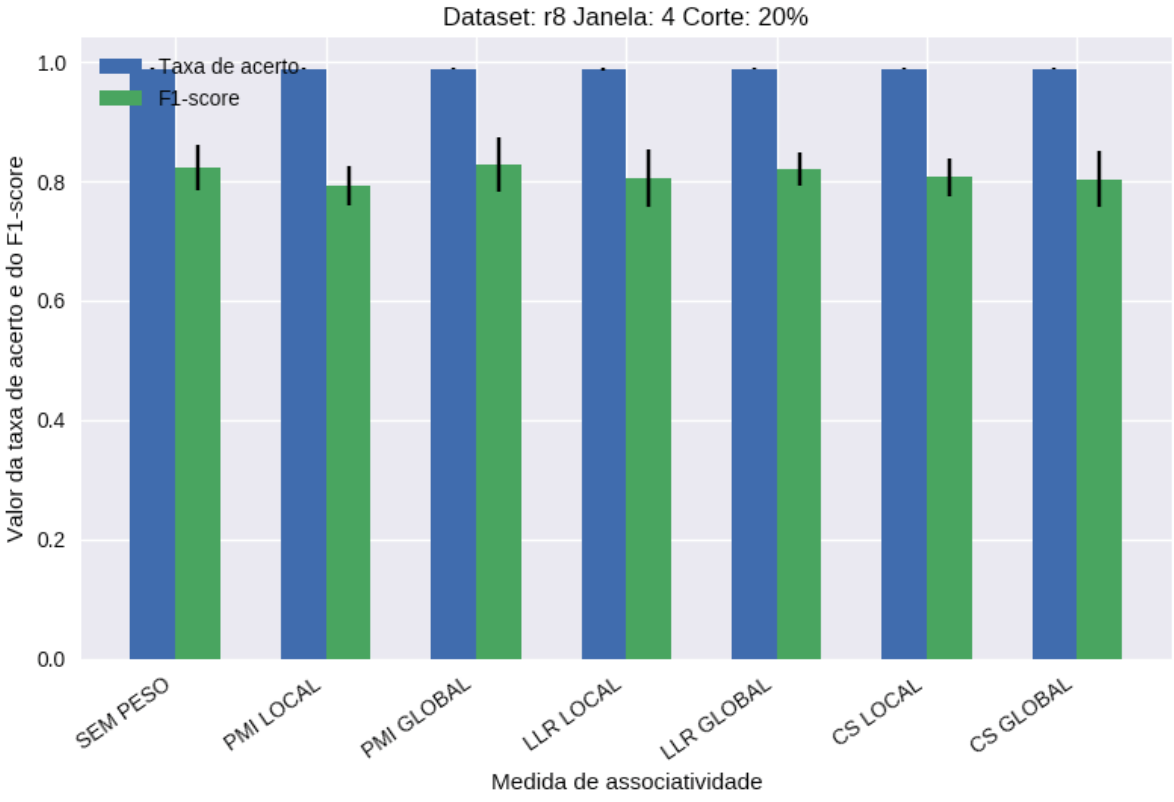


Figura A.21: R8 - Janela de coocorrência 4, filtragem de 20% das arestas

A.3.2 R8 - Janela de coocorrência de tamanho 12

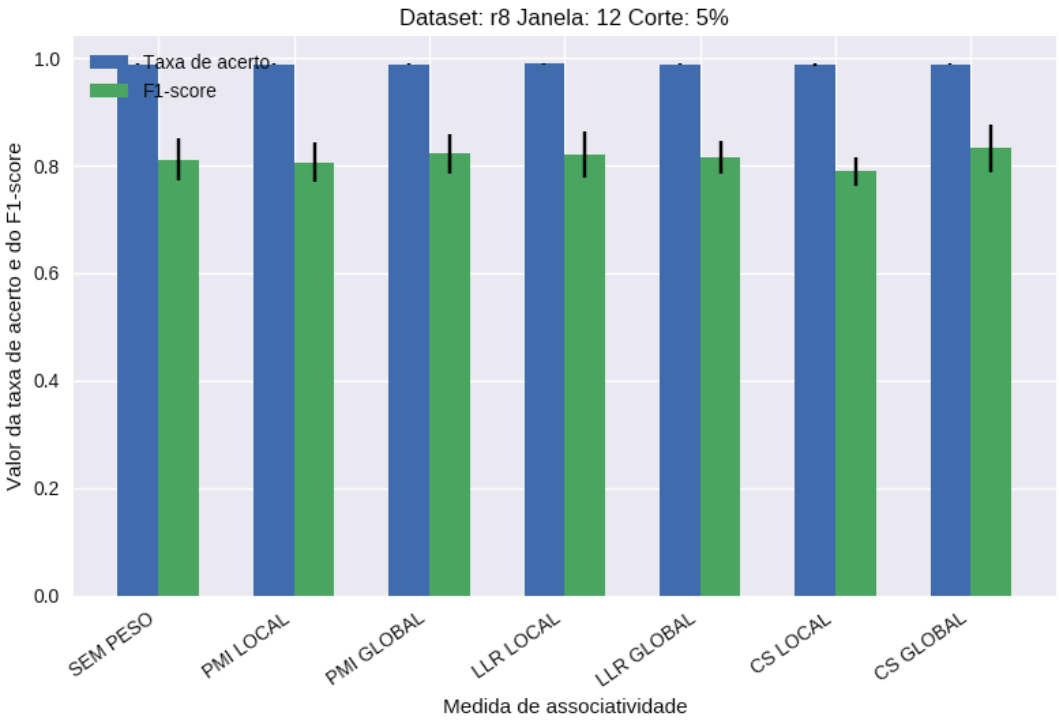


Figura A.22: R8 - Janela de coocorrência 12, filtragem de 5% das arestas

