UNIVERSIDADE
FEDERAL RURAL
DE PERNAMBUCO

João Marcos Nascimento da Silva

Graph Embeddings para Node Classification em representação baseada em grafos de frases em linguagem natural

Recife

2019

João Marcos Nascimento da Silva

Graph Embeddings para Node Classification em representação baseada em grafos de frases em linguagem natural

Monografia apresentada ao Curso de Bacharelado em Ciências da Computação do Departamento de Computação (DC) da Universidade Federal Rural de Pernambuco, como requisito parcial para obtenção do título de Bacharel em Ciências da Computação.

Universidade Federal Rural de Pernambuco – UFRPE

Departamento de Computação

Curso de Bacharelado em Ciências da Computação

Orientador: Rinaldo José de Lima

Recife

2019

Dados Internacionais de Catalogação na Publicação
Universidade Federal Rural de Pernambuco
Sistema Integrado de Bibliotecas
Gerada automaticamente, mediante os dados fornecidos pelo(a) autor(a)

S586g Silva, João Marcos Nascimento da
Graph embeddings para node classification em representação baseada em grafos de frases em linguagem natural / João Marcos Nascimento da Silva. - 2019.
69 f. : il.

Orientador: Rinaldo José de Lima.
Inclui referências.

Trabalho de Conclusão de Curso (Graduação) - Universidade Federal Rural de Pernambuco, Bacharelado em Ciência da Computação, Recife, 2019.

1. Graph embedding. 2. Classificação. 3. Base de dados baseado em grafo. I. Lima, Rinaldo José de, orient. II. Título

CDD 004

Agradecimentos

Em primeiro lugar gostaria de agradecer à minha família, constituída de minha mãe Sílvia Romão, de meu pai Alex Sandro, ao meu irmão caçula, de mesmo nome, chamado de Belete e minha irmã mais velha Emanuela. Não deixar de lembrar da minha tia Margarida, minha vó Fátima e meus tios e tias e uma infinidade de primos.

Em seguida ao meu professor orientador Rinaldo José, que não somente me guiou no desenvolvimento desse trabalho como me auxiliou para sua conclusão assim como durante boa parte do meu tempo nessa instituição de ensino. Ao grupo LSIS no qual faço parte, por ceder recursos físicos para execução desse trabalho. Porém, não posso esquecer de mencionar o professor Tiago Oliveira, mais conhecido por DK, do departamento de matemática, no qual tive o primeiro contato de como fazer ciência.

Agradecer a meu gestor Thiago Brayner e meu coordenador Eudes Filho por permitirem que me ausentasse em alguns momentos do trabalho para a finalização desse trabalho.

E, por fim, a todos meus amigos que fiz em minha vida, em especial àqueles no que fiz nessa universidade no qual compartilhamos trabalhos, muitas dores de cabeças e alegrias.

“A persistência é o caminho do êxito.”
(Charles Chaplin)

Resumo

Devido a grande quantidade de pesquisas desenvolvidas na área biomédica e na disponibilidade de enormes bases de dados sobre entidades biomédicas, incluindo proteínas, genes e vírus, vem a necessidade de se poder indexar de forma automática tais bases de conhecimento humano.

Tal necessidade tem levado ao desenvolvimento e ferramentas computacionais para auxiliar o pesquisador na recuperação de informações específicas envolvendo certas proteínas e suas relações. Neste contexto, dois dos principais problemas na área biomédica envolvendo técnicas de Mineração de Textos (Text Mining) mais investigados são o reconhecimento de entidades nomeadas (REN) e extração de relações.

Este trabalho foca no primeiro problema que serve de base para o segundo, isto é, primeiramente tem-se que se identificar e classificar as entidades para, em seguida, com as entidades identificas e classificadas, identificar as relações existentes entres elas, se houver.

A abordagem adotada neste trabalho é baseada em técnicas recentes de aprendizado supervisionado/não supervisionado de redes neurais profundas, ou Deep Learning (DL) em inglês.

Em particular, investiga-se o problema de REN usando técnicas recentes de representação densa de características (ou features, do inglês) usando DL. Dessa forma, em um primeiro momento, as frases de um corpus da área biomédica são representadas em forma de grafo graças à geração de anotações (metadados) gerados de forma automática por ferramentas de processamento de linguagem natural, tais como tokenização, parsing sintático etc. Em seguida, esses grafos são importados em um banco de dados baseada em grafo para que se possa otimizar diversas consultas que são submetidas a esta base a fim de se extrair atributos (ou features) léxicos e sintáticos das entidades (ou nós) presentes nos grafos. Com informação gerada na etapa anterior, emprega-se uma categoria de algoritmos de Deep Learning chamados Graph Embedding (GE) que mapeam a representação de nós do grafo (entidade) em uma representação densa em um espaço vetorial que possui diversas propriedades de interesse para esta pesquisa. Finalmente, faz-se uso desta representação densa de features (vetor de números reais) como entrada para algoritmos de classificação.

Este trabalho apresenta um estudo experimental onde são comparados alguns dos algoritmos de GE, aliados a diversas formas de representação das frases baseadas

em grafos e seus impactos na tarefa de classificação de entidades (REN), ou node classification. Os resultados experimentais obtidos são promissores alcançando nos melhores casos, mais de 90% de acurácia.

Palavras-chave: Graph Embedding, Classificação, Base de Dados baseado em Grafo.

Abstract

Due to the large amount of works developed in the biomedical field and the availability of huge databases on biomedical entities, including proteins, genes and viruses, it comes the need to be able to automatically index such human knowledge bases.

Such need has led to the development and computational tools to assist the researcher in the recovery of specific information involving certain proteins and their relations. In this context, two of the main problems in the biomedical area involving techniques of Text Mining most investigated are the Named Entity Recognition (NER) and Relation Extraction.

This work focuses on the first problem that serves as a basis for the second, i.e., first we have to identify and classify the entities and then, with the identified/classified entities, identify the existing relations between them, if any. The approach adopted in this paper is based on the recent techniques of supervised/non-supervised learning of deep neural networks, or Deep Learning (DL). In particular, the problem of NER is investigated using recent techniques of dense feature representation using DL.

At first, the sentences from a biomedical corpus are represented as graphs thanks to the generation of annotations (metadata) generated automatically by natural language processing tools, such as tokenization, syntactic parsing, etc. These graphs are then imported into a graph-based database so that various queries submitted to this database can be optimized in order to extract both lexical and syntactic attributes (or features) of the entities (or nodes) present in the graphs. The information generated in the previous step is used as input Deep Learning-based algorithms called Graph Embedding (GE) that map the representation of graph nodes (entity) in a dense vector representation (vector of real numbers) that has several properties of interest for this search. Finally, such dense representation of features) are employed as input for supervised machine learning algorithms.

This work presents an experimental study where some of the existent algorithms of GE are compared, along with several types of sentence representation based on graphs, and their impacts on the task of entity classification (NER), or node classification. The experimental results are promising, reaching more than 90% accuracy in the best cases.

Keywords: Graph Embedding, Classification, Graph-based database.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

AD	Árvore de Decisão
ARBITER	Assess and Retrieve Binding Terminology
AUC	Area Under Curve
BDG	Banco de Dados baseado em Grafo
BDR	Banco de Dados Relacional
CV	Cross Validation
DL	Deep Learning
EI	Extração de Informação
GE	Graph Embedding
GETM	Ghent Text Mining
GGP	Gene or Gene Product
NER	Named Entity Recognition
PNL	Processamento de Linguagem Natural
POS	Part-of-Speech
REN	Reconhecimento de Entidade Nomeada
ROC	Receiver Operating Characteristic
SVM	Support Vector Machine
TEES	Turku Event Extraction System
UMLS	Unified Medical Language System

Sumário

1 Introdução

Nota-se um grande esforço da comunidade biomédica por meio da grande quantidade de publicações realizadas e divulgadas em conferências como artigos e periódicos em revistas especializadas e respeitadas na área. Em consequência, um volume muito grande de documentos é produzido e pode tornar-se um problema para especialistas da área, de ficar a par das recentes descobertas produzidas. Além disso, encontrar uma informação específica dentre este mar de informações pode ser uma tarefa ao mesmo tempo árdua e custosa (JIANG, 2012).

Uma das soluções para atenuar o problema de busca de informação foi o lançamento da versão online do Sistema de Busca e Análise de Literatura Médica (MEDLARS, 1964), o MEDLINE (NLM, 2017). Trata-se de um repositório de resumos da literatura médica e citações em jornais científicos ao redor do mundo mantido pela Biblioteca Nacional de Medicina dos Estados Unidos da América, a NLM em inglês. Atualmente, o banco de dados possui mais de 24 milhões de referências (NLM, 2017), em sua grande parte, conteúdo da área de biomedicina, porém também contém textos de odontologia, farmacologia, veterinária, dentre outras áreas médicas.

Apesar dos esforços contínuos do projeto MEDLINE de manter uma base de dados extensa e atualizada de documentos da literatura médica, o problema de encontrar informações relevantes presentes dentro desses documentos ainda não é completamente resolvido, sendo um grande desafio de pesquisa.

Visando contribuir na solução deste problema, têm sido propostas ferramentas para auxiliar na identificação e extração de informações precisas a partir de um conjunto de documentos. Para isso, são usadas técnicas de representação de dados e Processamento de Linguagem Natural (PLN).

O Reconhecimento de Entidades Nomeadas (REN) é um ramo do PLN que procura extrair e classificar as entidades nomeadas em um texto escrito em linguagem natural. Para essa tarefa, utiliza-se conhecimentos de linguística computacional para manipular as palavras e realização de inferências para sua classificação. Dentro da área biomédica, o reconhecimento de entidades é tido como de grande importância. Pois é a partir dessa tarefa que é possível observar relacionamentos existentes entre entidades mencionadas comuns na área biomédica (proteínas, genes etc.), podendo encontrar achados como a dosagem de um medicamento X aumenta a produção da proteína Y.

Devido a essa importância, uma série de sistemas NER tem sido desenvolvidos com a proposta de identificar entidades nomeadas no domínio biomédico, usando

diferentes abordagens,que podem ser comumente categorizadas em: NER baseada em regras, baseada em dicionário e em aprendizagem de máquina. A primeira delas é a definição de regras ortográficas e morfológicas para a extração dessas entidades.A segunda é no uso de uma estrutura auxiliar para armazenagem das entidades para comparação no texto em linguagem natural. E, por fim, o de aprendizagem de máquina, que é o uso de algoritmos de machine learning. Essa última abordagem será usada no trabalho.

Diante do contexto apresentado e técnicas e procedimentos empregados, pode-se enxergar necessidade no desenvolvimento de bases de conhecimento de conteúdos biomédicos por uso de alguns dos métodos, estabelecendo-se o objeto proposto neste trabalho.

1.1 Problema de Pesquisa

O problema de pesquisa desse trabalho está na análise do comportamento do corpus biomédico IEPA com a geração de diferentes modelagens para sua representação em um banco de dados baseado em grafo aplicados em algoritmos de node *embedding* com a finalidade de serem usados em algoritmos de classificação para detectar e identificar proteínas.

Diante disso essa monografia busca responder as seguinte perguntas:

1. Quais os impactos para classificação de nós quando se varia os modelos de representação baseada em grafos das sentenças, contendo mais ou menos informações?
2. Qual algoritmo de *embedding* e classificação alcança os melhores resultados para cada uma das representações da base?

1.2 Objetivos

Objetivo Geral:

Classificar entidades nomeadas em textos biomédicos com aplicação de algoritmos de node embedding.

Objetivos Específicos:

1. Estudar estado da arte de algoritmos de *node embedding* e classificação.
2. Extrair entidades nomeadas da área biomédica (proteínas, genes, moléculas etc.) a partir de documentos em linguagem natural da MEDLINE.
3. Modelar e povoar banco de dados baseado em grafo com as informações do corpus biomédico IEPA.
4. Aplicar algoritmos de *node embedding* sobre a base biomédica.
5. Comparar desempenho de algoritmos de *node embedding* na tarefa de classificação (Named Entity Recognition).
6. Determinar quais combinações de algoritmos e tipos de *node embedding* são mais efetivos.

1.3 Etapas de pesquisa

As etapas da pesquisa foram divididas em cinco grandes componentes conforme o fluxograma apresentado na Figura 1.

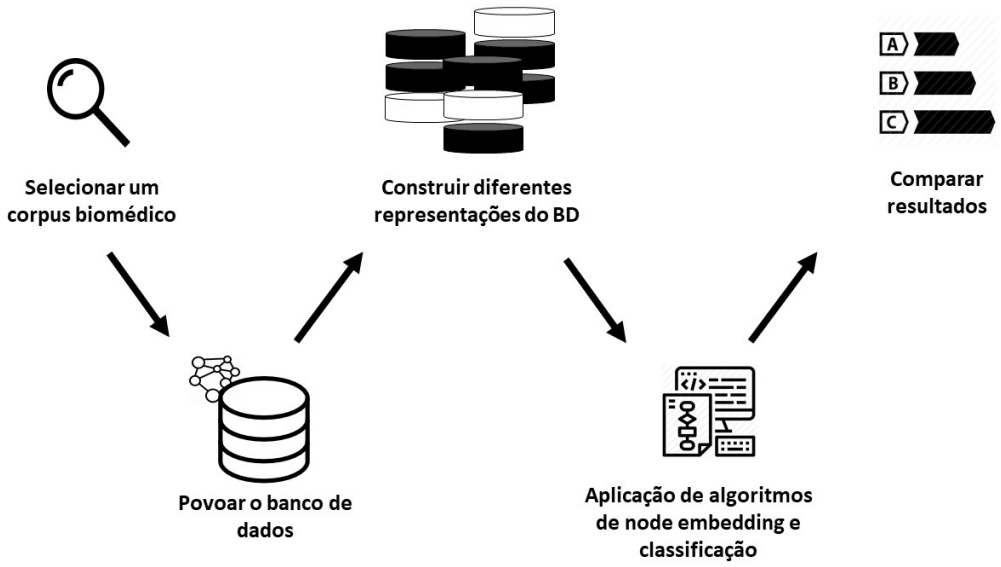

Figura 1 – Macrovisualização das etapas de pesquisa

1. Selecionar um corpus biomédico.
2. Povoar banco de dados com informações extraídas de entidades biomédicas de um corpus de referência da área.
3. Construir representações distintas de frases em linguagem natural a partir de base de dados baseada em grafos.
4. Aplicar algoritmos de node embedding para construção dos vetores a serem usados como entrada nos classificadores.
5. Avaliar e comparar os resultados de classificação.

1.4 Contribuição

Tendo como norte os objetivos propostos para esse trabalho, podemos destacar as seguintes contribuições:

- **Contribuição 1**: Verificação de se a adição de informações ao node embedding contribui com melhoria significativa dos resultados de classificação.
- **Contribuição 2**: Identificação da melhor combinação entre algoritmos de embedding e de classificação para alavancar os resultados finais.

1.5 Estrutura do Documento

Esse trabalho se encontra estruturado da seguinte maneira:

O Capítulo 2 apresenta os conceitos-base usados no decorrer desse trabalho.

No Capítulo 3, é fornecida uma revisão da literatura sobre extração da informação, node embedding e algoritmos NER.

Os resultados e a discussão são apresentados no Capítulo 4.

Finalmente, as observações finais são apresentadas no Capítulo 5

2 Fundamentação Teórica

O propósito desse capítulo é apresentar conceitos-chave importantes para a compreensão deste trabalho. Na primeira seção desse capítulo são abordados conceitos de Processamento de Linguagem Natural. Em seguida, abordado definições importantes de grafo e suas representações nas duas seções seguintes. Complementado com noções de *graph embedding* e definição de algoritmos de *embedding* e classificação nas últimas seções.

2.1 Processamento de Linguagem Natural

Processamento de Linguagem Natural (PLN), também conhecido pela sigla em inglês NLP (*Natural Language Processing*), é uma subárea de ciência da computação e linguística que estuda problemas decorrentes da linguagem humana natural.

Em PLN, comumente há três tipos de unidades sintáticas mais usuais, são elas: a sentença, o sintagma (ou *chunk*) e o *token*.

- **Sentença** é uma construção sintática com sentido completo, composta por uma ou mais palavras; também chamada de frase.
- ***Chunk*** refere-se a um grupo de elementos linguísticos contíguos em uma sentença, composta de um ou mais vocábulos que formam orações; também chamado de sintagma. Pode ser verbal, nominal, preposicional etc.
- ***Token*** pode ser entendido como a menor parte de significado de uma sentença. Há uma forte correlação entre *tokens* e palavras, porém esses não são somente palavras; pode ser os sinais de pontuação, abreviaturas, datas etc.

PLN é uma subárea vasta que dá suporte à resolução de tarefas do mundo real, dentre as quais, têm-se: a extração de informação (EI), recuperação da informação (RI), reconhecimento de entidade nomeada (REN ou NER) e extração de relacionamento.

Extração de Informação é a tarefa que objetiva a captura automática de informações estruturadas dentro de um texto semiestruturado ou não estruturado (PENTEADO et al., 2014). Recentemente, conteúdos de multimídias como imagens, vídeos, áudios etc. tem sido usados para a realização da tarefa. Devido à natureza difícil do problema, comumente sistemas de EI são adotados para domínios específicos.

O Reconhecimento de Entidade Nomeada é uma subtarefa de EI que localiza termos que podem ser denotados por substantivo próprio e *classifica* esses termos

(entidades nomeadas) em categorias pré-definida, como pessoas, locais, organizações e objetos de estudos de uma área do conhecimento (PENTEADO et al., 2014). Embora em línguas como português e inglês haja o auxílio da capitalização de nomes próprios, em alguns contextos essa informação é imprecisa, insuficiente ou não presente em alguns domínios. Um exemplo de caracterização de imprecisão nessas línguas, é que a letra inicial da primeira palavra de uma sentença também é capitalizada. Em outros casos, se mostra insuficiente quando a entidade é composta e se apresenta como não útil, por exemplo, no contexto de pesquisas biomédicas, no qual as entidades podem ser proteínas, doenças, drogas, estrutura subcelular (local), dentre outros, que não são comumente maiusculizadas.