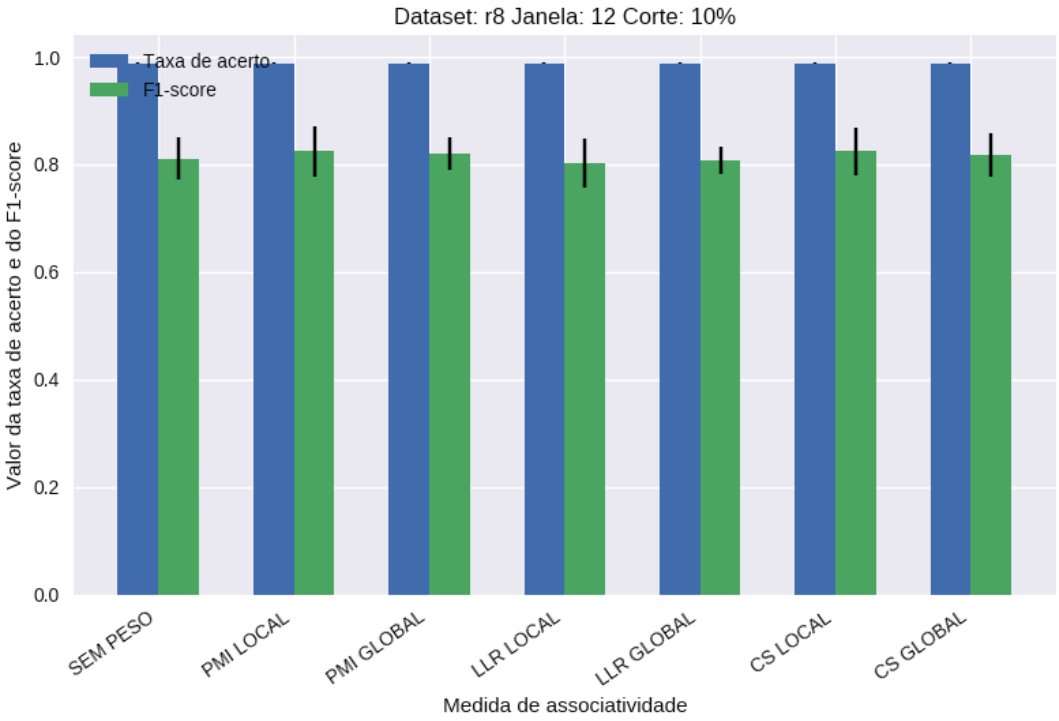


Figura A.23: R8 - Janela de coocorrência 12, filtragem de 10% das arestas

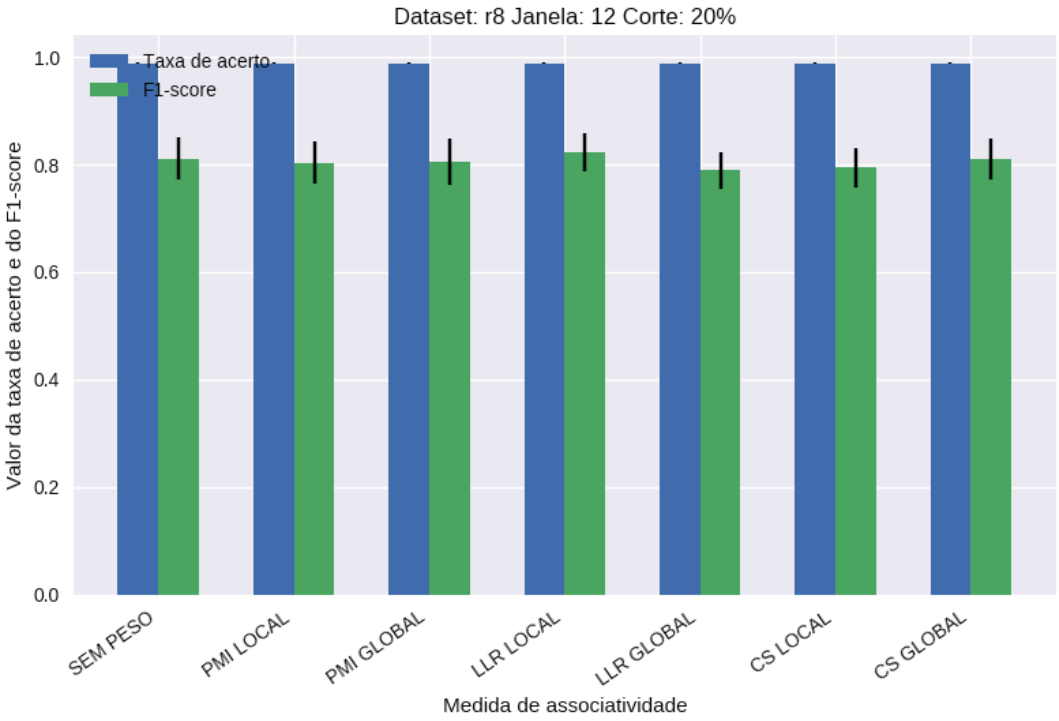


Figura A.24: R8 - Janela de coocorrência 12, filtragem de 20% das arestas

A.3.3 R8 - Janela de coocorrência de tamanho 20

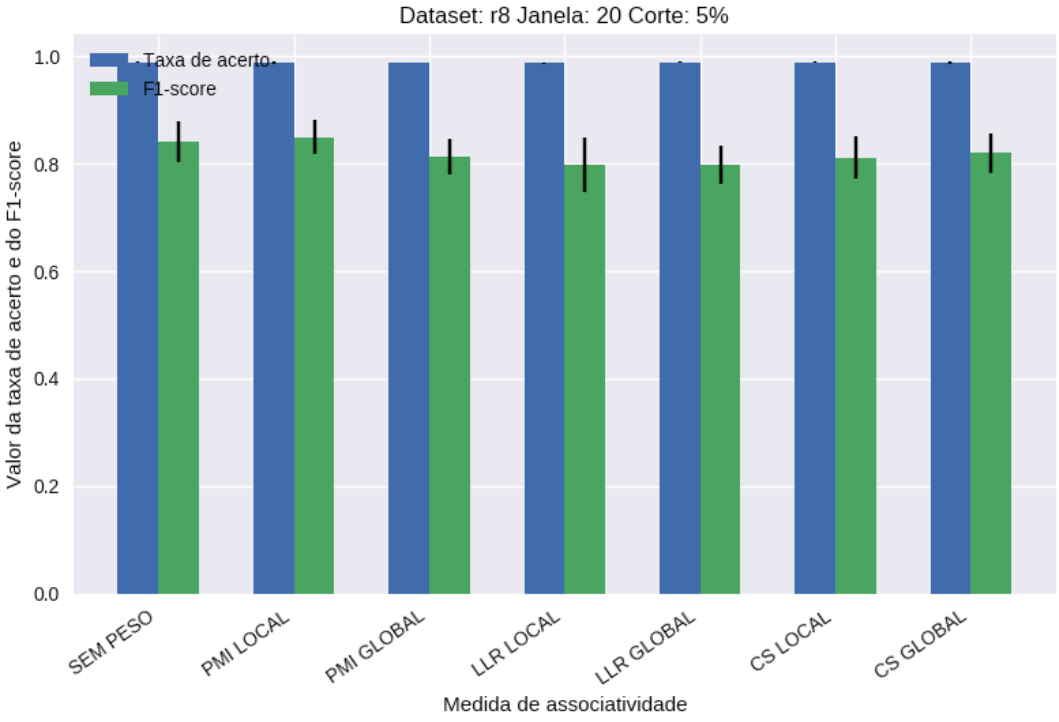


Figura A.25: R8 - Janela de coocorrência 20, filtragem de 5% das arestas

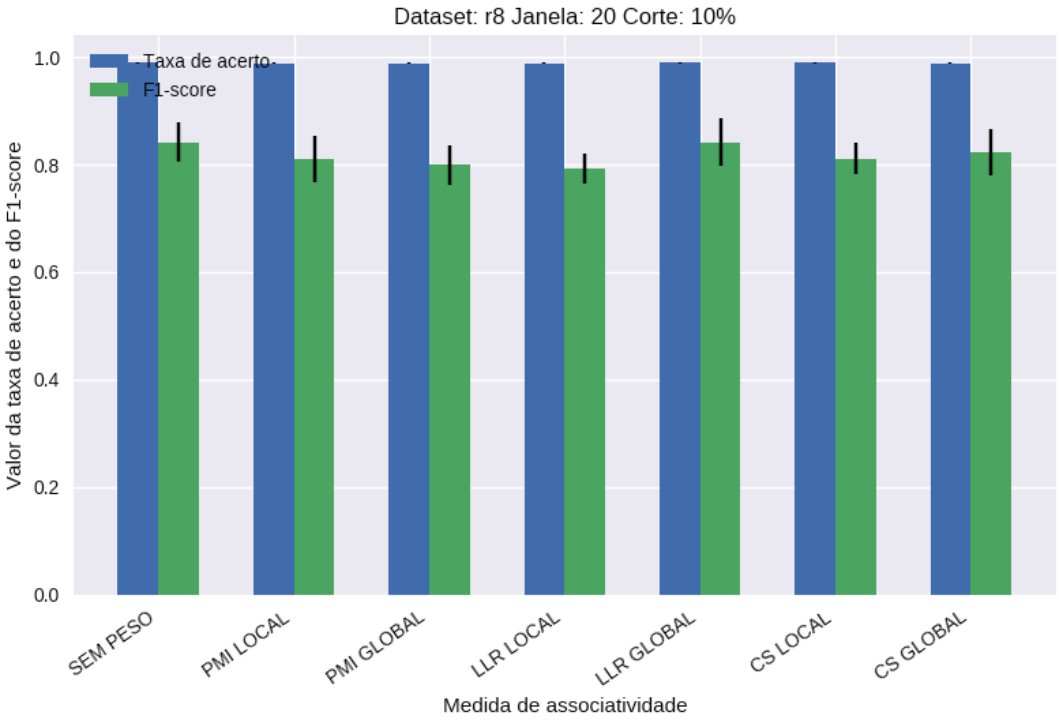


Figura A.26: R8 - Janela de coocorrência 20, filtragem de 10% das arestas

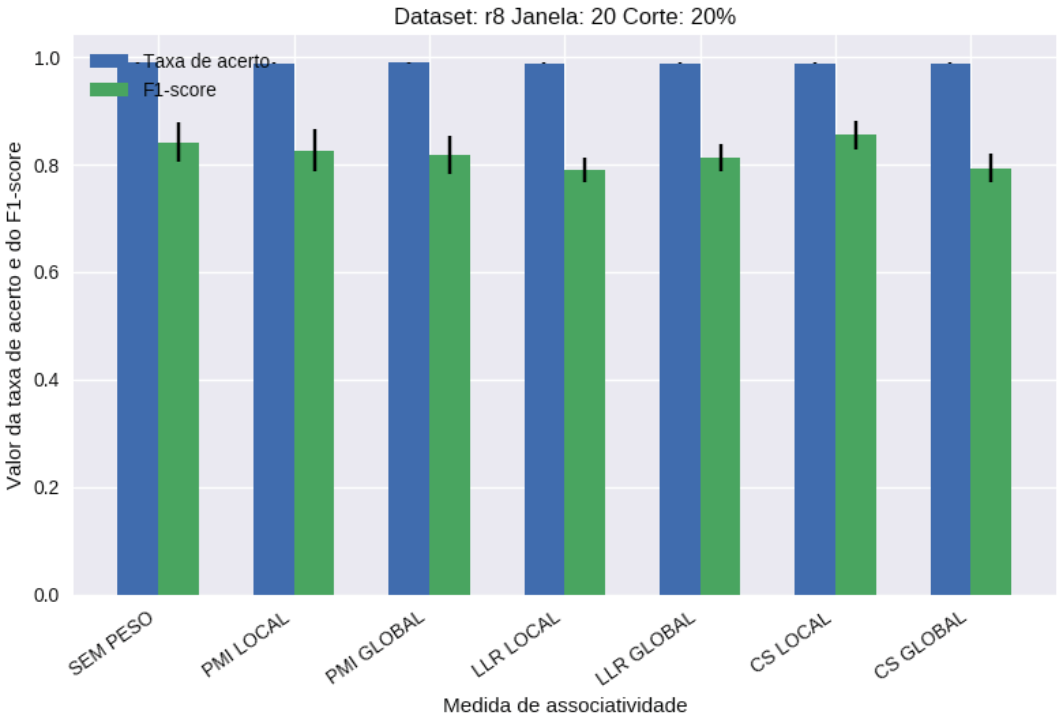