A Extração de Relacionamento é uma subtarefa de EI que tem por objetivo capturar relacionamentos semânticos em textos entre entidades descobertas pelo processo de EI (JIANG, 2012). Uma abordagem usada para sua realização são as ontologias. Os relacionamentos presentes nos textos podem ser representados em diversas linguagens e formalismos, sendo o Resource Description Framework (RDF) (W3C, 2019) um exemplo disso. Um exemplo da aplicação dessa subtarefa é na captura de relacionamentos de termos biomédicos, como proteína-proteína e gene-doença, que relaciona a presença ou ausência de um gene a uma doença.

2.2 Grafos

Os grafos são objetos matemáticos formados por conjunto de vértices V, também chamados de nós, e um conjunto de arestas E, que representam os relacionamentos entre esses nós. Quando um grafo apresenta em seus relacionamentos um valor numérico associado, chamado de peso, que comumente indica a importância ou o custo daquele relacionamento para o grafo, dizemos que o grafo é ponderado.

Quanto à sua estrutura, os grafos podem ser classificados de muitas maneiras. Uma primeira classificação é referente à presença de sentido nos relacionamentos. Ao apresentar essa característica, são chamados de grafos orientados ou dígrafos, caso não, são apenas grafos não orientados.

A partir dos tipos de nós e relacionamentos que apresenta, um grafo pode ser homogêneo ou heterogêneo. É homogêneo quando apresenta um único tipo de nó e um único tipo de relacionamento. Qualquer outro grafo que seja diferente dessa condição é tratado como grafo heterogêneo.

Além dos nós e relacionamentos, um grafo também pode apresentar atributos em sua estrutura. Atributos são propriedades acerca de um nó ou de um relacionamento. Quando o grafo tem a presença de atributos o chamamos de grafo rotulado.

Outros conceitos importantes para a compreensão do trabalho são a relação de adjacência, grau de um nó e caminho. A relação de adjacência é dada pela existência de uma aresta e entre os nós u e v. O grau de um nó é referente ao número de arestas que partem (de saída) ou que chegam (de entrada) num nó. Em grafos não orientados, os graus de entrada e saída de um mesmo nó são iguais. Por fim, um passeio é uma sequência qualquer de arestas adjacentes que ligam dois vértices, que formam um caminho de nós.

Os bancos de dados baseados em grafo (BDG) são uma forma de representação e armazenagem dos dados conciso e simples (JIANG, 2012). Abstratamente, pode-se entender que um BDG é formado por vértices, também chamado de nós, que são ligados entre si por diversos tipos de relacionamentos, as arestas do grafo.

Diferentemente dos bancos de dados relacionais (BDR), forma mais comum de representação dos dados, os BDGs possibilitam que dados possam ser armazenados nos relacionamentos e a definição da direção dos relacionamentos. Outra vantagem sobre os BDRs é a não presença de chaves estrangeiras e, portanto, não há dentro de um nó informações que não sejam pertencentes a ele. Com isso, os BDGs são tidos como uma representação de dados mais natural (JIANG, 2012).

2.2.1 Representações Gráficas

Um conjunto de informações pode ser representado de diversas formas num grafo. Para as informações na Tabela 1, sobre pessoas com relacionamentos de parentesco e suas idades, temos possibilidades de representar os dados da forma mais conveniente para o problema ou situação.

ID	Objeto	Propriedade	Relacionamento	
	Pessoa	**Idade**	**É responsável por**	**É casado com**
0	Cássio	69	Maria	Elizabeth
1	Elizabeth	69	Maria	Elizabeth
2	José	25	-	-
3	Maria	46	José	Paulo
4	Paulo	47	José	Maria
5	Pedro	71	Paulo	-

Tabela 1 – Informações sobre pessoas com parentesco

Uma primeira abordagem para representação em grafo é ter um nó para cada um dos objetos que constitui a base e inserir suas informações sobre eles como atributos. Adicionalmente tratar as informações que os relacionam como arestas desse grafo, conforme imagem na Figura 2.

Outra representação possível (Figura 3) é, assim como a anterior, ter cada elemento da base como um nó, porém, seus atributos também. Cada atributo distinto é criado um nó no qual é ligado para cada um dos nós que representem um membro da família. Os relacionamentos continuam sendo representados como arestas. Esse tipo de representação pode ser ideal para algoritmos de *embedding* que ignoram atributos. Ao transformar esses atributos em nós, esses algoritmos passam a considerá-los.

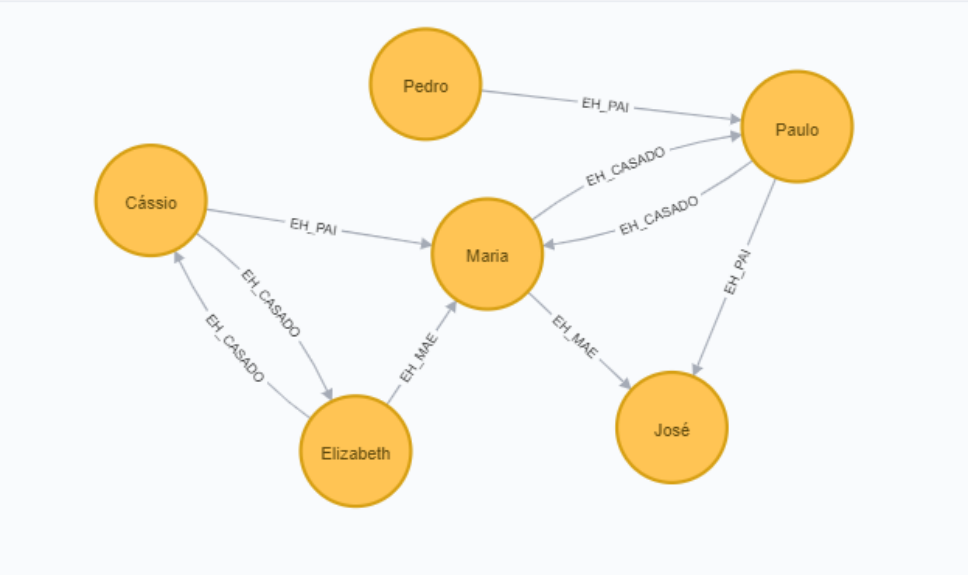

Figura 2 – Representação de um grafo com atributos como propriedades do nó

Várias outras representações em grafo são possíveis. Não há um padrão único de representação. A representação a ser escolhida depende do sentido que deseja se passar por meio do grafo, e, no caso, qual o algoritmo de *graph embedding* será adotado. Nesse trabalho adotaremos a segunda forma, com intuito de permitir aos algoritmos de *embedding* clássicos, DeepWalk (PEROZZI; AL-RFOU; SKIENA, 2014) e node2vec (GROVER; LESKOVEC, 2016), *enxerguem* as informações extraídas, sem grande prejuízo no *embedding* aplicado ao role2vec.

2.2.2 Representações Textuais

Além da representação gráfica, são usadas outras formas de representação, dessa vez textual, comumente usados para leitura por parte de máquinas. A primeira dessas representações é a lista de arestas (*edgelist*). Essa representação mapeia cada relacionamento existente no conjunto de arestas E indicando os nós que o compõem, podendo ainda apresentar adicionalmente o valor de pesos de cada relação. Esse tipo de representação não exige ordem da arestas representadas, porém um vez feita, sua identificação é feita por índices. Devido a isso, tanto o espaço de representação quanto o tempo de busca de um relacionamento no grafo é de $O(E)$.

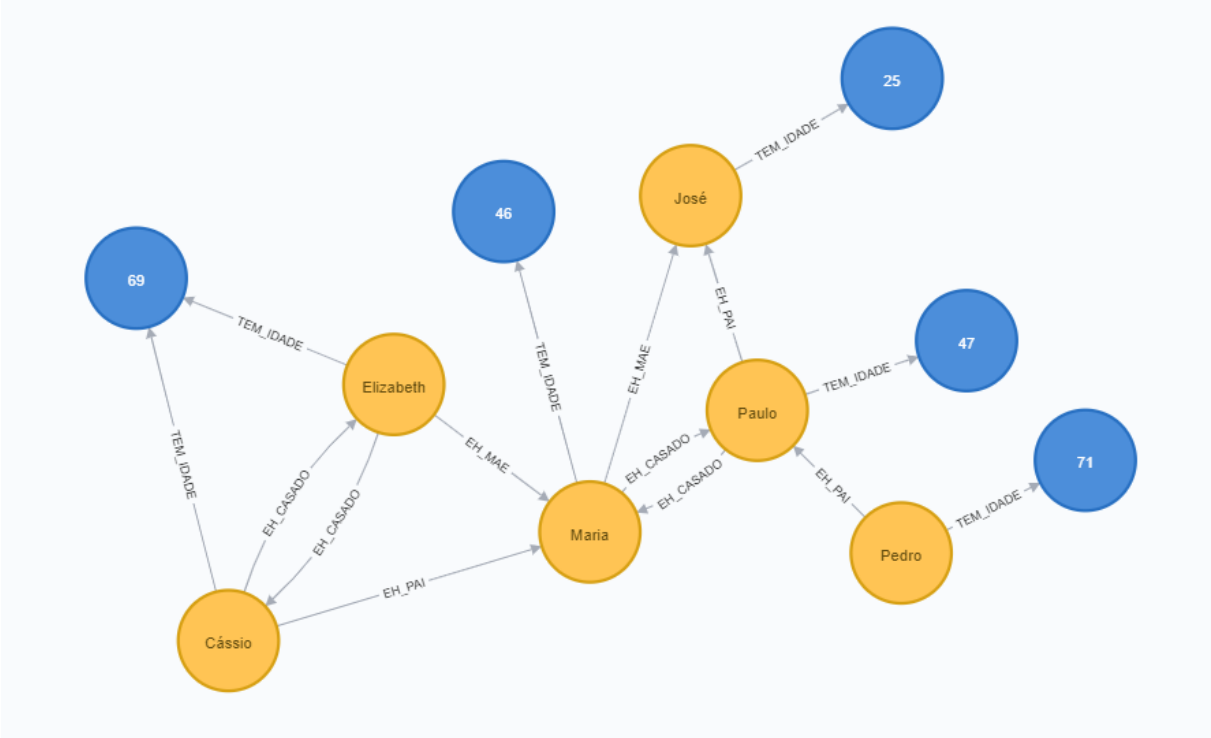

Figura 3 – Representação de um grafo com atributos como nós

```
0 1
0 3
1 0
1 3
3 2
3 4
4 2
4 3
5 4
```

Figura 4 – Representação do grafo *Família* numa edgelist

A outra representação dos grafos se chama matriz de adjacência. Nela, cada nó do grafo é disposto numa linha e numa coluna de uma matriz M de tamanho VXV, sendo V a quantidade de nós no grafo. Cada ponto de interseção na matriz representa uma aresta. Caso a aresta e entre os nós u, v não exista no grafo G, inserimos 0 na posição (u, v) da matriz de adjacência. Caso haja uma aresta ligando os dois nós, inserimos o valor 1 naquela posição. Há ainda uma variação, para representar pesos em grafos, no qual inserimos o valor do peso na posição (u, v) da matriz, em

vez de somente, a indicação de presença de aresta. Para grafos que não possuam nós autorreferenciados, a diagonal principal da matriz de adjacência é preenchida somente por zeros. Outro ponto interessante das matrizes adjacência, é que somente por observação dela é possível distinguir se um grafo é direcionado ou não direcionado. Sendo não direcionado, por não apresentar arestas com sentido, para posições opostas na matriz têm-se o mesmo valor, construindo-se uma matriz simétrica.

	0	1	2	3	4	5
0	0	1	0	1	0	0
1	1	0	0	1	0	0
2	0	0	0	0	0	0
3	0	0	1	0	1	0
4	0	0	1	1	0	0
5	0	0	0	0	1	0

Figura 5 – Representação do grafo *Família* numa matriz de adjacência

Uma matriz adjacência possui uma ordem constante - $O(1)$- para inserção, remoção e busca de arestas. Contudo, por necessitar representar todos os nós do grafo, para um grafo muito grande, de ordem de milhões e bilhões de nós, o espaço requerido para essa representação é um grande problema - $O(V^2)$. O uso desse espaço se mostra ineficiente quando, num universo de grafos muito grandes, a maior parte das posições na matriz são ocupadas por zero, denotando uma matriz esparsa.

Para corrigir o problema deixado pela representação da matriz adjacência, têm-se a lista de adjacência, que é um modelo híbrido dos dois já referenciados acima. Nessa representação, cada vértice do grafo é listado, tido como a chave da lista de nós a qual esteja ligado, funcionando como a estrutura de armazenamento *tabela de decisão*. Com isso, para encontrar todos os possíveis vizinhos de um nó, o tempo de busca é de $O(1)$. Para uma aresta específica (u, v), precisamos encontrar o nó v na lista do nó u, sendo necessário uma busca na ordem de $O(d)$, sendo d o grau do nó. Todos os nós a qual não é ligado, que na matriz adjacência são tidos por zero, não são representados, diminuindo o espaço necessário para sua modelagem.

Nesse trabalho, daremos foco à representação *edgelist*, por ser uma forma simples de representação e ter um custo de busca e espaço de ordem constante, sendo a usada pelos algoritmos de *embedding* utilizados nesse trabalho.

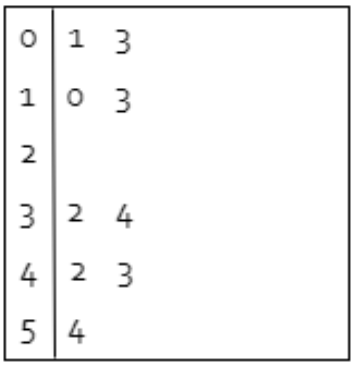

Figura 6 – Representação do grafo *Família* numa lista de adjacência

2.2.3 Graph Embedding

Graph embedding é o processo de transformação de um grafo num conjunto de vetores espaciais de baixa dimensionalidade. O espaço gerado pelo conjunto de vetores preserva propriedades dessa rede, de modo a garantir que nós mais similares, que compartilhem relações, fiquem mais próximos entre si nesse novo espaço. Há uma série de motivos para uso do *embedding* em contrapartida de aplicação de métodos sobre o grafo (GODEC, 2018).

O primeiro motivo é que há todo um universo já amplamente estudado e um grande conjunto de ferramentas para aplicação de algoritmos em vetores (*embedding*) devido à bagagem de conhecimento acumulado pelos anos de estudo sobre vetores e suas propriedades que possibilitaram alcançar bons resultados em algoritmos que trabalham com esses objetos matemáticos em detrimento de grafos. Outro ponto positivo para uso de *embedding* é sua representação comprimida no qual comumente exige menor espaço de armazenamento e, por conseguinte, menor tempo de processamento. Além disso, os *embeddings* por serem vetores possibilitam operações matemáticas rápidas e simples, como soma e multiplicação por escalar, se comparadas com operações sobre grafos.

Apesar das vantagens do uso de *embedding*, este possui uma série de desafios que apesar de estarem sendo superados não estão ainda suplantados. Por ser um *embedding*, há a necessidade de representação num objeto menor, o vetor resultado do processo, o nó ou o grafo de entrada sem que haja grande perda da informação presente. Devido a isso são usadas uma série de considerações de conceitos presentes em grafos e busca-se aplicá-los para uma melhor representação do grafo. Além desses conceitos, o uso de informações como pesos nas arestas do grafo ou a presença de atributos nos nós e relacionamentos podem ser primordiais para uma melhor resultado de *embedding*.

Outro ponto desafiador é a eficiência desses algoritmos em grafos de grande dimensão. Grafos muito grandes podem exigir muito esforço computacional para percorrer

cada nó/subgrafo presente no grafo e gerar n caminhos aleatórios para representação em vetores. Ainda seguindo nessa área, maiores *embeddings* preservam mais informação, porém induzem mais espaço e tempo de complexidade (128 e 256 são usualmente usados). Em virtude disso, eficiência no consumo de tempo e espaço são questões relevantes no âmbito de geração de *embedding* em grafos.

É importante salientar que existe na literatura uma sobrecarga do termo *graph embedding*. Isso se deve pela existência de duas categorias de *embeddings* que podem ser realizados sobre grafos. A primeira categoria corresponde a conversão de cada nó do grafo num vetor de espaço vetorial R^t, sendo t um número inteiro positivo. Essa abordagem é a tradicional, usada nas maiorias das situações, e, é conhecida por o *vertex/node embedding*. Enquanto o último, uma abordagem mais recente, o *embedding* é realizado em nível de subgrafos (não nós), os transformando em vetores de R^t. Tal abordagem pode ser encontrada no mapeamento de compostos químicos, no qual sua natureza pode ser interpretada como um grafo. Contudo, pelas áreas não terem se desenvolvidos ao mesmo tempo, o termo *graph embedding* era inicialmente empregado somente para *embeddings* sobre nós. Devido a isso, se usa até hoje o termo *graph embedding* para representá-la.

Nesse trabalho ao falarmos de graph embedding estaremos nos referindo a node embedding, a não ser que seja explicitamente dito o contrário.

2.2.4 Algoritmos de embedding

Para a obtenção dos vetores de embedding de um grafo, existe hoje em dia, uma miríade de algoritmos, categorizados a partir do modo de seu funcionamento (GOYAL; FERRARA, 2018), que são:

1. **Fatoração** Matrizes são usadas para representar as conexões entre os nós, podendo ser usadas as matrizes de adjacência, laplaciana, de similaridade, dentre outras. A matriz escolhida passa pelo processo de fatoração, que é a decomposição da matriz em outras duas ou mais matrizes. É um método que para grafos muito grandes pode não ser o ideal, por ter um custo computacional na etapa de fatoração, na ordem de $O(|V|^2)$, sendo $|V|$ a quantidade de vértices num grafo. (IME, 2019)

2. **Deep Learning** Com o advento das pesquisas e aumento das aplicações de *Deep Learning*, o uso dessa técnica para realização de *embedding* está se popularizando. Para isso são usados redes neurais especiais chamadas *autoencoders*. *Autoencoders* são redes neurais que compactam a entrada em um espaço latente, para em seguida, a reconstruir em sua saída (KINGMA; WELLING, 2013). Portanto tais redes neurais tentam aprender a aproximação à função identidade, em

geral com aplicação de restrições, como menor dimensão que a entrada original, sendo usado como redutor de dimensionalidade.

3. **Random Walk** Nesse conjunto de algoritmos, são realizadas para cada nó v contido no grafo G n caminhadas aleatórias de tamanho t, de forma a mapear os vizinhos desse nó v de n maneiras (GOYAL; FERRARA, 2018). Em muitos algoritmos, como DeepWalk e node2vec, as orientações das arestas não são levadas em consideração. Essas n caminhadas aleatórias geradas são usadas para a criação de um vetor espacial de tamanho l para cada um dos nós.

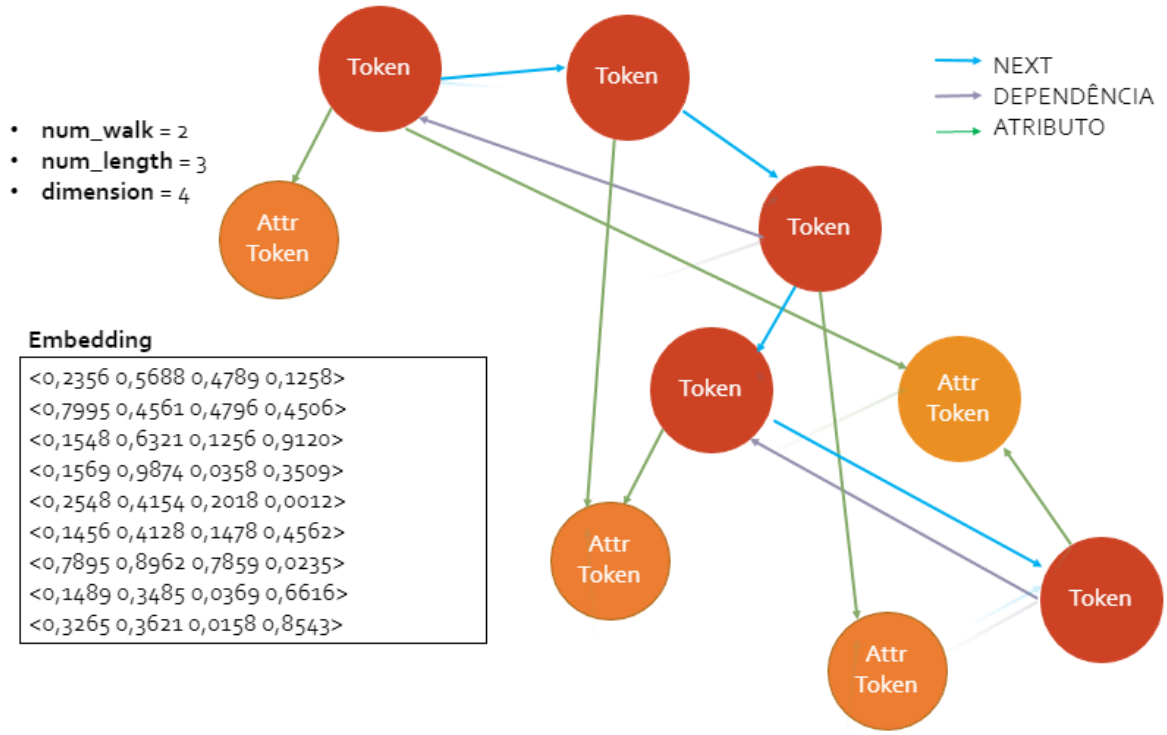

Figura 7 – Embedding com duas caminhadas aleatórias de tamanho 3 para cada nó com geração de um conjunto de vetores espaciais quadrimensional

Devido à sua simplicidade, custo computacional controlado e sua ampla adoção noutros trabalhos, figurando seus algoritmos dentre os principais no estado da arte, usaremos de algoritmos de embedding dessa classe. Abaixo, estão os algoritmos usados nesse trabalho.

DeepWalk O algoritmo DeepWalk é o precursor de todos os algoritmos de caminhadas aleatórias não supervisionados, sendo também o mais simples. É tido como um algoritmo de caminhada aleatória uniforme pois a sua estratégia de amostragem dos nós para comporem o caminho é o uso randômico (sem qualquer viés). Tal processo é bastante parecido ao usado no embedding de palavras como o word2vec (PEROZZI; AL-RFOU; SKIENA, 2014).

Após esse passo, as caminhadas aleatórias são alimentadas ao algoritmo word2vec para geração do vetor de representação de cada nó. É importante salientar que

DeepWalk devido à sua natureza de distribuição uniforme não faz distinção entre pesos presentes em grafo, portanto sendo ignorados, pois para ele, qualquer aresta entre dois nós possui o mesmo valor semântico.

Node2Vec A diferença desse algoritmo para o anterior se reside no uso de duas variáveis, p e q, que são usadas para produzir o viés pelo dos algoritmos de busca em largura (BFS) e de busca em profundidade (DFS). Após a transição para o nó v de u, o hiperparâmetro de retorno, p, e o hiperparâmetro de entrada-saída, q controlam a probabilidade de uma caminhada revisitar nós (u) e permanecendo perto dos nós precedentes (x_1) ou movendo-se para fora mais longe (x_2, x_3) respectivamente. Em resumo, o parâmetro p escolhe os nós vizinhos mais próximos, enquanto o último faz caminho explorando o grafo por vizinhos mais distantes (GROVER; LESKOVEC, 2016).

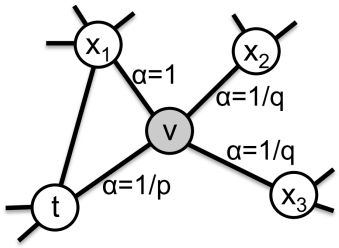

Figura 8 – Ilustração do random walk no node2vec. Fonte: (GROVER; LESKOVEC, 2016)

Role2Vec É um algoritmo que usa dos diferentes papéis que cada nó pode assumir dentro de um grafo para geração de seu embedding. Aqui é entendido que nós com papéis semelhantes possuem estruturas parecidas como tipos de nós a qual está ligado, orientação no sentido do relacionamento etc. (AHMED et al., 2018). Possui semelhanças com o struc2vec (RIBEIRO; SAVERESE; FIGUEIREDO, 2017).

2.3 Classificação

A aprendizagem supervisionada é um dos modos de aprendizagem de máquina. Nela são fornecidos exemplos em pares de entrada e saída desejada ao algoritmo com finalidade de construir regras para correto mapeamento entre eles. Uma das tarefas mais comuns da aprendizagem supervisionada é a classificação, que é o processo de atribuição de um rótulo a uma entrada. Na subseção abaixo detalharemos os algoritmos de supervisionados usados na tarefa de classificação nos experimentos desse trabalho.

Segundo (DIMENSõES, 2014), um bom modelo de predição é construído a partir da boa generalização dos dados de treinamento para qualquer conjunto de dados do domínio. A generalização é tida como o grau de aprendizado de um algoritmo,

definindo o quanto ele aprendeu de fato. O *underfitting* assim como o *overfitting* correm quando o algoritmo não consegue generalizar bem o problema. O primeiro, também conhecido como sobreajuste, refere-se à situação no qual o algoritmo se ajustou demais ao conjunto de treinamento, perdendo a capacidade de generalizar para exemplos ainda não apresentados (DIMENSõES, 2014). Em tais situações, não é incomum dizer que o algoritmo *decorou*. Nessa situação, as métricas de avaliação no conjunto de treinamento possuem bom desempenho, porém durante o teste ou validação, é verificado uma queda agressiva desses valores. Enquanto *underfitting* denuncia um problema oposto. Ocorre quando um modelo de aprendizado de máquina não é complexo o suficiente para capturar com precisão as relações entre as características de um conjunto de dados e a variável de destino, construindo um modelo que não consegue predizer bem na validação como também no treinamento.

2.3.1 Algoritmos de classificação

Naive Bayes Algoritmo de classificação baseado no teorema de Bayes com assunção de independência entre as características. Isto é, cada um dos atributos do vetor é tratado como condicionalmente independentes, mesmo que essas características dependam umas das outras ou da existência de outras características, todas essas propriedades contribuem independentemente para a probabilidade; o que não é comum de se encontrar em situações do mundo real. Sendo daí, a origem do nome *ingênuo* no algoritmo. Apesar da simplicidade em sua definição, o classificador continua a ser utilizado na literatura científica, apresentando resultados tão bons quantos os resultados de algoritmos mais sofisticados em diversos problemas.

Regressão Logística É uma técnica estatística para analisar um conjunto de dados no qual existem uma ou mais variáveis independentes que determinam um resultado. O resultado é medido com uma variável dicotômica (na qual existem apenas dois resultados possíveis). O objetivo da regressão logística é encontrar o modelo mais adequado para descrever a relação entre a característica dicotômica de interesse (variável aleatória = resposta ou variável de resultado) e um conjunto de variáveis independentes (preditoras ou explicativas).

Árvore de Decisão (AD) É um modelo estatístico que trabalha com dados categóricos e numéricos. Usa da estratégia dividir para conquistar ao quebrar os atributos em subconjuntos para montagem da árvore de forma incremental. A cada iteração o atributo que consiga seccionar melhor os exemplos, apresentando maior ganho de informação, é escolhido. Sendo esse porém, um dos grandes problemas enfrentados pelas ADs. No algoritmo de partição recursiva, a árvore estende a sua profundidade até o ponto de classificar perfeitamente os elementos do conjunto de treinamento. Tal comportamento pode ocasionar o *overfitting* do problema.

Random Forest (Floresta Aleatória) É um método de aprendizagem supervisionado de *ensamble* que constrói um conjunto de árvores de decisão no treinamento com finalidade aumentar a acurácia e ter maior estabilidade na classificação. Ao invés de procurar pela melhor característica ao fazer a partição de nós como uma AD, ele busca a melhor característica em um subconjunto aleatório das características. Este processo cria uma grande diversidade, o que geralmente leva a geração de modelos melhores. Por trabalhar com subconjuntos aleatórios de características, tende a construir árvores menores. Com essa ação, florestas de árvores conseguem contornar bem o sobreajuste.

XGBOOST Algoritmo de aprendizagem de máquina baseado em árvore de decisão que se utiliza de gradient boosting desenhado como uma solução algorítmica eficiente e rápida aos seus antecessores. Devido a essa característica, pode ser aplicado a conjuntos enorme de dados, da casa dos bilhões (CHEN; GUESTRIN, 2016).

Redes Neurais Consiste de um conjunto de unidades chamadas neurônios arranjadas em camadas. Caso a camada esteja disposta no início da rede é chamada de camada de entrada; ao fim dela, chamada de camada de saída, ou ainda, entre esses neurônios, a camada escondida. Cada neurônio da rede recebe um sinal diretamente da entrada ou da camada anterior multiplicado ao seu peso. São esses pesos que são otimizados pela rede neural na fase de treinamento. Somado a isso há a aplicação de uma função escolhida, a função de ativação, a cada um dos neurônios para a introdução de um componente não linear no aprendizado da rede. Atualmente é uma técnica de grande destaque na literatura científica devido aos bons resultados apresentados em comparação a outros algoritmos.

SVM O objetivo desse algoritmo é traçar um hiperplano que consiga separar as classes com maximização da distância entre esse hiperplano e os pontos dessas classes. Com essa maximização, é reforçado que a classificação terá maior confiança, pois há uma margem maior para categorizar um nova observação. Para conseguir essa maximização é necessário encontrar os vetores de suporte, no qual dão o nome ao algoritmo, que são vetores que auxiliam no delineamento da separação das observações. É um algoritmo que funciona muito bem em domínios não lineares com margem de separação clara, contudo não funciona bem para conjunto de dados com grandes ruídos ou muito grandes.

3 Trabalhos Relacionados

Muita propostas já foram realizadas desde a tarefa de extração de conteúdo biomédico como também a construção de bases de conhecimento relacionadas. Algumas delas se encontram listadas abaixo e foram usadas como referência para proposição deste trabalho.

Em (MCCRAY; SRINIVASAN; BROWNE, 1994), apresenta-se uma abordagem na resolução do problema de variação léxica, que pode ser morfológica (derivações ou inflexões) e ortográfica (diferentes escritas de uma palavra), dos termos biomédicos. Tal classe de problema, é tratado como difícil para sistemas computacionais, pois há um grande número de possibilidades de variações do termo como também de sua combinação com outras palavras. Algoritmos de *stemming*, subtarefa do PLN que é constituída da aplicação de uma série de regras ao texto, como a supressão de sufixos e prefixos, para encontrar os troncos de cada palavra, são usados para mitigar o problema.

Para solucionar o problema, conta-se com a ferramenta *SPECIALIST Lexicon*, que é um léxico de termos biomédicos em inglês, que compõe o conjunto de aplicações disponibilizadas pela *Unified Medical Language System* (UMLS). Em cada uma das entradas, que podem ser termos simples ou compostos, tem-se informação morfológica, sintática e ortográfica. Termos que possuem o mesmo tronco linguístico, a forma não flexionada das palavras, encontram-se sob uma mesma entrada no dicionário léxico.

Pela proposta apresentada, o programa escrito na linguagem C usa dados do *SPECIALIST*, como também compara as diversas fontes de itens léxicos. É composto por vários módulos, que podem ser combinados para obter diferentes resultados, que podem ser somente a inflexão do termo ou remoção de *stopwords*.

O princípio do programa se baseia na realização de procedimento comum tanto na base de origem como na base alvo. Se os termos na base origem são minusculizados, então na base de destino deve também ser realizado o mesmo procedimento. Uma solução complementar foi adicionada para a normalização de palavras. Nessa rotina, todas as sentenças passam pelo processo de serem segmentadas, sendo cada termo transformado para sua forma minúscula, convertida para sua forma inflexionada, ignorando-se pontuação e palavras compostas sendo ordenadas alfabeticamente. O procedimento também reconhece variações greco-romanas assim como variações irregulares de plurais de palavras. Ao usar essa rotina, torna-se possível que ao buscar por ”suture”, traga-se resultados como ”suture”, ”closely by suture”, ”suture granuloma”ou ”suturing”, por exemplo.

Três bases são usadas no artigo, a primeira base contém mais de 10 mil pares de variações derivacionais conhecidos, como *dosage/dose*. Ela é usada para a geração das regras para o módulo de morfologia derivacional. A segunda base é composta por aproximadamente dois mil pares que relaciona termos com o mesmo significado, por exemplo, *asphyxiation/suffocation*. Se um termo no par está presente no *Metathesaurus* e o outro não, uma entrada à base de conhecimento é adicionada. A última base contém quatro mil variantes de escrita, por exemplo, *leukocity/leucocity*, extraídos do *SPECIALIST Lexicon*. Também pode ser usada para adicionar uma nova entrada ao *Metathesaurus* sob a mesma condição já explicitada acima.

(CRAVEN; KUMLIEN, 1999), apresentam a construção de uma base de conhecimento a partir da extração de informação de textos biomédicos. Os textos usados são um conjunto aleatório de resumos colhidos da plataforma MEDLINE. As informações mapeadas na base de conhecimento desenvolvida foram focadas na relação entre proteínas e estruturas subcelulares em que são encontradas.

Para realizar o objetivo, inicialmente, em cada texto processado foi realizado o processo de stemming para reduzir as variações das palavras. Passada essa fase, é executada a etapa de extração das entidades. Extrai-se uma relação r(x, y) se x pertence a classe X e y pertence a Y e se ela for considerada positiva pelo modelo estatístico. De acordo com o artigo, Kumlien foi responsável pela anotação manual dos resumos no corpus com instâncias da relação desejada. A anotação manual dos resumos para essa relação somente realizara se fosse encontrado explicitamente no texto que ”proteína x está localizada em y”.

Em (RINDFLESCH; RAJAN; HUNTER, 2000), é apresentado o *Assess and Retrieve Binding Terminology* (ARBITER), programa desenvolvido em Prolog usado para extrair asserções sobre relações macromoleculares de textos biomédicos. O seu diferencial frente a outros é que o ARBITER usa uma fonte de domínio de conhecimento existente, o MEDLINE, e informações sintáticas providas por um parser. Ao fim, o objetivo é construir automaticamente uma base de dados de relacionamentos em citações do MEDLINE.

Os relacionamentos de interesse no artigo são os que representam ligações moleculares. Para capturar esses relacionamentos, foram usados 491 mil resumos do MEDLINE, durante o qual foram extraídos quase 25.000 relacionamentos de ligação (RL) adequados para a entrada em um banco de dados de função macromolecular. Para encontrar os RLs, é usado como indicador de possível relacionamento a presença da palavra inglesa bind e suas variações numa determinada sentença.

Primeiramente, o software SPECIALIST, do conjunto de ferramentas da UMLS, é aplicado aos textos. Tal aplicação é baseado no conceito de fronteiras de palavras, proposto por Tersmette et al. (1998), usado para detectar termos biomédicos compostos.

O algoritmo delimita fronteiras numa sentença a partir de palavras com pouco conteúdo informacional biomédico, como preposições, artigos, conjunções etc. Ao fim do procedimento são fornecidas informações sintáticas parciais sobre os termos capturados, concentrando-se em frases nominais simples.

No passo seguinte, usando a ferramenta MetaMap, associa-se cada termo detectado a um conjunto reduzido de significado do Metathesaurus. A última etapa corresponde a identificação de termos ligantes, que são estruturas da língua que podem ser usadas para ligar duas frases nominais. Como exemplo, têm-se as preposições, os particípios, apostos ou verbo to be. Ao encontrar essas condições, em conjunto de palavras variantes de bind, são capturados os relacionamentos de interesse.

Em (LANDEGHEM et al., 2012), em primeiro momento, é dada a importância de compreender frases nominais com a presença de símbolos de genes, as relações de entidades, para a correta interpretação dos resultados de mineração. Em seguida, são comparados dois frameworks de aprendizagem de máquina presentes no evento de *BioNLP Shared Task 2011*, o *Turku Event Extraction System* (TEES) e o *Ghent Text Mining* (GETM), que usam dessa abordagem.