Figura A.27: R8 - Janela de coocorrência 20, filtragem de 20% das arestas

A.4 BASE DE DADOS: 20 NEWSGROUPS

A.4.1 20 Newsgroups - Janela de coocorrência de tamanho 4

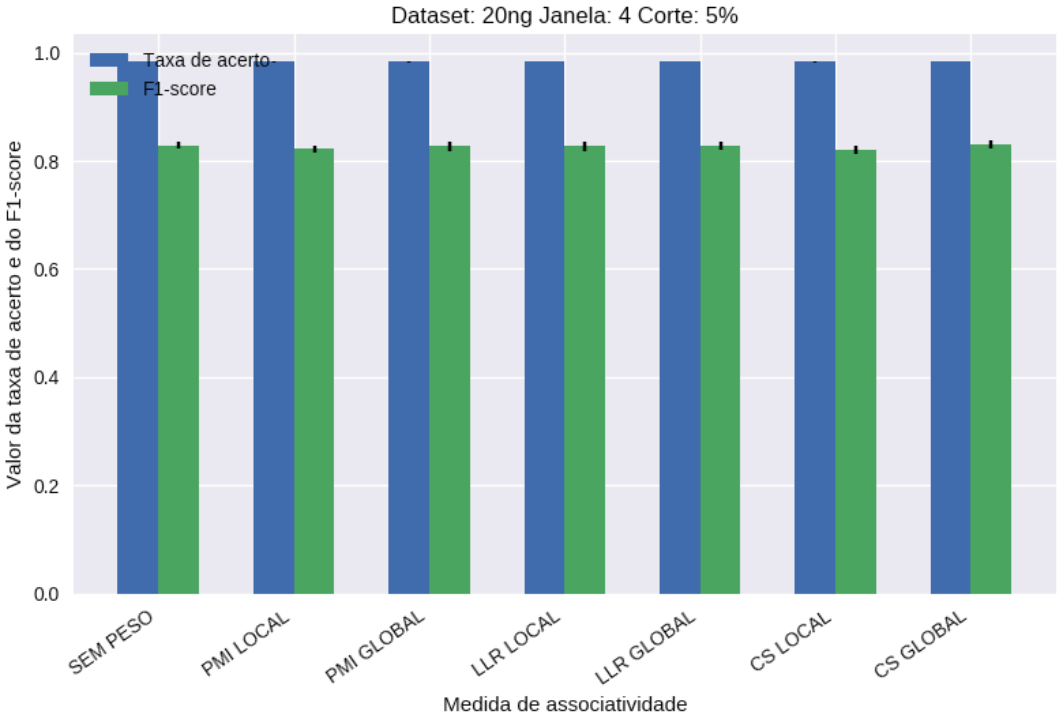


Figura A.28: 20NG - Janela de coocorrência 4, filtragem de 5% das arestas