O TEES é uma ferramenta de extração de relacionamentos representado em grafos extensíveis. O sistema é constituído de três principais componentes: o Reconhecedor, a Detecção de Termo e a Detecção de Arestas. O primeiro componente é um parsing que tem como objetivo construir a análise sintática de dependência do texto. No segundo componente, cada token extraído é classificado sendo de domínio ou não domínio usando caminhos de dependências gerado a partir de árvores de profundidade. Na Detecção de Arestas, para cada par de genes ou produtos de genes (GGP em inglês) presente numa sentença, uma aresta candidata em potencial é gerada entre o termo GGP e o termo predito.

No GETM, na etapa da Análise Semântica, para capturar as variações presentes nos textos, um conjunto de 1000 resumos são anotados manualmente com os termos de domínio. Essas anotações são usadas para ligar padrões léxicos a categorias semânticas. Em seguida, no módulo de Aprendizagem de Máquina, é obtida as informações léxicas de cada sentença que tenha a presença de uma GGP. Na extração de padrões gramaticais, são usadas as mesmas técnicas que no TEES, exceto pelo mapeamento semântico aplicado aos padrões derivados dos grafos de dependência e obtenção de generalização ao substituir cada palavra por sua tag de parte do discurso (POS em inglês).

Na etapa de Detecção de Termo, se a sentença possuir ao menos um GGP, a sua classificação é realizada. Ao classificar uma frase é necessário identificar o termo de domínio exato que está relacionado ao GGP. Devido a isso, um algoritmo de busca baseado em regras, que emprega informações semântica em combinação com

dicionários a partir dos dados de treinamento, em torno do símbolo do gene é aplicado, para a realização da classificação.

(ABACHA; ZWEIGENBAUM, 2011) apresentam o *Medical Texts Annotation and Exploration* (MeTAE), uma plataforma para anotação semântica automática de textos médicos como também a consulta de informação. O MeTAE, segundo o artigo, permite extrair e anotar entidades e relacionamentos médicos e explorar semanticamente as anotações RDF produzidas.

Primeiramente, os textos biomédicos são segmentados pelas ferramentas não especializadas LingPipe e Treetagger-chunker. Em seguida, define-se as entidades médicas, conceitos e tipos semânticos do UMLS com o MetaMap. Por fim, o conjunto de entidades é filtrada usando-se uma lista dos erros mais frequentes e restrição de captura de tipos semânticos para relações objetivos.

Ao fim do procedimento, o corpus de artigos médicos está estruturado em XML. Para cada um desses artigos, os textos são segmentados em frases usando o modelo do projeto LingPipe. Aplica-se MetaMap a cada uma das sentenças, permanecendo para a apenas aquelas que tiverem ao menos um par de conceitos ligados a uma relação R definida. Essa pré-análise semântica, reduz os esforços necessários para construir os padrões em expressões regulares manualmente.

Em (ZHANG; ELHADAD, 2013), a abordagem proposta é constituída de três passos principais: conjunto de termos semente, detecção de borda e classificação de entidade. Para essa abordagem, duas bases de dados foram usadas a i2b2 e a GENIA. A i2b2 é tida como o primeiro corpus médico público para avaliação de sistemas NER.

Na primeira etapa, o objetivo é obter um conjunto de termos sementes para gerar o vetor de assinatura (ou somente assinatura), vetor responsável por manter informações semânticas, numa etapa mais avançada. No passo posterior, a detecção de bordas, o objetivo é capturar do corpus candidatos a entidades. Na última fase, a abordagem adotada para a classificação é que entidades de mesma classe têm contextos e vocabulários semelhantes.

Nessa última fase, podemos considerar duas subfases, a obtenção do vetor de assinatura para cada termo ou classe de entidade. No primeiro caso, o vetor é obtido calculando-se para cada vocábulo do conjunto sua assinatura sobre o termo no documento e no contexto (no artigo é definido como as duas palavras anteriores e posteriores). Para o vetor de classe de entidade, é o somatório de todas as palavras que são termos semente da classe divido pela sua cardinalidade.

A outra subfase, é o cálculo de similaridade, por uso da métrica de cossenos, entre o vetor de assinatura dos termos candidatos do corpus e das classes. Ao vetor de assinatura de uma classe que o vetor de assinatura do termo candidato estiver mais

próximo e ser maior que o limiar definido, é atribuída a classe daquele vetor ao termo. Caso contrário, o termo é excluído de entidades nomeadas reconhecidas.

Para (SEGURA-BEDMAR; MARTÍNEZ; PABLO-SÁNCHEZ, 2010), o motivo pelo qual poucas abordagens conseguem bons resultados na extração de conteúdos biomédicos e a identificação dos relacionamentos entre as entidades é porque estruturas linguísticas como aposição, coordenação e frases complexas, que são muito comuns na literatura biomédica, são desconsideradas.

Em primeiro momento, por não existir corpus adequado, foi-se criado o primeiro corpus anotado para o estudo de interação droga-droga (DDI, sigla em inglês), o banco de dados DrugDDI, que consiste de 579 documentos aleatoriamente selecionados da base DrugBank.

Com o banco de dados criado, os textos são processados pelo MMTx para a realização de processos de segmentação de sentenças, tokenização, POS-tagging, chunking e ligações de frases com conceitos do UMLS. Em seguida, uma abordagem híbrida é adotada, no qual combina a execução de um parser e a implementação de padrões léxicos para capturas de entidades.

De início, ataca-se os problemas comuns em textos biomédicos. Para resolver isso, é implementado um conjunto de padrões que implementam regras sintáticas com o objetivo de simplificar frases, que segmentam longas sentenças em partes menores ao detectar aposições, construções coordenadas e cláusula relativas.

Por fim, apesar da grande diversidade de expressar presentes numa linguagem natural, as construções textuais que indicam a presença de uma DDI são em pequeno número. Por isso, a captura de entidades nomeadas biomédicas usando padrões léxicos se torna uma boa solução. Os padrões são definidos com auxílio de um farmacologista a partir de observações e experiências de construções no qual a presença da relação objetivo esteja presente.

Em (LIMA; ESPINASSE; FREITAS, 2018), é apresentado o OntoILPER, um sistema de extração de informação baseado em ontologia – Ontology-based Information Extraction (OBIE) em inglês. Esse sistema povoa uma base de conhecimento (ontologia de domínio) a partir de ricas anotações linguísticas de um conjunto de documentos de notícias em inglês. Primeiramente, após a sua seleção, as notícias passam pela etapa de pré-processamento usando técnicas de PLN, cujas anotações são persistidas usando um formato baseado em XML. Em seguida, tais anotações linguísticas são usadas para se gerar atributos relacionais que serão usados por um algoritmo de aprendizado de máquina.

Diante o que foi encontrado na literatura científica, o presente trabalho visa contribuir, por meio da proposta de um sistema de extração de relações entre entidades,

que terá como base o uso de ontologias para anotação de informações e extração de relações de documentos da área biomédica e, ao fim, a construção de uma base de dados baseado em grafo para seu armazenamento, busca e inferência.

3.1 Este trabalho em comparação com os trabalhos relacionados

Nessa seção são apresentadas as diferenças entre os trabalhos relacionados e este trabalho.

Em (LIMA; ESPINASSE; FREITAS, 2018) é realizado o processamento de linguagem natural com aplicação a notícias de texto em inglês. Neste trabalho faremos uso dessas técnicas num corpus biomédico, o IEPA.

Em (MCCRAY; SRINIVASAN; BROWNE, 1994) e (RINDFLESCH; RAJAN; HUNTER, 2000) é usado para detecção de termos biomédicos em frases em linguagem natural, técnicas de PLN em nível léxico usando de solução específica, o SPECIALIST. (RINDFLESCH; RAJAN; HUNTER, 2000) em fase posterior, aplica o MetaMap que associa a cada termo um significado no Metathesaurus. Neste trabalho realizamos técnicas de PLN até o nível sintático com uso de ferramentas genéricas, não específicas ao problema biomédico, afim de verificar se o uso de ferramentas não específicas comprometem os resultados.

Em (LANDEGHEM et al., 2012) , (ABACHA; ZWEIGENBAUM, 2011) e (SEGURA-BEDMAR; MARTÍNEZ; PABLO-SÁNCHEZ, 2010) são usados após aplicação de técnicas de PLN até o nível semântico, a extração da informação biomédica com uso de expressões regulares (regex). Em (ABACHA; ZWEIGENBAUM, 2011) é explicitamente descrito que a geração dessas regras foi feita manualmente. Neste trabalho fazemos uso de algoritmos de aprendizado supervisionado para a classificação, que realizam a inferência para a extração dos termos biomédicos a partir dos dados fornecidos, sem que seja necessário a construção de regras com auxílio direto de especialistas ou de forma manual.

Em (ZHANG; ELHADAD, 2013), para a realização da classificação, são usados vetores construídos da análise do conjunto de termos do corpus com aplicação de cosseno para obtenção da similaridade entre o termo e as classes, afim de definir a categoria do termo. Neste trabalho, os vetores serão construídos usando técnicas de *graph embedding* e aplicados a algoritmos de classificação supervisionados de grande uso na literatura.

4 Proposta

Este capítulo trata da proposta sugerida para esse trabalho, que tem como objetivo principal aplicar técnicas de *graph embedding* para classificar frases em linguagem natural. Como campo escolhido, foi definido o campo biomédico, no qual tem a classificação de entidades da área biomédica um dos pontos-chave. A proposta desse trabalho gira em torno de apresentar uma comparação na classificação de entidades nomeadas usando diferente algoritmos de *graph embedding* e classificadores. A abordagem de usar *graph embedding* em entidades nomeadas do campo biomédico é o diferencial dessa proposta. Outro ponto é determinar uma combinação de algoritmos de *embedding* e classificação que otimizem os resultados de classificação, e consequentemente, o reconhecimento de entidade nomeada.

4.1 Metodologia

Nesta seção, detalharemos cada uma das partes da metodologia proposta nesse trabalho indicada na Figura 9.

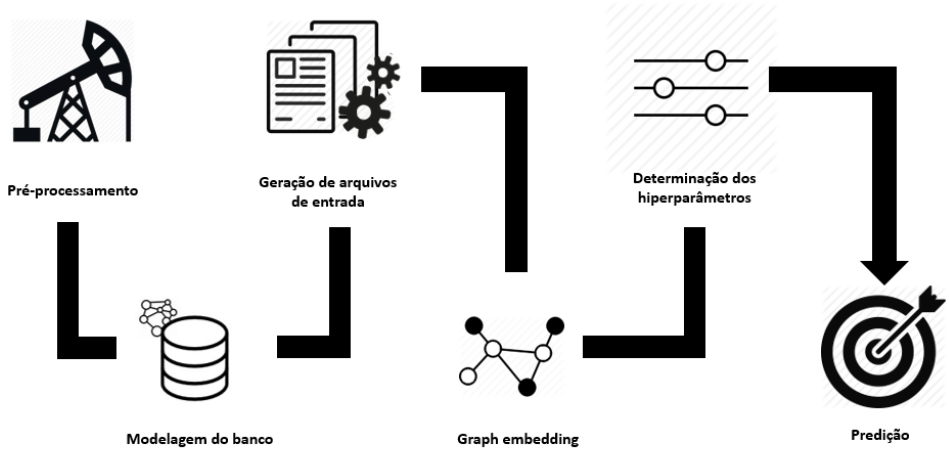

Figura 9 – Metodologia proposta para esse trabalho

Tal metodologia é composta das etapas de:

1. **pré-processamento**, execução de tarefas de PLN para extração de informações sobre os objetos de estudo:
2. **modelagem**, remoção de informações redundantes, carregamento dos dados no BDG em Neo4J, definição das representações da base original;

3. **geração dos arquivos de entrada para o embedding**, criação de egdelist para cada um das representações para ser lida pelos algoritmos de embedding;
4. **graph embedding**, aplicação dos algoritmos de embeddings às representações definidas;
5. **determinação de hiperparâmetros**, escolha das melhores configurações a cada arquivo de *embedding* para cada algoritmo de classificação na etapa de treinamento;
6. **predição**, validação dos resultados obtidos no treinamento usando-se as melhores configurações aplicada ao conjunto de teste (exemplos ainda não visto pelo algoritmo de classificação).

4.1.1 Pré-processamento

Com o corpus de resumos de artigos científicos biomédicos em inglês definido, partiu-se para a extração de dados dos *tokens* que o constitui. Com isso, os textos do corpus passaram pelo Processamento de Linguagem Natural (PLN) (LIMA; ESPINASSE; FREITAS, 2018), dentre elas:

- a **tokenização** Subdivisão da base em *token*s; para esse trabalho foi usado o *English PTBTokenizer*, que foi escrito inicialmente de acordo com Penn Treebank, porém, atualmente apresenta opções para personalização. Nenhuma personalização foi usada (CORENLP, 2019c).
- a **stemização/stemming** Processo de redução de palavras à sua base; para esse trabalho foi usada a classe *Stemmer*, que faz implementação do algoritmo de stemming Porter (CORENLP, 2019b).
- a **Part-of-Speech (POS)** Desambiguação da classe de palavra (artigo, substantivo, adjetivo, verbo etc.) para cada *Token*; foi usada a opção padrão de modelo de POS, o *left3words* (CORENLP, 2019a).
- a **extração de informações morfológicas** Como tamanho do *token*, tipo do *Token* (palavra, símbolo ou número) etc.
- a **análise sintática** Conjunto de tarefas que definem a função sintática (sujeito, predicado, agente da passiva, complemento nominal, dependência dos termos) do *Token* na frase.

Na Figura 10, é apresentado o processamento de uma sentença do corpus sob alguma das tarefas de PLN descritas acima.

Amyloid beta protein (25-35) stimulation of phospholipases A, C and D activities of LA-N-2 cells.

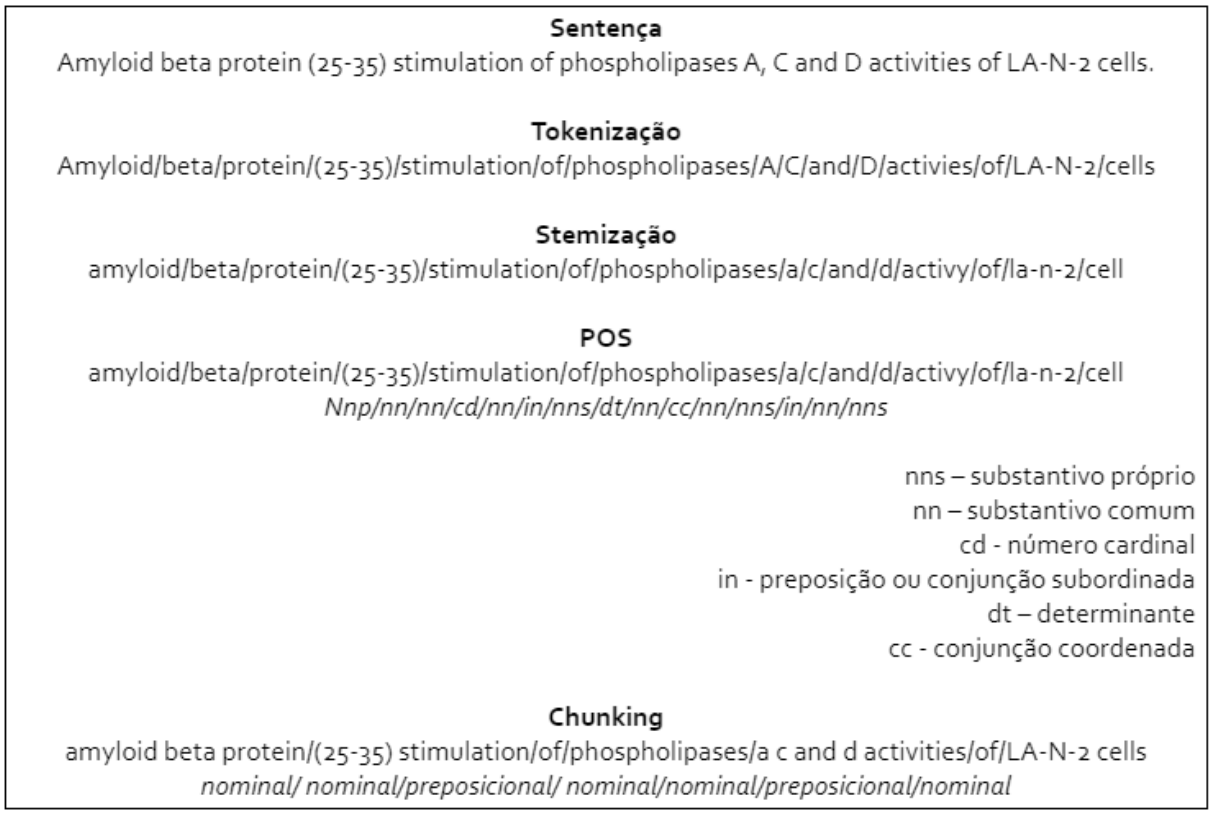

Figura 10 – Aplicação de tarefas de PLN numa sentença do corpus

A extração dessas informações morfológicas e sintáticas sobre o corpus foram transformadas em fatos da linguagem Prolog com o uso da ferramenta Stanford CoreNLP (MANNING et al., 2014). Quatro arquivos foram gerados. O primeiro arquivo contém as propriedades morfológicas e sintáticas obtidas para cada um dos objetos linguísticos considerados, que são os *tokens*, os *chunks* e as *sentenças* (Figura 11). Nos dois outros arquivos estão presentes informações que relacionam dois desses objetos linguísticos. Um deles contém todas as dependências linguísticas existentes entre os tokens de uma sentença (Figura 12). No outro arquivo a indicação do token seguinte a outro (Figura 13). Por fim, no último arquivo, a indicação dos *tokens* presentes na base que são Proteínas (Figura 14). Abaixo está uma sentença presente no corpus IEPA e informações extraídas em cada um dos arquivos usando a ferramenta Stanford CoreNLP.