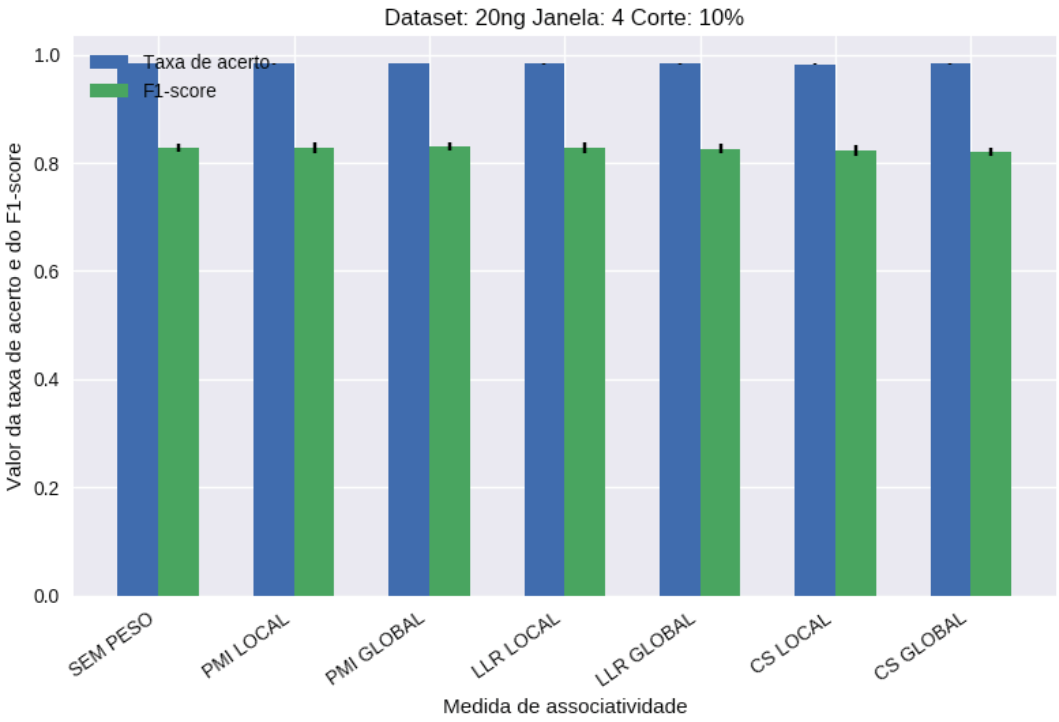


Figura A.29: 20NG - Janela de coocorrência 4, filtragem de 10% das arestas

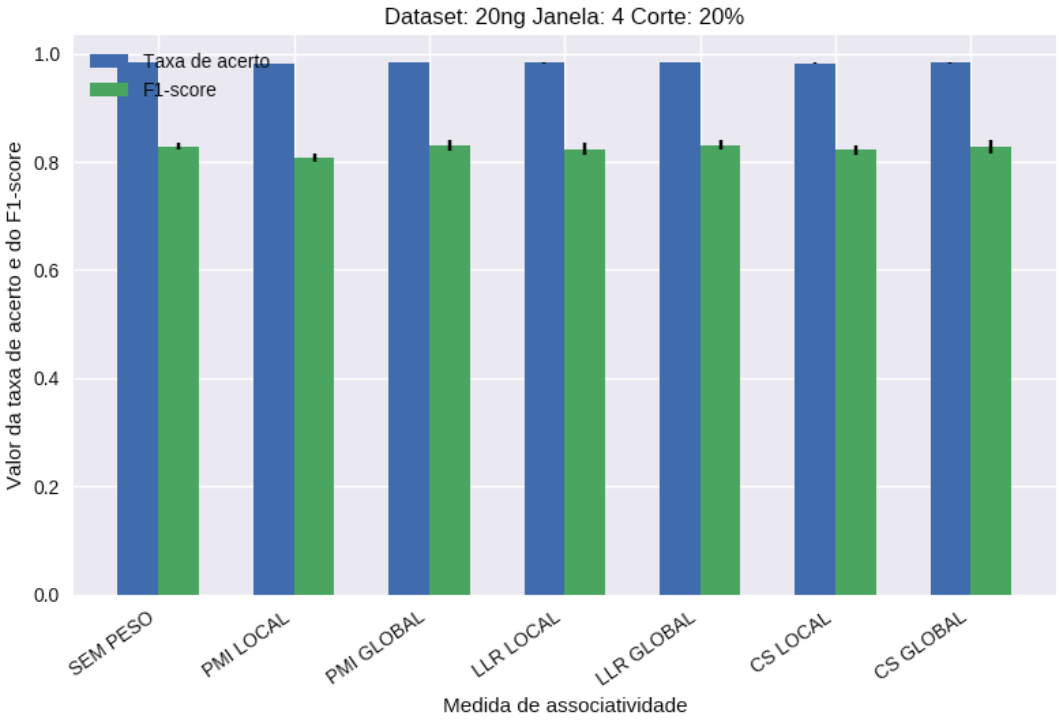


Figura A.30: 20NG - Janela de coocorrência 4, filtragem de 20% das arestas

A.4.2 20 Newsgroups - Janela de coocorrência de tamanho 12

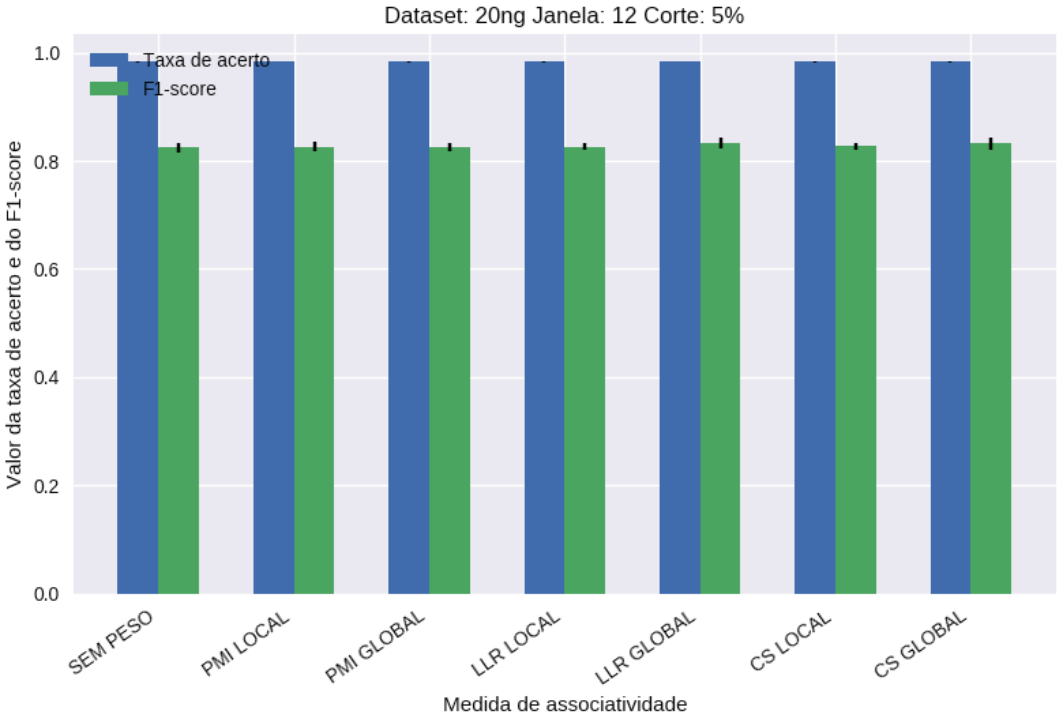


Figura A.31: 20NG - Janela de coocorrência 12, filtragem de 5% das arestas

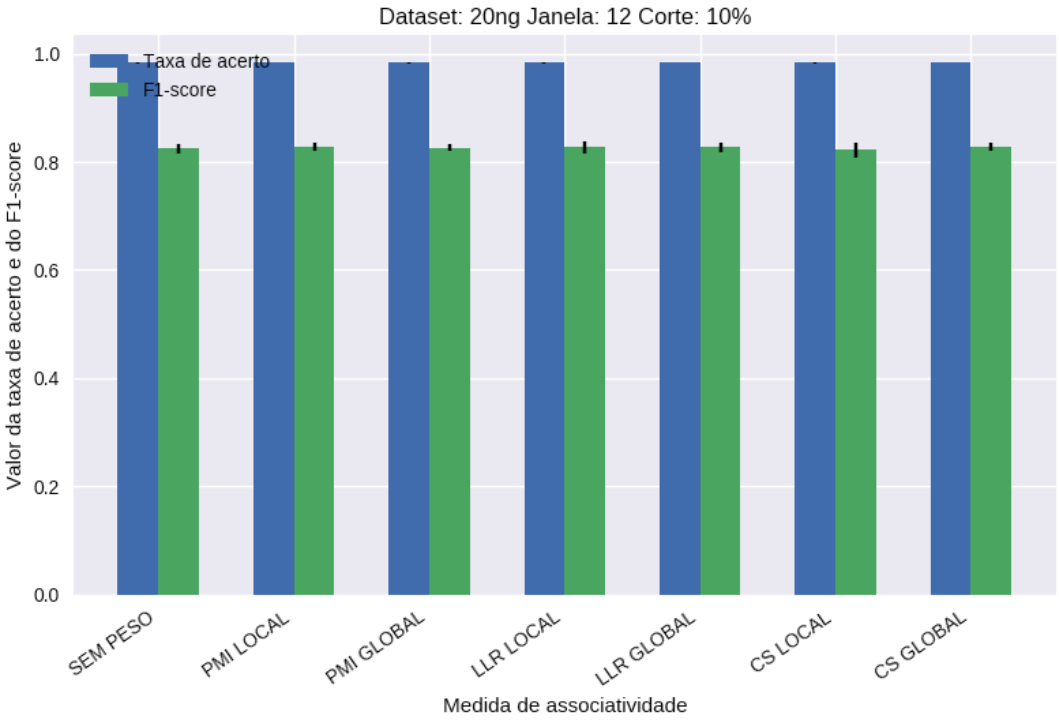


Figura A.32: 20NG - Janela de coocorrência 12, filtragem de 10% das arestas