Oxytocin stimulates IP3 production in dose-dependent fashion as well.

```
st(s_1).
st_hasText(s_1, "Oxytocin stimulates IP3 production in dose-dependent fashion as well .").
st_hasVoice(s_1, ative).
st_hasChunks(s_1, [ck_1, ck_2, ck_3, ck_4, ck_5, ck_6] ).
st_hasChunk(s_1, ck_1).
...
st_hasChunk(s_1, ck_6).

chunk(ck_1).
ck_hasTokenList(ck_1, [t_1] ).
ck_hasType(ck_1, np).
ck_has_tokens(ck_1, t_1).
ck_hasHead(ck_1, t_1).
ck_posRelPred(ck_1, -1).

token(t_1).
t_stem(t_1, 'oxytocin').
t_pos(t_1, nn).
t_type(t_1, word).
t_ck_tag_ot(t_1, b-np).
...
t_bigPosAft(t_1, vbz-nn).
t_trigPosAft(t_1, vbz-nn-nn).
...
```

Figura 11 – Informações extraídas sobre uma das sentenças e seus chunks e tokens

```
% Sentence 1
t_hasDep(nsubj, t_2, t_1).
t_hasDep(dobj, t_2, t_4).
t_hasDep(advmod, t_2, t_9).
t_hasDep(nn, t_4, t_3).
t_hasDep(prep_in, t_4, t_7).
t_hasDep(amod, t_7, t_6).
t_hasDep(advmod, t_9, t_8).
t_root(t_2).
```

Figura 12 – Informações extraídas sobre as dependências presentes numa sentença

```
% Sentence 1
t_next(t_1, t_2).
t_next(t_2, t_3).
t_next(t_3, t_4).
t_next(t_4, t_5).
t_next(t_5, t_6).
t_next(t_6, t_7).
t_next(t_7, t_8).
t_next(t_8, t_9).
```

Figura 13 – Informações extraídas sobre o próximo token em relação a outro token

```
t_class(t_1, protein).
t_class(t_10000, protein).
t_class(t_10027, protein).
t_class(t_10065, protein).
t_class(t_10107, protein).
t_class(t_10112, protein).
t_class(t_10117, protein).
```

Figura 14 – Informações extraídas sobre os tokens que são proteínas

4.1.2 Modelagem

A partir daí, foi dado início à modelagem do *dataset* para uma base de dados baseado em grafo. Em primeiro momento foi identificado do conjunto de fatos em Prolog quais detinham informação para serem usados no problema. Identificou-se que os objetos de tipo sentença e chunks continham poucas informações ou eram redundantes em comparação com as já existentes em *Token*. Devido a isso, informações extraídas de chunks e sentenças foram descartadas. Para os *Token*s, todos os atributos existente foram usados.

Com a seleção dos objetos a constituírem a base, foi dado prosseguimento à modelagem, com a fase de transformação dos dados. Essa fase converte cada fato presente no arquivo Prolog numa base de dados baseado em grafo. Para a fase de transformação dos resultados obtidos após utilização de PLN ao corpus IEPA foi necessário o desenvolvimento um programa em Java com essa finalidade. O programa consiste na aplicação de duas etapas. A primeira dessas etapas é a leitura dos fatos que compunham a base gerada para a criação do banco de dados baseado em grafo na ferramenta NEO4J. Para efetuar a comunicação do programa em Java e o servidor local do banco para geração dos nós e dos relacionamentos a partir dos arquivos Prolog foi usado o *driver* NEO4J para Java. Dos arquivos, foram tomados somente os fatos extraídos para os *Token*s, descartando as *chunks* e sentenças, por motivos já explicitados. Na etapa seguinte, para cada linha dos arquivos que atendesse aos critérios definidos anteriormente, o fato em Prolog era transformado para um comando na linguagem usada para execução de *queries* no NEO4J, a Cypher, sendo logo prontamente executada.

Uma abordagem definida para a construção da base de dados foi a transformação dos atributos dos *Token*s em nós do grafo diferentemente do comum, que é a transformação desses fatos em Prolog para atributos do próprio *Token*. Isso decorre pois em diversos algoritmos de *embedding* sobre nós, como DeepWalk ou node2vec, atributos em nós são ignorados. Para que as informações extraídas sobre os *Token*s não sejam descartadas, a transformação deles em nós se faz necessária. Com isso,

cada atributo distinto é criado como um nó com uma seta de relacionamento partindo dele para o seu *Token*. A cada novo *Token* que tenha esse atributo como valor, um novo relacionamento é criado. Sendo dessa forma, valores de *Token* não são ignorados.

Após o carregamento da base de dados no NEO4J foi possível obter estatísticas sobre o banco. Ao todo são 17.621 nós, sendo 4.327 nós do tipo *AttrToken*, que são nós que trazem informações sobre *Token*s, e, portanto, 13.294 nós do tipo *Token*. Desses últimos, apenas 1.115 são *Token* classificados como proteínas.

Estatística	**Valor**
Número de nós	17.621
Número de nós AttrToken	4.327
Número de nós Token	13.294
Número de nós Token do tipo Proteína	1.115

Tabela 2 – Estatística sobre os nós do grafo construído a partir do corpus IEPA

Os relacionamentos do grafo são criados a partir da conversão dos fatos em Prolog para consultas em Cypher executadas no Neo4J. Para cada propriedade sobre um *Token*, é gerado um tipo diferente de relacionamento. Então um *Token t_102* que tenha uma propriedade *t_pos* = *cc* e *t_type* = *word* gerará dois relacionamentos, um do tipo *T_TYPE* com valor *word* e um outro do tipo *T_POS* com valor *cc*. Para os fatos em Prolog que trazem o próximo *Token* na sequência de uma sentença, é gerado um relacionamento do tipo *NEXT*. E, por fim, para cada fato em Prolog que traga um tipo de dependência diferente é gerado um tipo de relacionamento distinto. Se para o *Token t_104* há uma dependência do tipo *amod* com o *t_103* e uma outra do tipo *prep_of* com o *t_101*, o primeiro relacionamento será do tipo *HAS_DEP_AMOD*, enquanto o segundo relacionamento será do tipo *HAS_DEP_PREP_OF*. Todos os relacionamentos são orientados. Na Figura 15 há um exemplo da modelagem usada para a construção dos relacionamentos.

Com a abordagem de transformar atributos de *Token*s em nós, a quantidade de relacionamentos aumentou bastante, dado que cada atributo é relacionado com o seu *Token* por meio de uma aresta. Dos esperados 22.429 da soma dos relacionamentos de existência de dependência entre *Token*s (*HAS_DEP*) e de um *Token* ser seguido por outro (*NEXT*) passou para 170.259.

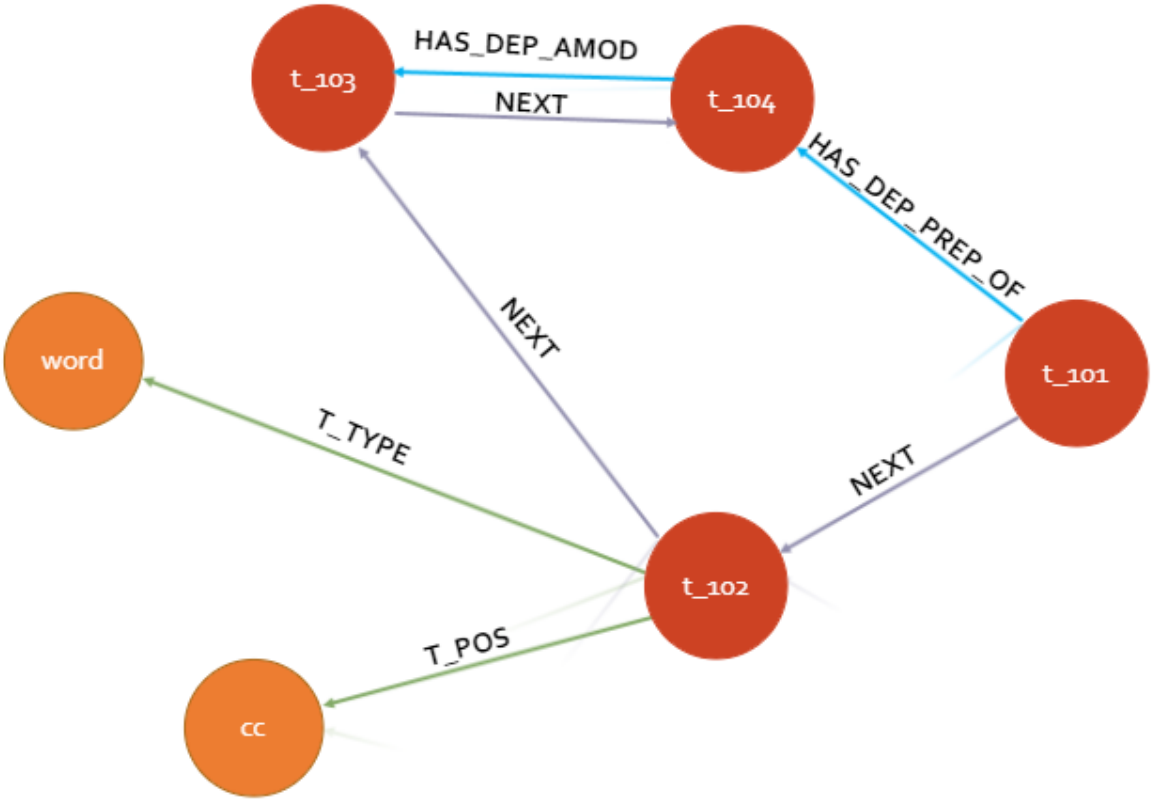

Figura 15 – Exemplo da construção dos relacionamentos do grafo

Estatística	**Valor**	**Tipos de nós envolvidos**
Número de relacionamentos	170.259	-
Número de relacionamentos do tipo Atributo	147.820	Token-AttrToken
Número de relacionamentos do tipo Dependência	9.631	Token-Token
Número de relacionamentos do tipo *NEXT*	12.808	Token-Token

Tabela 3 – Estatística sobre os relacionamentos do grafo construído a partir do corpus IEPA

Com a base de dados modelada, foi gerada quatro diferentes representações. Cada uma dessas representações da base IEPA são *edgelists* a ser usado como entrada para os algoritmos de *embedding*. Essas representações foram geradas a partir de realização de consultas (*queries*) escritas na linguagem Cypher ao banco criado na ferramenta NEO4J. A descrição de cada uma das representações, a quantidade de nós e relacionamentos e *queries* usadas para sua definição.

Modelo	**Consultas sobre o banco**
Primeira BASE	match (m:Token)-[r]->(n:AttrToken) return id(m), id(n), 1
Segunda BASE	match (m:Token)-[r]->(n:AttrToken) return id(m), id(n), 1 match (m:Token)-[r]->(n:Token) where type(r) <> ”NEXT”return id(m), id(n), 1
Terceira BASE	match (m:Token)-[r]->(n:AttrToken) return id(m), id(n), 1 match (m:Token)-[r:NEXT]->(n:Token) return id(m), id(n), 1
Quarta BASE	match (m:Token)-[r]->(n:AttrToken) return id(m), id(n), 1 match (m:Token)-[r]->(n:Token) where type(r) <> ”NEXT”return id(m), id(n), 1 match (m:Token)-[r:NEXT]->(n:Token) return id(m), id(n), 1

Tabela 4 – Consultas realizadas sobre o banco para formação dos modelos

4.1.2.1 MODELO 1 (Primeira BASE)

Esse primeiro modelo é constituído de nós do tipo *Token* e *AttrToken*. Somente as arestas entre os *AttrToken* e *Token*, que são os relacionamentos entre o nó e seus atributos, são incluídos nessa representação (Figura 16). Essa representação ao total possui 17.623 nós e 147.830 relacionamentos.

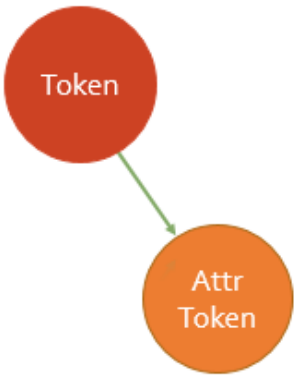

Figura 16 – Representação dos nós e relacionamentos da Primeira BASE

4.1.2.2 MODELO 2 (Segunda BASE)

Nesse segundo modelo, também é constituído dos nós do tipo *Token* e *AttrToken*. Além dos relacionamentos entre *Token* e *AttrToken*, as dependências de um *Token* de outro *Token* (relacionamento *HAS_DEP*) foram incluídas (Figura 17). Essa representação ao total possui 17.623 nós e 157.451 relacionamentos. Um acréscimo de um pouco menos de 10 mil relacionamentos em relação ao modelo anterior.

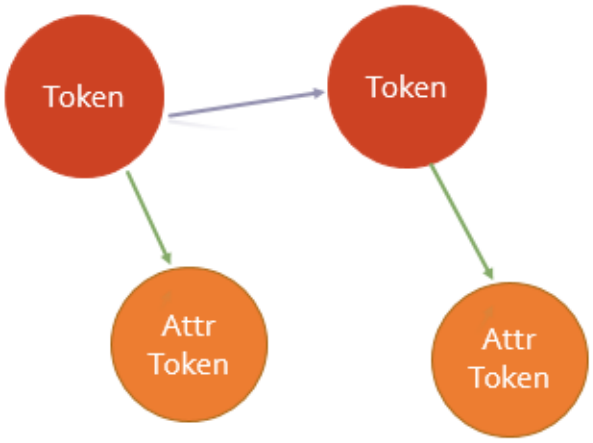

Figura 17 – Representação dos nós e relacionamentos da Segunda BASE

4.1.2.3 MODELO 3 (Terceira BASE)

O terceiro modelo, também constituído dos nós do tipo *Token* e *AttrToken*. Além dos relacionamentos entre *Token* e *AttrToken*, os relacionamentos de indicação de próximo *Token* na sentença (*NEXT*) foram incluídas (Figura 18). Essa representação ao total possui 17.623 nós e 160.638 relacionamentos. Um aumento de um pouco mais de 3 mil relacionamentos ou de quase 13 mil em relação a *Primeira BASE*.

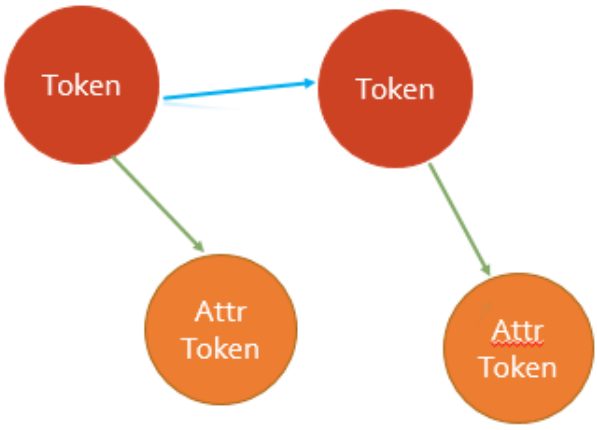

Figura 18 – Representação dos nós e relacionamentos da Terceira BASE

4.1.2.4 MODELO 4 (Quarta BASE)

No quarto modelo, o último, também são constituídos dos nós *Token* e *AttrToken*. Essa representação é mais completa, pois possui todos os relacionamentos definidos na base de dados (Figura 19). Essa representação ao total possui 17.623 nós e 170.259 relacionamentos. Um incremento de quase 10 mil relacionamentos em relação ao modelo anterior ou mais de 20 mil em relação ao primeiro modelo.

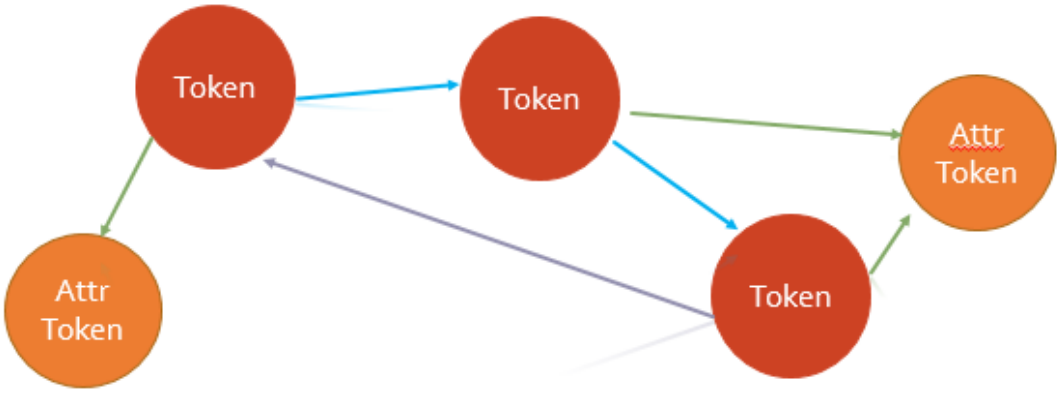

Figura 19 – Representação dos nós e relacionamentos da Quarta BASE

4.1.3 Geração dos arquivos de entrada para o embedding

Antes da etapa de realização do *embedding*, foram gerados os seus arquivos de entrada. Os arquivos de entrada são resultados do retorno da *queries* definidas na subseção anterior para cada modelo da base IEPA. Como formato do arquivo de entrada dos algoritmos de *embedding* selecionados usam o *edgelist*, esta foi a representação naturalmente escolhida. Como a implementação do role2vec exige a passagem do peso w de cada relacionamento existente, foi usado $w = 1$ para indicação de que todos os relacionamentos têm o mesmo grau de importância e manter as mesmas condições postas aos outros dois algoritmos (Figura 20).

4991	1
5283	1
5291	1
5902	1
17645	2
17644	2
17623	2
17595	2
17619	2
17594	2
17593	2
17588	2

4991	1	1
5283	1	1
5291	1	1
5902	1	1
17645	2	1
17644	2	1
17623	2	1
17595	2	1
17619	2	1
17594	2	1
17593	2	1
17588	2	1

Figura 20 – À esquerda representação *edgelist* usada nos algoritmos DeepWalk e node2vec. À direita representação *edgelist* usada no role2vec, com a presença de peso 1 ao fim de cada linha

4.1.4 Graph Embedding

Nessa etapa são gerados os *embeddings* para cada representação da base usando os algoritmos escolhidos. A escolha desses algoritmos se fez por uma série de motivos. O primeiro deles foi pelos bons resultados apresentados nas suas aplicações em problema de classificação (PEROZZI; AL-RFOU; SKIENA, 2014)(GROVER; LESKOVEC, 2016)(AHMED et al., 2018). Um segundo ponto é que cada um desses algoritmos possuem diferenças significativas na forma de leitura dos grafos para geração dos *embeddings*. Enquanto o role2vec consegue fazer distinção entre os tipos de relacionamentos, por conseguir ler grafos heterogêneos, o DeepWalk e node2vec entende todos os relacionamentos como um só. Tal comparativo com as principais características de cada algoritmo se encontra na Tabela 5. Como último ponto, o motivo de escolha do DeepWalk e node2vec, algoritmos clássicos, é verificar se tais algoritmos, mesmo com a existência de soluções mais recentes, como o role2vec, conseguem gerar um bom *embedding*, que ao ser usado na tarefa de classificação tenha um bom desempenho.

Algoritmo	**Ler grafos heterogêneos?**	**Ler grafos rotulado?**	**Parâmetros de controle da caminhada?**
DeepWalk	N	N	N
node2vec	N	N	S
role2vec	S	S	S

Tabela 5 – Estatística sobre o grafo construído a partir do corpus IEPA

Cada um dos algoritmos de *embedding* são configurados com os mesmos conjuntos de parâmetros sobre as representações modeladas. Disso, são resultados os arquivos de *embedding*. Para os algoritmos DeepWalk e node2vec é usado o *Modelo 1*, no qual a primeira linha indica o números nós ”embeddados”e o tamanho do *embedding*, com as linhas seguintes contando com os *embeddings* de cada nó. Para o role2vec, o *Modelo 2*, a primeira linha apresenta as descrições de cada uma das colunas (id do nó e coordenadas), com as próximas linhas trazendo os *embeddings*. As diferenças de representação se faz pelas implementações escolhidas pelos seus autores. Logo abaixo, na Figura 21, apresentamos os dois modelos de arquivos usados.

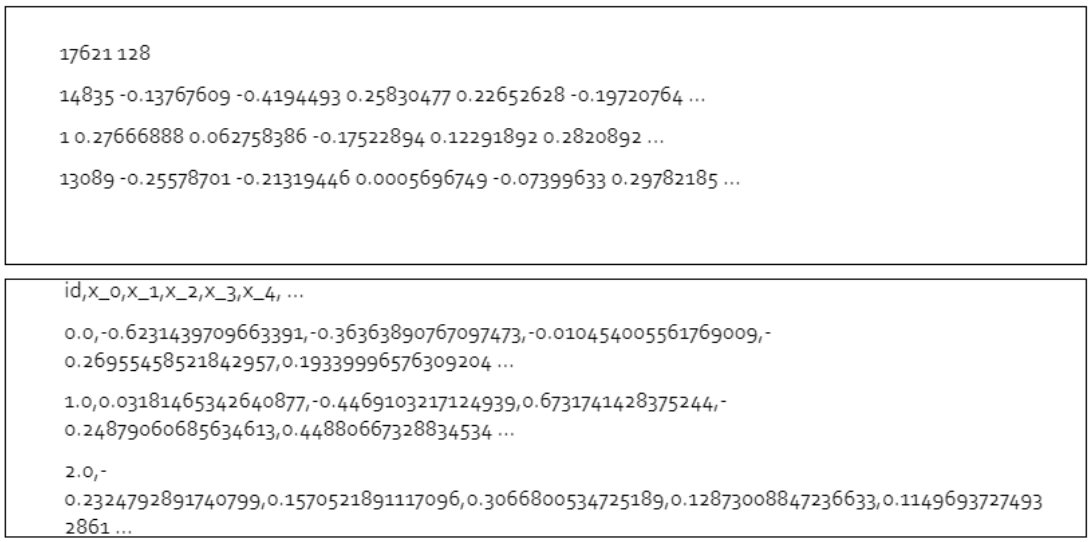

Figura 21 – Acima a representação do *Modelo 1* para um *embedding* para a Primeira BASE. Abaixo a representação usando o *Modelo 2* também para a Primeira BASE

Como observado, os *embeddings* apresentam quantidade de casas decimais diferentes. Devido a isso, foi realizado programaticamente truncamento dos valores de *embedding* de cada coordenada dos vetores para 6 casas decimais.

4.1.5 Determinação de hiperparâmetros

Nessa etapa são usados algoritmos de aprendizagem de máquina para a classificação dos *Token*s em uma das duas classes, *Proteína* e *Não Proteína*. Como temos à nossa disposição informações de cada classe a cada um dos *Token* em nossa base processada, serão usados algoritmos de classificação supervisionados. Ao total, sete algoritmos de classificação foram escolhidos, baseando-se nas indicações de (WU et al., 2008), sendo eles: SVM, árvore de decisão, random forest, XGBOOST, regressão logística, Naive Bayes e redes neurais.

Tendo sido finalizada a aplicação do *graph embedding*, foi realizada a divisão de cada um dos arquivos de *embedding* usando a técnica de validação cruzada.

A validação cruzada é uma técnica de avaliação de modelos de dados por meio do particionamento da entrada em subconjuntos mutualmente exclusivos e uso para validação do modelo com um subconjunto não usado no treinamento. Existem duas abordagens dessa técnica usadas nesse trabalho. A primeira abordagem é a *holdout*. Essa abordagem divide a base em dois conjuntos, um de treinamento, no qual o modelo é ajustado, e o outro de validação, para realização das predições. A outra abordagem, a *k-fold*, subdivide a base em k partes, no qual essas $k-1$ partes são usadas para treinamento e o subconjunto restante para validação. Esse processo é repetido k vezes, com um subconjunto diferente reservado para avaliação (e excluído do treinamento) a

cada vez. Um k grande implica em maior custo computacional (executa k vezes), uma maior variância dos exemplos em cada fold e menor viés (*bias*). Um k pequeno implica em menor custo computacional, menor variância e maior *bias*. O objetivo dessa técnica é verificar se o modelo predito consegue generalizar bem para exemplos não vistos.

Inicialmente, a base foi dividida em duas partes, usando a técnica *holdout*. Para o conjunto de treinamento foi amostrado 70% da base, destinando o restante (30%) para validação. Para o conjunto de treinamento é aplicado internamente a abordagem *k-fold*, sendo $k = 3$. A escolha de $k = 3$ é motivada primeiramente para manutenção de uma taxa destinada para validação durante o treinamento (33%), o *fold* que não será usado na geração do modelo, semelhante à validação desse modelo na etapa após a escolha do melhor hiperparâmetro (aplicando-se os 30% restante da base). Apesar de (KOHAVI, 2001) indicar o uso de $k = 10$, em testes realizados com partes de exemplos da base IEPA, com esse valor, o k se mostrou com desempenho igual ou pior que $k = 3$. Por $k = 10$ apresentar um tempo computacional bem maior que $k = 3$, foi escolhido o último como o valor a ser adotado nesse trabalho.

A fim de reduzir o tempo e custo computacional de execução das validações, foi realizado, no conjunto de treinamento, a escolha dos melhores parâmetros. Essa escolha é feita aplicando-se uma lista prévia de configurações próprias a cada algoritmo e fazendo o treinamento com esses hiperparâmetros. Ou seja, para cada conjunto de configurações para um algoritmo num determinado arquivo de *embedding* é feito o treinamento 3 vezes (por $k = 3$) com finalidade de testar as combinações de configurações e encontrar uma que tenha melhor desempenho para métrica escolhida.

4.1.6 Predição

Após uma extensiva busca dentro de uma lista prévia de parâmetros, na etapa de predição é aplicado a melhor combinação de configuração de um algoritmo para cada arquivo de *embedding*. Somente esse conjunto de configuração é usado para a validação sobre os 30% (exemplos a serem preditos).

A classificação dos *Token*s é feita entre duas categorias: *Proteína* e *Não Proteína*. A primeira é o conceito biomédico no qual estamos interessados em classificar (representado pelo valor 1), enquanto a última são valores *Não Proteína*, que podem ser, na nossa base, *tokens* que não são proteínas como também nós do grafo que apresentam informações extraídas em PLN, os *AttrToken* (representado pelo valor 0).

5 Experimentos

Este capítulo trata dos experimentos realizados nesse trabalho para conseguir responder às contribuições desejadas ao término dele. Foram realizados dois experimentos. O primeiro é referente a descobrir se à medida que se adicionasse mais informação às representações, por meio do acréscimo de relacionamentos, melhoraria o desempenho dos classificadores. O segundo experimento é relativo a encontrar uma melhor combinação para a classificação.

5.1 Corpus

Para a realização desse trabalho, o corpus usado foi o *Interaction Extraction Performance Assessment* (IEPA). Esse corpus é uma coleção de trezentos resumos de textos biomédicos que contém interações entre dois termos recuperados da base MEDLINE. Nestes resumos encontram-se várias menções a nome de proteínas (entidades) estudadas na área biomédica e, para o interesse desse estudo, iremos considerar tais entidades como sendo formadas por um token e ou mais tokens ou palavras. Para a construção desse *dataset* um conjunto de regras foi sugerido por especialistas na área biomédica para a definição das consultas (DING et al., 2002). Uma interação entre dois termos foi definido como uma influência direta ou indireta de um sobre a quantidade ou atividade do outro. Abaixo se encontram exemplos de regras usadas para obtenção dos resumos como também uma frase com a presença de uma entidade biomédica em destaque de um desses resumos.

- A increased B.
- A activated C, and C activated B.
- A-induced increase in B is mediated through C.
- Inhibition of C by A can be blocked by an inhibitor of B.

Regra A increased B: **Oxytocin** *stimulates* **IP3** production in dose-dependent fashion as well.

5.2 Métricas para avaliar o sistema de NER

Para a avaliação de cada um desses algoritmos, foi usado um conjunto de métricas de avaliação usadas comumente pela comunidade de pesquisa NER, são

elas:

Acurácia: É a medida da quantidade de casos de predição corretos dividido pelo número total de predições.

$$\frac{VP + VN}{VP + FN + FP + VN} \tag{5.1}$$

Precisão: A métrica de precisão (ou precision) nos informa quantas amostras previstas são relevantes, ou seja, nossos erros na classificação da amostra como correta se não for verdadeira.

$$\frac{VP}{VP + FP} \tag{5.2}$$

Cobertura: A métrica cobertura (ou recall) mostra quantas amostras relevantes são selecionadas, o que significa quão bem o nosso modelo pode prever todas as amostras de interesse no nosso conjunto de dados.

$$\frac{VP}{VP + VN} \tag{5.3}$$

F1-Score: A métrica F1 é a média harmônica da precisão e da chamada e é calculada como

$$2 * \frac{Precision * Recall}{Precision + Recall} \tag{5.4}$$

Receiver Operating Characteristic Curve (ROC): É a curva de probabilidade, sendo uma representação gráfica que ilustra o desempenho (ou performance) de um sistema classificador binário à medida que o seu limiar de discriminação varia.

Area Under Curve (AUC): A *Area Under Curve (AUC)* assume um valor que varia de $0,0$ até $1,0$ e mostra o quão bom o modelo de predição criado pode distinguir entre duas classes.

5.3 Experimentos

Com as representações da base IEPA e os algoritmos definidos, foi dado início à etapa de experimentação. Ela é composta por duas partes. A primeira parte corresponde à geração de *embeddings* para cada um dos nós pertence ao grafo. A última é o uso desses *embeddings* aplicados a algoritmos de classificação. Para cada uma delas, detalhamos os seus processos.

5.3.1 Geração de Embedding

Para a geração dos *embeddings* foram definidos parâmetros em comum aos algoritmos que pudessem ter valores alternados para a realização da comparação. A

facilidade de comparar os resultados foi um dos motivos pela escolha de algoritmos que somente usam da técnica do *random walk* para geração do *embedding*.

Foram determinados os seguintes cinco parâmetros número de passeios, tamanho do passeio, dimensão do *embedding*, p e q. O número de passeios corresponde à quantidade de vezes que um passeio partirá de um nó. O tamanho do passeio à extensão do passeio em quantidade de nós percorrido. E, por fim, dimensão do *embedding*, que indica o comprimento do vetor de cada *embedding*. Os dois últimos parâmetros são usados como controle da expansão do passeio, sendo p o valor para probabilidade de retornar ao vértice anterior e q o de explorar o próximo nó. Ambos não são usados pelo algoritmo DeepWalk (PEROZZI; AL-RFOU; SKIENA, 2014).

Além desses parâmetros, um outro foi usado no experimento, o tamanho da janela. Esse parâmetro se refere à quantidade de nós usados antes e depois na fase de treinamento na rede, no qual foi usado o valor 5 para todos os experimentos.

Na Tabela 4 se encontra os valores definidos para cada um dos parâmetros. A escolha desses valores foi tomado como referência a partir de resultados para o DeepWalk (PEROZZI; AL-RFOU; SKIENA, 2014), algoritmo mais básico dos três usado nesse trabalho.

Parâmetro	**Valores**	**Explicação**
número de caminhadas	20 e 50	São valores que respectivamente ficam fora e dentro do intervalo recomendado pelos autores do DeepWalk, que é 32 e 64
tamanho da caminhada	20 e 100	São valores extremos para verificar se há o prejuízo de se ter um valor alto ou baixo de nós percorridos
dimensão do embedding	128 e 256	Valores usuais de embedding
p	0.25 e 0.75	
q	0.25 e 0.75	

Tabela 6 – Parâmetros usados no embedding

A combinação dos valores de cada um dos parâmetros definem um experimento a ser aplicado ao algoritmo de *embedding*. Diante disso, temos 8 experimentos DeepWalk e 16 experimentos para cada um dos outros dois algoritmos, node2vec e role2vec, devido à variação de p e q, totalizando 40 experimentos. Essa quantidade é replicada para cada uma das quatro representações da base IEPA,perfazendo 160 experimentos.

5.3.2 Classificação

Na segunda etapa dos experimentos, cada um dos *embeddings* gerados na etapa anterior foram aplicados em todos os algoritmos de classificação. Foram escolhidos sete algoritmos de classificação ao total. Antes da realização da classificação, foi necessária a execução de algumas ações para obtenção de uma melhor predição. Essas ações assim como a predição são descritas logo abaixo.

5.3.2.1 Correção do Desbalanceamento

A correção do desbalanceamento foi a ação tomada para corrigir o grande desequilíbrio existente na quantidade de observações para cada uma das classes na base. Conforme descrito na subseção 5.3.2.1, a porcentagem de nós Proteína corresponde a 8,38% dos *Token*s e 6,32% do conjunto de nós (*Token* e *AttrToken*). Com isso, todos os nós *AttrToken* foram removidos e o número de *Token*s da classes negativa tirada de uma amostragem aleatória restringido à quantidade de exemplos positivos. A escolha por essa metodologia foi definida após tentativas com diferentes valores.

5.3.2.2 Separação da Base

Após a correção do desbalanceamento, a base foi seccionada em duas partes para o processo de validação. A primeira parte, correspondendo a 70% da base balanceada, usada na etapa de treinamento. E o restante, 30% para os testes (validação).

Na etapa de treinamento, foi aplicado a escolha dos melhores parâmetros para cada um dos algoritmos de cada experimento por meio do *GridSearchCV* do pacote *Scikit-Learn* (PEDREGOSA et al., 2011). Para isso, foi usado $k = 3$, dividindo a base de treinamento em 3 folds para aplicação da abordagem k-fold d validação cruzada. Diferentemente da abordagem *holdout*, no qual a base é dividida em duas partes, uma para o conjunto de treinamento e a outra para o conjunto de validação, no *k-fold* os dados são particionados em k conjuntos, onde um conjunto é utilizado para validação e enquanto o restante é utilizado para o treinamento do modelo.

O conjunto de hiperparâmetros usados para cada um dos sete algoritmos se encontra na Tabela 7.

Algoritmo	**Parâmetros**
SVM	**C**: 0,01; 0,1; 1,0 **gamma**: 0,001; 0,0001 **kernel**: linear, rbf
Árvore de Decisão	**max_depth**: 10, 30 **min_samples_leaf**: 2, 3, 4 **min_samples_splitt**: 8, 10, 12
Random Forest	**bootstrap**: True, False **max_depth**: 5, 10 **min_samples_leaf**: 2, 3, 4 **min_samples_split**: 8, 10, 12 **n_estimators**: 10, 25, 50
XGBOOST	**booster**: gbtree, gblinear **learning_rate**: 0,001; 0,01; 0,1 **subsample**: 0,5; 0,8; 1 **colsample_bytree**: 0,5; 0,8; 1
Regressão Logística	**C**: 0,01; 0,1; 1,0 **penalty**: l1, l2 **solver**: liblinear, saga **max_iter**: 100, 200, 300
Naive Bayes	Não há hiperparâmetros
Redes Neurais	**momentum**: 0,5; 0,9 **learning_rate_init**: 0,001; 0,01 **hidden_layer_sizes**: (100, 50, 25), (100,50) **activation**: tanh, relu

Tabela 7 – Hiperparâmetros por algoritmos

5.3.3 Predição

Na etapa de validação, a melhor configuração de parâmetros obtido no *GridSearchCV* para um algoritmo num determinado experimento é usado aplicando-se a base de validação. Após a classificação são tiradas métricas para permitir a comparação entre os resultados dos algoritmos.

5.4 Discussão de Resultados

Nessa seção são discutidos os resultados encontrados após a execução da metologia.

5.4.1 O melhor algoritmo de classificação

Como a proposta-base desse trabalho é a classificação de entidades biomédicas, o primeiro resultado a ser discutido é em relação à definição dos melhores algoritmos de classificação quando usado sobre vetores gerados por algoritmos de *node embedding*. De imediato, os resultados mostram que mesmo com as diferentes representações, temos que a ordem dos melhores algoritmos não foi alterada. Outras considerações quanto à parte de classificação são apresentados abaixo.

As métricas apresentadas pelo algoritmo de Árvore de Decisão foram as de pior desempenho. Em média, desconsiderando as variações dos modelos e algoritmos de *embedding* adotados, conforme a Figura 22 abaixo, mal ultrapassou os 60%.