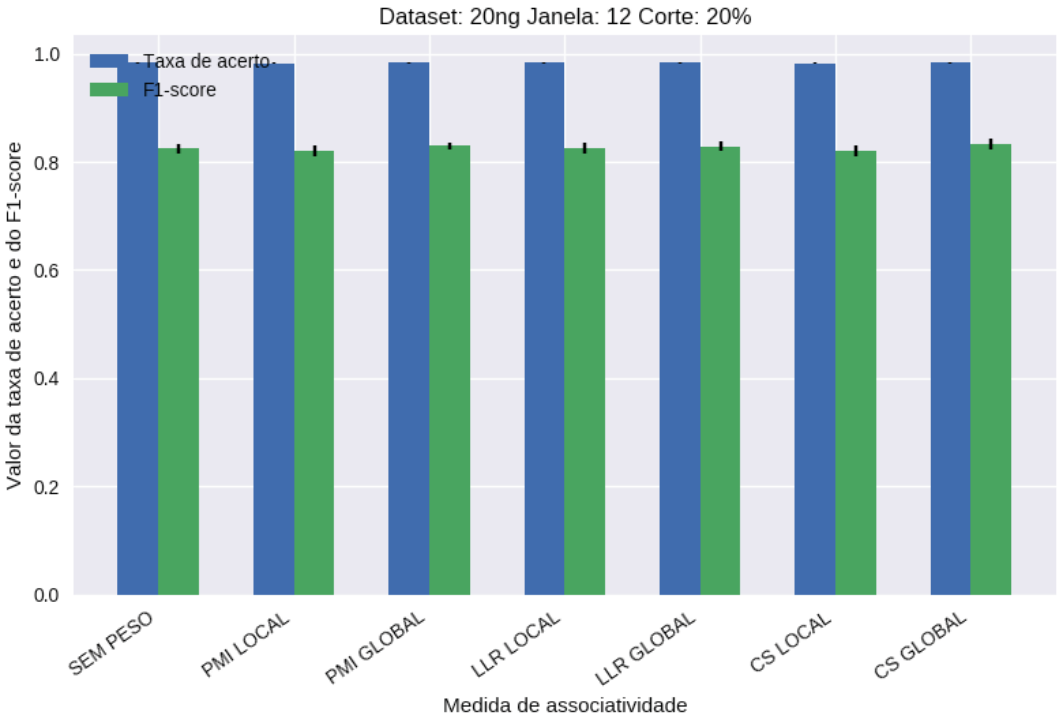


Figura A.33: 20NG - Janela de coocorrência 12, filtragem de 20% das arestas

A.4.3 20 Newsgroups - Janela de coocorrência de tamanho 20

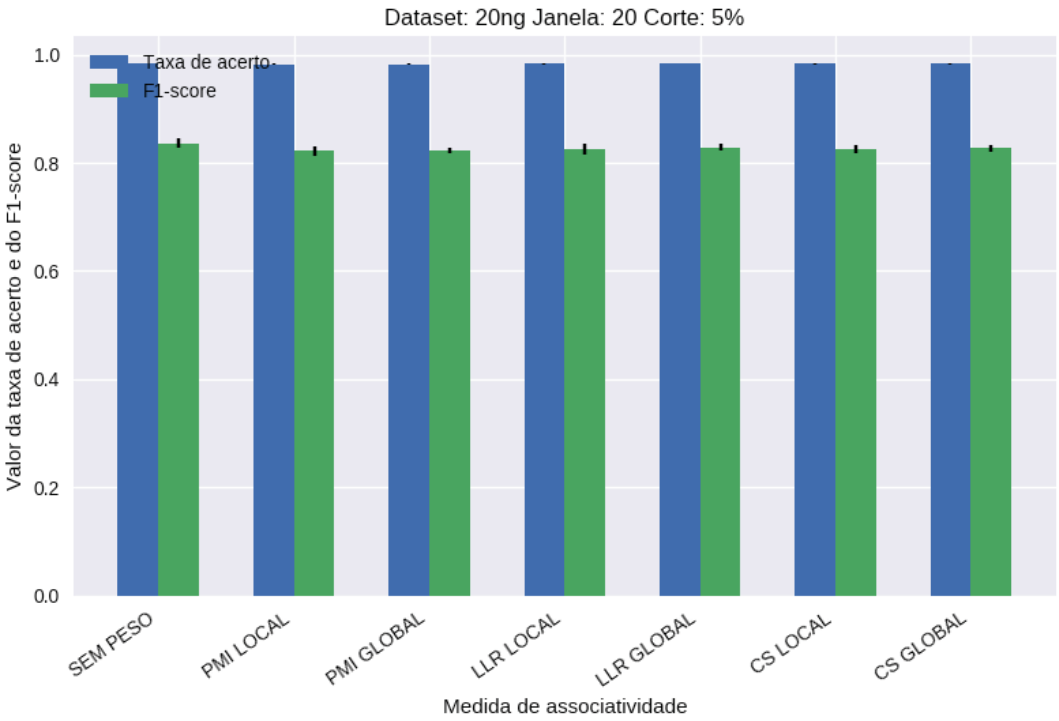


Figura A.34: 20NG - Janela de coocorrência 20, filtragem de 5% das arestas