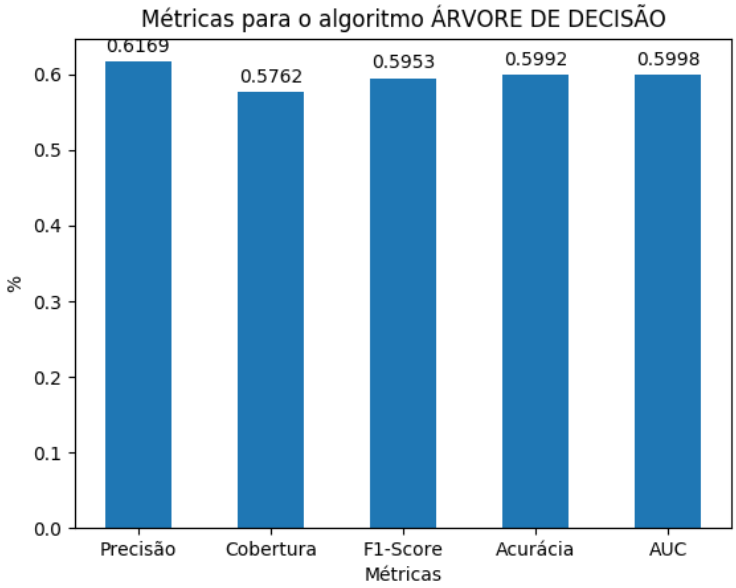

Figura 22 – Média das métricas para o algoritmo Árvore de Decisão

No geral, o algoritmo de Árvore de Decisão foi ligeiramente melhor que uma decisão baseada no lançamento de uma moeda. Inicialmente havia-se cogitado que o motivo dessa baixa performance foi pelo sobreajuste dos parâmetros, a ponto de o algoritmo *decorar* o modelo, algo comum em AD. Contudo, pelos melhores resultados de classificação desse algoritmo, temos que a árvore de decisão não conseguiu ter uma boa *performance* inclusive no treinamento, denotando que os dados a serem aprendidos não foram compreendidos pelo algoritmo, portanto um caso de *underfitting*. O aumento da profundidade da AD pode ser um caminho para melhoria do estimador.

Os outros dois algoritmos que fazem uso de árvores, XGBOOST e Random Forest (RF), ocuparam respectivamente a penúltima e antepenúltima posição no ranking dos melhores classificadores. Apesar do XGBOOST apresentar um AUC ROC maior que a RF, essa diferença foi bastante pequena. Como noutras métricas, como a cobertura, o XGBOOST teve um pior desempenho, a propensão pelo RF pareceu-se mais justa. No XGBOOST, a subamostragem das características do vetor como também dos exemplos foi comum em melhores resultados. Enquanto na RF, contudo, foi percebido que para os melhores resultados, o parâmetro de estimadores com valor 50 foi o escolhido, podendo denotar que os outros valores não foram uma boa escolha para esse problema.

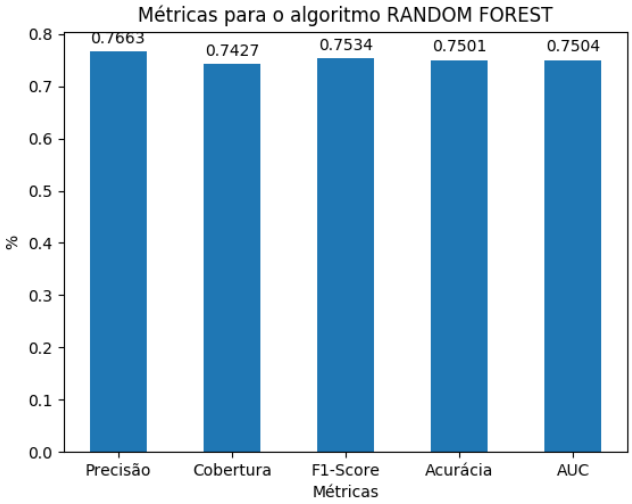

Figura 23 – Média das métricas para o algoritmo Random Forest

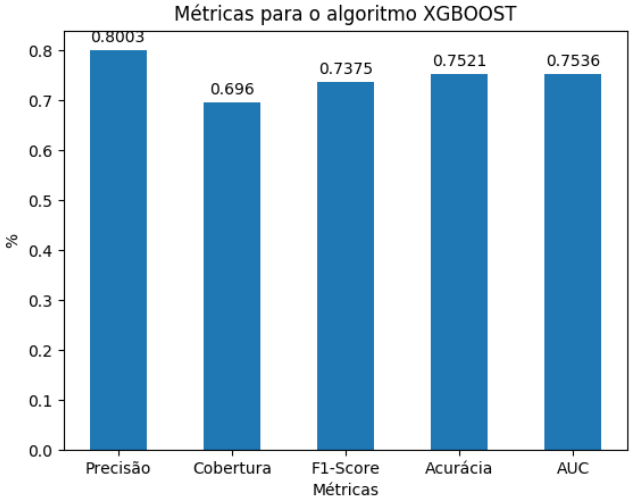

Figura 24 – Média das métricas para o algoritmo XGBOOST

No outro extremo, o algoritmo de Redes Neurais apresentou os melhores valores para as métricas avaliadas (Figura 25). Para elas, os parâmetros mais comuns foram a

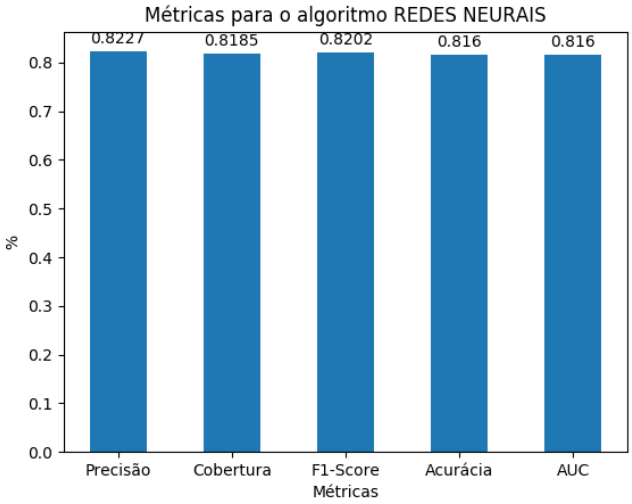

Figura 25 – Média das métricas para o algoritmo Rede Neural

learning_rate_init = $0,01$ e o *momentum* = $0,5$. Com resultados próximos a ele, o SVM se mostra uma boa alternativa. Nesse algoritmo, para os cinco melhores resultados, os parâmetros são os mesmos. Tendo ainda um bom desempenho, temos o Naive Bayes e a Regressão Logística.

5.4.2 O melhor algoritmo de embedding e a melhor representação

Afim de avaliar a influência dos parâmetros de *node embedding* na classificação, foi tomado dos pior e melhor algoritmos, as 10 melhores e as 10 piores configurações de *embedding* para cada uma das representações da base.

Dos 10 melhores resultados de classificação de redes neurais, 8 foram aplicados sobre a *Primeira BASE*, a representação mais simples da base, contendo apenas os relacionamentos de *Token* e *AttrTokken*. Nenhum *embedding* usando role2vec se encontra na lista. Na Tabela 8, é mostrada a melhor configuração de embedding para cada representação da base o com sua respectiva métrica de AUC ROC.

Base	**AUC ROC%**	**Melhor embedding**
Primeira BASE	91,02%	DEEPWALK 50 100 50
Segunda BASE	89,84%	NODE2VEC 50 100 50 Pq
Terceira BASE	89,88%	DEEPWALK 20 100 50
Quarta BASE	89,59%	NODE2VEC 50 100 100 pQ

Tabela 8 – Melhores resultados de classificação para Redes Neurais vs Representação

Dos 10 piores resultados das redes neurais, a base contendo apenas os *tokens* e atributos não apareceram nessa lista. A base mais completa, composta por todos os relacionamentos definidos, a *Quarta BASE*, teve o pior desempenho, ocupando 5

das posições. O destaque negativo fica para o algoritmo de embedding role2vec, que ocupou todo esse ranking. Na Tabela 9 a pior configuração de embedding para cada representação da base o com sua respectiva métrica de AUC ROC.

Base	**AUC ROC%**	**Pior embedding**
Primeira BASE	72,45%	ROLE2VEC 50 200 100 opt3
Segunda BASE	70,76%	ROLE2VEC 20 200 100 opt6
Terceira BASE	67,79%	ROLE2VEC 50 200 100 opt6
Quarta BASE	69,78%	ROLE2VEC 50 200 200 opt6

Tabela 9 – Piores resultados de classificação para Redes Neurais vs Representação

Dos 10 melhores resultados de Árvore de Decisão, 5 deles foram usando a *Quarta BASE*, enquanto apenas um, a *Primeira BASE*. Isso difere completamente da situação encontrada para o melhor algoritmo de classificação. Dentre os algoritmos de *embedding*, o DeepWalk não se encontra na lista de melhores *embedding*. *Embeddings* de tamanho 100 dominaram a lista.Na Tabela 10 a melhor configuração de *embedding* para cada representação da base o com sua respectiva métrica de AUC ROC.

Base	**AUC ROC%**	**Melhor embedding**
Primeira BASE	69,74%	NODE2VEC 50 100 100 pQ
Segunda BASE	60,22%	NODE2VEC 20 100 100 pQ
Terceira BASE	67,26%	NODE2VEC 50 100 50 Pq
Quarta BASE	65,14%	NODE2VEC 50 100 50 Pq

Tabela 10 – Melhores resultados de classificação para Árvore de Decisão vs Representação

Os 10 piores resultados de classificação compõe uma mistura das quatro representações, porém as *Primeira BASE* e *Terceira BASE* aparecem quatro vezes cada uma. O algoritmo de *embedding* node2vec foi o mais frequente. Mais uma vez, difere dos resultados obtidos com o melhor algoritmo de classificação para os demais algoritmos, que costumam seguir o de Redes Neurais. Na Tabela 11 a pior configuração de *embedding* para cada representação da base o com sua respectiva métrica de AUC ROC.

Base	AUC ROC%	Pior embedding
Primeira BASE	53,69%	ROLE2VEC 20 200 200 opt6
Segunda BASE	51,44%	ROLE2VEC 20 200 200 opt6
Terceira BASE	50,86%	ROLE2VEC 50 100 100 opt3
Quarta BASE	51,42%	ROLE2VEC 50 200 200 opt6

Tabela 11 – Piores resultados de classificação para Árvore de Decisão vs Representação

Para identificar quais as melhores combinações de representações e algoritmo de classificação, foi usado um gráfico de *heatmap* para encontrar as melhores subregiões usando as métricas de AUC ROC e F1-Score como referência.

Nas Figuras 26 e 27 podemos notar que a região referente aos algoritmos de rede neural é onde se encontra os maiores valores, seguido com uma margem de dois a mais pontos percentuais pelo SVM, segundo melhor algoritmo. A região em azul somente presente na Árvore de Decisão, mostra o baixo desempenho desse algoritmo. Os demais algoritmos se encontram numa mesma faixa, tendo resultados muito parecidos.

Também percebe-se que para todos os algoritmos, o uso da representação com relacionamento apenas entre os *tokens* e seus atributos foram suficientes para uma boa classificação. Ainda tendo a adição de relacionamentos degradado o resultado da classificação.

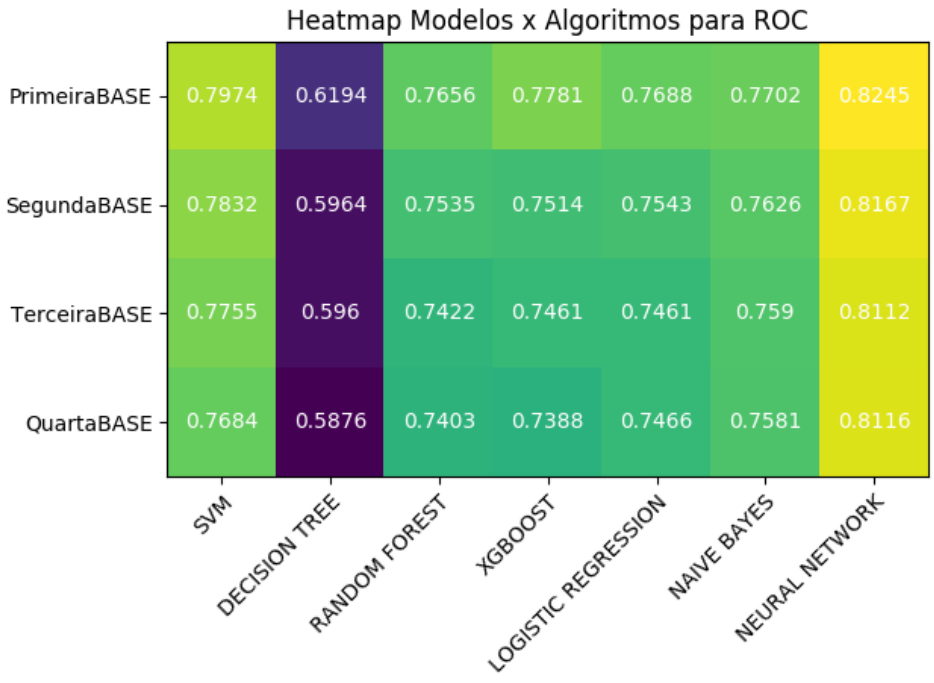

Figura 26 – Heatmap das representações da base pelos algoritmos em relação a ROC

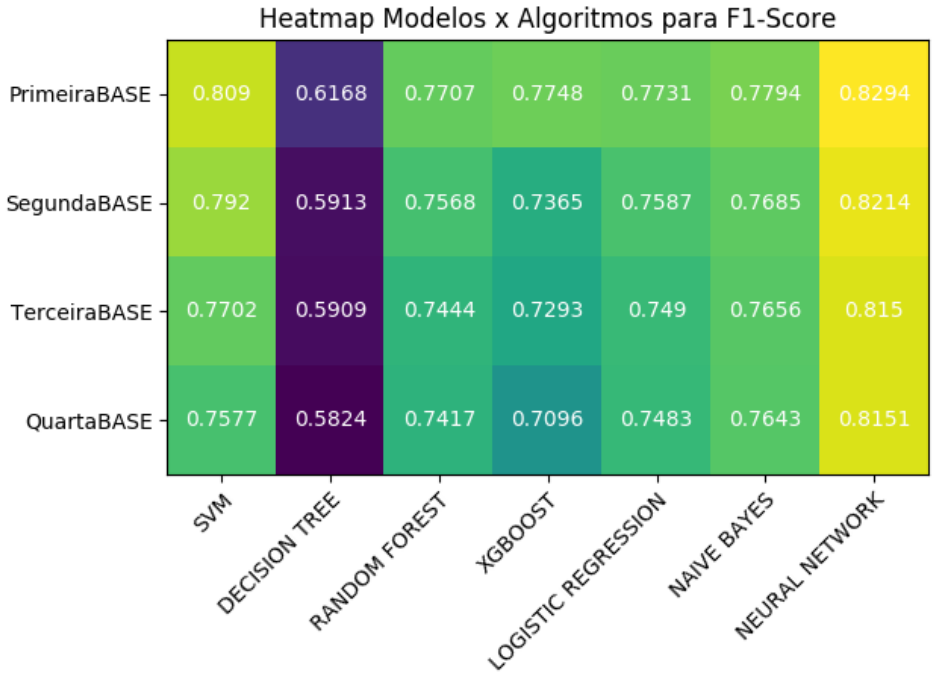

Figura 27 – Heatmap das representações da base pelos algoritmos em relação a F1-Score

Nas duas tabelas a seguir (Tabelas 12 e 13), temos a quantidade de resultados, das 100 melhores classificações, por representação e algoritmo de *embedding*. Delas confirmamos o péssimo desempenho do algoritmo role2vec, figurando com valor 0, e o sobressaimento da Primeira BASE em relação às demais.

Algoritmo de embedding	**Quantidade**
DeepWalk	28
node2vec	72
role2vec	0

Tabela 12 – Quantidade de resultados entre as 100 melhores classificações por algoritmo de embedding

Representação	**Quantidade**
Primeira BASE	31
Segunda BASE	20
Terceira BASE	27
Quarta BASE	22

Tabela 13 – Quantidade de resultados entre as 100 melhores classificações por representação

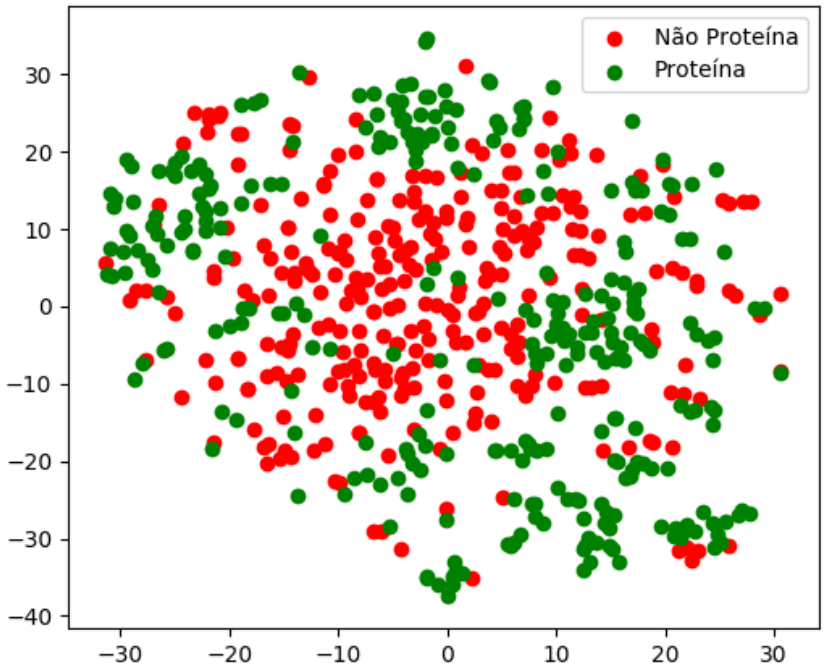

Figura 28 – T-SNE do node2vec

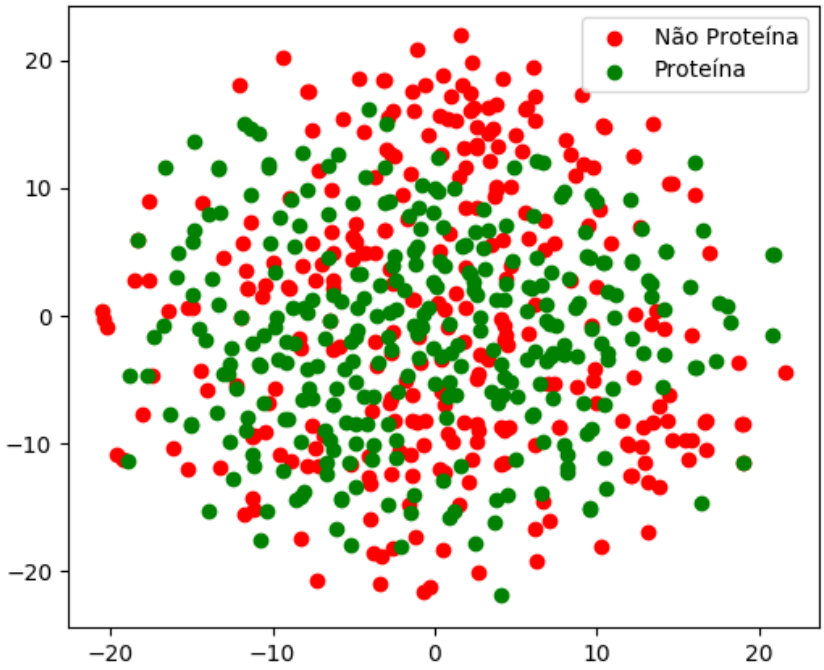

Figura 29 – T-SNE do role2vec

Os dois gráficos (Figuras 28 e 29) acima são gerados a partir da aplicação do algoritmo de *embedding* t-SNE, construído a partir de uma amostra aleatória quase estratificada (variando entre 270 a 330 de cada classe) dos vetores de *embeddings* dos algoritmos node2vec e role2vec respectivamente. Eles ajudam a compreender o motivo do resultado indicado nessa subseção. Para a Figura 28, percebemos que os pontos em vermelho (Não Proteína), conseguem se aglomerar mais centralmente no gráfico, estando os pontos verdes (Proteínas), mas afastados do meio do gráfico. Dessa forma, a plotagem da classe certa fica mais detectável. Porém, na Figura 29, existe uma mistura dos pontos verdes e vermelhos, tendo poucas regiões com alta concentração de pontos de uma única cor. Devido a isso, a detecção do tipo (classe) de cada um dos pontos fica mais difícil, tendo um pior resultado de classificação.

O motivo para esse acontecimento pode ser parcialmente respondido, ou justificado pelo seguinte fato: a modelagem com mais informação não foi útil por introduzir ruído em relação ao modelo de base. E o acréscimo dessas informação junto ao funcionamento do role2vec, impactou severamente seus resultados. O role2vec, como já indicado nesse trabalho, consegue ler grafos heterogêneos, portanto, diferenciando entre os tipos de relacionamentos, diferentemente dos algoritmos DeepWalk e node2vec. Para os relacionamentos de dependência, foi criado um para cada tipo de dependência identificado, conforme Figura 15. Sendo assim, foi gerado 98 tipos de relacionamentos de dependência. Todos esses relacionamentos não conseguiram apresentar uma boa diferenciação entre si, perdendo valor informacional no processo e gerando *embeddings* confusos, como a Figura 29 evidencia. Como os outros dois algoritmos leem esses relacionamentos como de um tipo só, não sofreu com esse problema, tendo bons resultados de classificação. Essa hipótese também explicaria porque as bases com a presença de relacionamentos de dependência tiveram os piores resultados.

Contudo, para confirmação dessa hipótese é necessário realização de outros experimentos a serem desenvolvidos em trabalhos futuros. Esses e outros resultados, assim como os algoritmos usados nesse trabalho podem ser acessados na referência a seguir (GITHUB, 2019).

5.4.2.1 As melhores configurações de embedding

Para a verificação dos melhores configurações de *embedding*, foram selecionados os 100 arquivos de *embeddings* que resultaram nas melhores classificações. Da análise desses arquivos, pôde-se chegar aos melhores valores de *embedding*.

O primeiro parâmetro é o *tamanho do embedding*. Nesse trabalho foi usado dois valores para esse parâmetro, 128 e 256. Dos 100 arquivos, 54 *embeddings* tiveram tamanho 256, com o restante (46), tamanho 128. Não há uma grande diferença na quantidade de dimensões presentes no vetor, apesar da leve inclinação ao uso de

embedding maior, de 256. Contudo, não há um predomínio na quantidade de melhores resultados nem também na ordenação desses 100 casos nem há a apresentação de uma diferença nas métricas muito grandes quando aquele é se sai melhor, para uma redução no tempo de processamento e custo de armazenamento a opção por um *embedding* de tamanho 128 entende-se como melhor opção.

Outro parâmetro é o *comprimento da caminhada*. Para esse trabalho foram usados os valores 20 e 100. Da análise sobre as 100 arquivos, 67 das melhores classificações são com tamanho 100 para esse parâmetro, portanto, 33 para os de tamanho 20. Diferentemente do parâmetro anterior, a escolha pelo uso de caminhada mais longa se mostrou evidente quanto sua eficiência para um melhor resultado frente ao problema.

Para o próximo parâmetro, a *quantidade de caminhadas*, foram usados os valores 20 e 50. Dos 100 arquivos, 57 das melhores classificações são com tamanho 50 para esse parâmetro, 43 para os de tamanho 20. Ainda desse conjunto de resultados, os cinco melhores dele são usando uma quantidade de caminhada. Diante disso, a realização de mais caminhadas no grafo possibilitou um melhor resultado na classificação.

Para os parâmetros de controle p e q foi analisado somente os resultados daqueles que não fossem *embedding* de DeepWalk. Ao total foram 72 resultados que usaram desses parâmetros. Em 38 deles, houve a prevalência do valor p sobre q, ou seja, $p = 0,75$ e $q = 0,25$. Em 34 deles, houve os valores trocados. Diante disso, a preferência por um outro, indicada por um valor maior, não parece mostrar uma grande diferença, contudo, pela maior aparição nos melhores resultados, $p = 0,75$ se mostra uma melhor opção.

Abaixo, está as quatro configurações que estão presentes nas ***10*** melhores classificações. A Tabela 14 confirma as escolhas dos melhores parâmetros feitas nesse trabalho. Pois indica uma preferência de número de caminhadas de 50 em detrimento a 20; um tamanho da caminhada de 100 em detrimento a 20 e uma variação de dimensão e os parâmetros de controle nos melhores resultados.

num_walk	num_length	dimension	p	q
50	100	256	0,75	0,25
50	100	128	0,75	0,25
50	100	128	0,75	0,25
50	100	256	0,75	0,25

Tabela 14 – Parâmetros dos 10 melhores resultados

6 Conclusão

Como apresentado nesse trabalho, a área de estudo de Reconhecimento de Entidade Nomeadas é uma área tradicional das técnicas de Processamento de Linguagem Natural.

Abordagens usando desde modelos estatísticos a baseados em *features* estão presentes em diversos trabalhos; e aplicações em *datasets* específicos como os corpus biomédicos são de grande interesse pela comunidade científica.

A proposta deste trabalho foi de realizar o reconhecimento de entidade nomeadas (de início focado em proteínas) em documentos biomédicos, com o uso de algoritmos de *graph embedding* como intermediário na representação dessas entidades para serem usados por algoritmos de classificação de aprendizado de máquina e apresentar uma baseline para indicação dos melhores algoritmos e seus parâmetros para esse fim.

Esta pesquisa traz fortes evidências experimentais que mostram a efetividade da proposta visto a combinação do algoritmo de *graph embedding* node2vec e redes neurais, que se mostrarem os mais efetivos nos experimentos realizados e discutidos neste trabalho.

Pode-se concluir que, a proposta apresentada neste trabalho apresentou uma abordagem efetiva para a NER e entende que esse trabalho atingiu seus objetivos e contribuiu conforme o esperado.

Limitações e Trabalhos Futuros Apesar dos resultados encorajadores, este trabalho apresenta algumas limitações e seus respectivos trabalhos futuros. A solução proposta para NER se baseia numa etapa prévia de anotação de frases em linguagem natural que leva em conta aspectos tanto léxicos quanto sintáticos. Por outro lado, para uma efetiva anotação e parsers das frases, as ferramentas de processamento de linguagem natural (PLN) devem ter sido treinadas usando datasets específicos para o domínio em questão.

No nosso caso, o pré-processamento linguístico não usou parsers específicos do domínio biomédico. Ficando então como trabalho futuro, a seleção e substituição das ferramentas de PLN genéricas usadas, por outras mais específicas para o domínio biomédico.

Além dos algoritmos de GE usados neste trabalho que priorizam as informações das arestas dos grafos no processo de geração de features caracterizando os vértices, existem outros que também levam em conta, de forma explícita, as informações de atributos dos vértices (attributed graph embedding) (SUN; ZHANG, 2019). Neste sentido,

este trabalho deverá ser estendido no tocante ao estudo de tais algoritmos de GE mais expressivos e na forma de como os grafos devem ser modelados para se tirar o máximo de proveito deles.

Não foi possível fornecer experimentos comparativos com outros sistemas similares, ficando assim como sugestão importante de trabalho futuro. Neste ponto, datasets usados em outros trabalhos deverão ser usadas e o mesmo protocolo experimental deverá ser adotado para que se acha uma comparação justa entre os sistemas NER comparados.

Além disso, outras linhas de trabalho futuro podem ser investigadas, entre elas:

i realizar experimentos e análise sobre variações nas representações de relacionamentos afim de verificar a representação que produz melhores resultados para algoritmos de *embedding* que fazem leitura de grafos heterogêneos;

ii testar a metodologia ora proposta em bases de dados bem maiores para teste de escalabilidade;

iii estender este trabalho, principalmente no tocante aos algoritmos de classificação, para que seja possível também realizar a tarefa mais complexa de *link prediction* que visa identificar pares de entidades que se relacionam de alguma forma pré-definida (ex. interações entre proteínas). Aqui, pode-se usar o mesmo pipeline de processos e até os mesmos algoritmos de GE, porém o algoritmo de classificação deverá também identificar as relações entre elas.

Referências

ABACHA, A. B.; ZWEIGENBAUM, P. Automatic extraction of semantic relations between medical entities: a rule based approach. *Journal of Biomedical Semantics*, v. 2, n. 5, p. S4, Oct 2011. ISSN 2041-1480. Disponível em: <https://doi.org/10.1186/2041-1480-2-S5-S4>. Citado 2 vezes nas páginas 35 e 37.

AHMED, N. K. et al. *Learning Role-based Graph Embeddings*. 2018. Citado 2 vezes nas páginas 29 e 48.

CHEN, T.; GUESTRIN, C. Xgboost. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, ACM Press, 2016. Disponível em: <http://dx.doi.org/10.1145/2939672.2939785>. Citado na página 31.

CORENLP, S. *Stanford - POSTagger*. 2019. Disponível em: <https://stanfordnlp.github.io/CoreNLP/pos.html#options>. Citado na página 39.

CORENLP, S. *Stanford - Stemming*. 2019. Disponível em: <https://github.com/stanfordnlp/CoreNLP/blob/master/src/edu/stanford/nlp/process/Stemmer.java>. Citado na página 39.

CORENLP, S. *Stanford - Tokenization*. 2019. Disponível em: <https://nlp.stanford.edu/software/tokenizer.html#Options>. Citado na página 39.

CRAVEN, M.; KUMLIEN, J. Constructing biological knowledge bases by extracting information from text sources. In: *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*. AAAI Press, 1999. p. 77–86. ISBN 1-57735-083-9. Disponível em: <http://dl.acm.org/citation.cfm?id=645634.663209>. Citado na página 33.

DIMENSõES, . *Overfitting e Underfitting*. 2014. Disponível em: <https://www.3dimensoes.com.br/post/overfitting-e-underfitting>. Citado 2 vezes nas páginas 29 e 30.

DING, J. et al. Mining medline: abstracts, sentences, or phrases? *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, v. 7, p. 326–37, 02 2002. Citado na página 51.

GITHUB. *Repositório de código e resultados do TCC "Graph Embeddings para Node Classification em representação baseada em grafos de frases em linguagem natural"defendido na UFRPE em 2019.2*. 2019. Disponível em: <https://github.com/JNMarcos/tcc_graphembedding_classification_iepa>. Citado na página 63.

GODEC, P. *Graph Embeddings — The Summary*. 2018. Disponível em: <https://towardsdatascience.com/graph-embeddings-the-summary-cc6075aba007>. Citado na página 26.

GOYAL, P.; FERRARA, E. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, Elsevier BV, v. 151, p. 78–94, Jul 2018. ISSN 0950-7051. Disponível em: <http://dx.doi.org/10.1016/j.knosys.2018.03.022>. Citado 2 vezes nas páginas 27 e 28.

GROVER, A.; LESKOVEC, J. node2vec: Scalable feature learning for networks. In: *KDD*. [S.l.: s.n.], 2016. Citado 4 vezes nas páginas 7, 23, 29 e 48.

IME, U. *Fatoração LU*. 2019. Disponível em: <https://www.ufrgs.br/reamat/CalculoNumerico/livro-oct/sdsl-fatoracao_lu.html>. Citado na página 27.

JIANG, J. Information extraction from text. In: ____. *Mining Text Data*. Boston, MA: Springer US, 2012. p. 11–41. ISBN 978-1-4614-3223-4. Disponível em: <https://doi.org/10.1007/978-1-4614-3223-4_2>. Citado 3 vezes nas páginas 13, 21 e 22.

KINGMA, D. P.; WELLING, M. *Auto-Encoding Variational Bayes*. 2013. Citado na página 27.

KOHAVI, R. A study of cross-validation and bootstrap for accuracy estimation and model selection. v. 14, 03 2001. Citado na página 50.

LANDEGHEM, S. V. et al. Semantically linking molecular entities in literature through entity relationships. *BMC Bioinformatics*, v. 13, n. 11, p. S6, 2012. ISSN 1471-2105. Disponível em: <https://doi.org/10.1186/1471-2105-13-S11-S6>. Citado 2 vezes nas páginas 34 e 37.

LIMA, R.; ESPINASSE, B.; FREITAS, F. Ontoilper: an ontology- and inductive logic programming-based system to extract entities and relations from text. *Knowledge and Information Systems*, v. 56, n. 1, p. 223–255, Jul 2018. ISSN 0219-3116. Disponível em: <https://doi.org/10.1007/s10115-017-1108-3>. Citado 3 vezes nas páginas 36, 37 e 39.

MANNING, C. D. et al. The Stanford CoreNLP natural language processing toolkit. In: *Association for Computational Linguistics (ACL) System Demonstrations*. [s.n.], 2014. p. 55–60. Disponível em: <http://www.aclweb.org/anthology/P/P14/P14-5010>. Citado na página 40.

MCCRAY, A. T.; SRINIVASAN, S.; BROWNE, A. C. Lexical methods for managing variation in biomedical terminologies. *Proceedings. Symposium on Computer Applications in Medical Care*, American Medical Informatics Association, p. 235–239, 1994. ISSN 0195-4210. 7949926[pmid]. Disponível em: <https://www.ncbi.nlm.nih.gov/pubmed/7949926>. Citado 2 vezes nas páginas 32 e 37.

NLM. *MEDLINE Fact Sheet*. 2017. Disponível em: <https://www.nlm.nih.gov/bsd/medline.html>. Citado na página 13.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011. Citado na página 54.

PENTEADO, R. R. M. et al. Um estudo sobre bancos de dados em grafos nativos. p. 10, 2014. Citado 2 vezes nas páginas 20 e 21.

PEROZZI, B.; AL-RFOU, R.; SKIENA, S. Deepwalk: Online learning of social representations. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 03 2014. Citado 4 vezes nas páginas 23, 28, 48 e 53.

RIBEIRO, L. F.; SAVERESE, P. H.; FIGUEIREDO, D. R. struc2vec. *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '17*, ACM Press, 2017. Disponível em: <http://dx.doi.org/10.1145/3097983.3098061>. Citado na página 29.

RINDFLESCH, T. C.; RAJAN, J. V.; HUNTER, L. Extracting molecular binding relationships from biomedical text. In: *Proceedings of the Sixth Conference on Applied Natural Language Processing*. Stroudsburg, PA, USA: Association for Computational Linguistics, 2000. (ANLC '00), p. 188–195. Disponível em: <https://doi.org/10.3115/974147.974173>. Citado 2 vezes nas páginas 33 e 37.

SEGURA-BEDMAR, I.; MARTÍNEZ, P.; PABLO-SÁNCHEZ, C. de. Extracting drug-drug interactions from biomedical texts. *BMC Bioinformatics*, v. 11, n. 5, p. P9, 2010. ISSN 1471-2105. Disponível em: <https://doi.org/10.1186/1471-2105-11-S5-P9>. Citado 2 vezes nas páginas 36 e 37.

SUN, G.; ZHANG, X. A novel framework for node/edge attributed graph embedding. In: ____. [S.l.: s.n.], 2019. p. 169–182. ISBN 978-3-030-16141-5. Citado na página 65.

W3C. *RDF 1.1 Concepts and Abstract Syntax*. 2019. Disponível em: <https://www.w3.org/TR/rdf11-concepts/>. Citado na página 21.

WU, X. et al. Top 10 algorithms in data mining. *Knowledge and Information Systems*, v. 14, n. 1, p. 1–37, Jan 2008. ISSN 0219-3116. Disponível em: <https://doi.org/10.1007/s10115-007-0114-2>. Citado na página 49.

ZHANG, S.; ELHADAD, N. Unsupervised biomedical named entity recognition: Experiments with clinical and biological texts. *Journal of Biomedical Informatics*, v. 46, n. 6, p. 1088 – 1098, 2013. ISSN 1532-0464. Special Section: Social Media Environments. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1532046413001196>. Citado 2 vezes nas páginas 35 e 37.