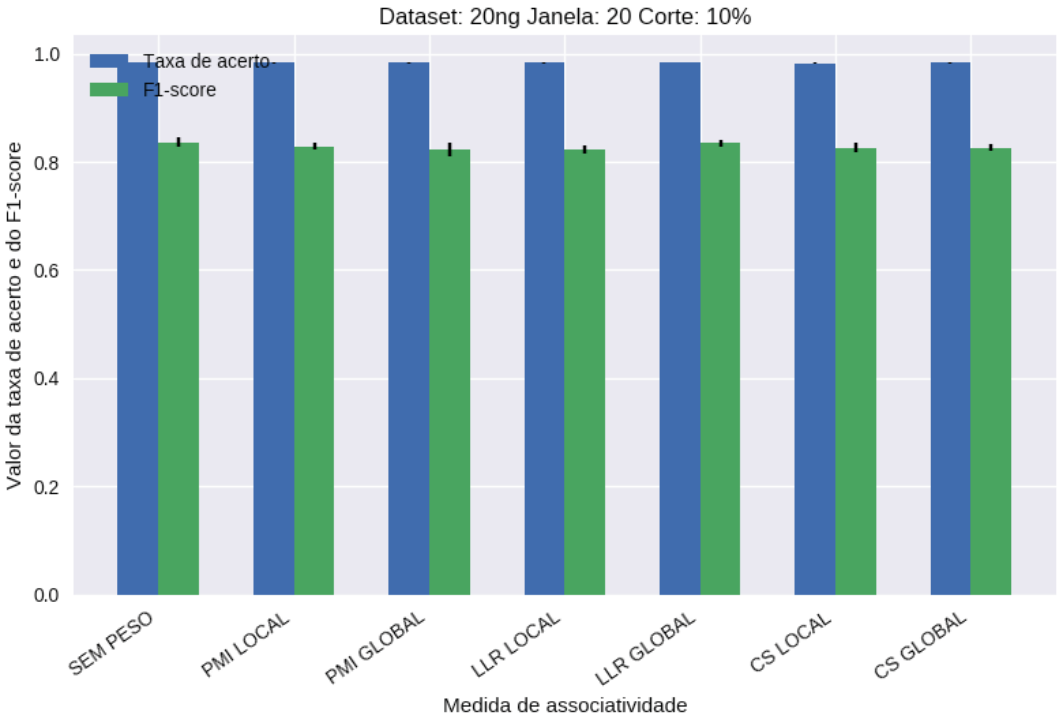


Figura A.35: 20NG - Janela de coocorrência 20, filtragem de 10% das arestas

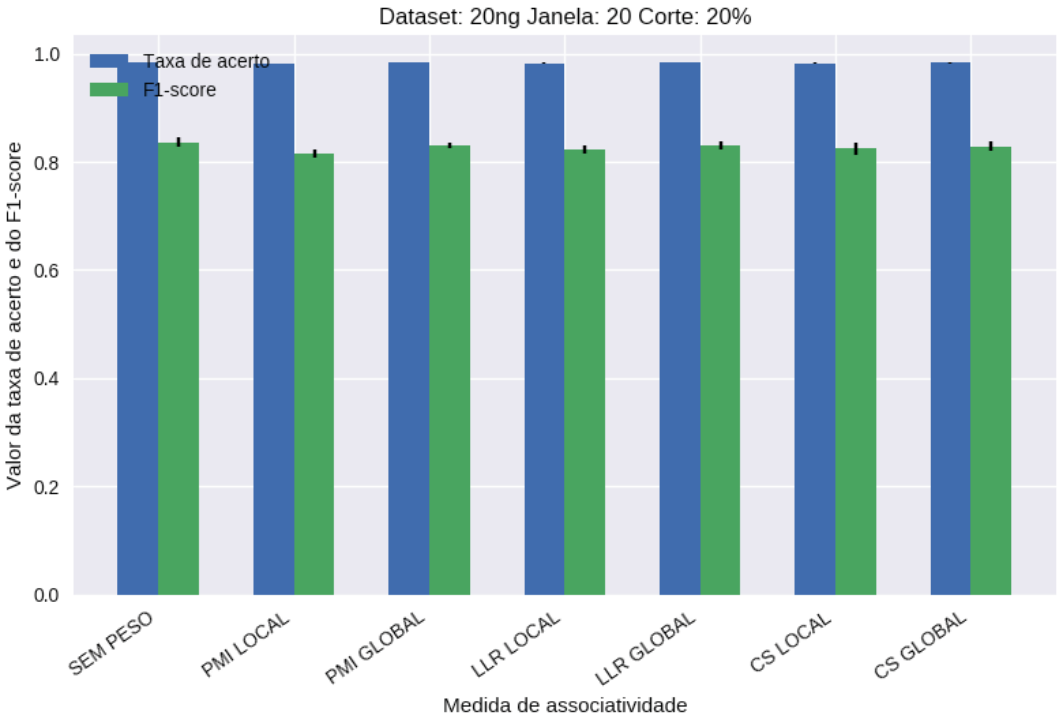


Figura A.36: 20NG - Janela de coocorrência 20, filtragem de 20% das arestas