UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE ENGENHARIA ELÉTRICA E ELETRÔNICA

PREDIÇÃO DE MÚLTIPLOS CÓDIGOS CID-10 A PARTIR DE NOTAS CLÍNICAS EM PORTUGUÊS BRASILEIRO

Arthur Deltregia Reys

Orientador: Prof. Danilo Silva, Ph.D.

Florianópolis, 30 de outubro de 2020.

ARTHUR DELTREGIA REYS

PREDIÇÃO DE MÚLTIPLOS CÓDIGOS CID-10 A PARTIR DE NOTAS CLÍNICAS EM PORTUGUÊS BRASILEIRO

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia Elétrica, Departamento de Engenharia Elétrica e Eletrônica da Universidade Federal de Santa Catarina como requisito parcial para obtenção do grau de Bacharel no Curso de Engenharia Elétrica.
Orientador: Prof. Danilo Silva, Ph.D. .

FLORIANÓPOLIS
2020

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Reys, Arthur Deltregia
Predição de múltiplos códigos CID-10 a partir de notas clínicas em português brasileiro / Arthur Deltregia Reys ; orientador, Danilo Silva, 2020.
58 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Engenharia Elétrica, Florianópolis, 2020.

Inclui referências.

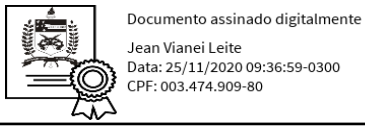
1. Engenharia Elétrica. 2. Codificação CID. 3. Notas clínicas. 4. Processamento de linguagem natural. 5. Redes neurais. I. Silva, Danilo. II. Universidade Federal de Santa Catarina. Graduação em Engenharia Elétrica. III. Título.

Arthur Deltregia Reys

PREDIÇÃO DE MÚLTIPLOS CÓDIGOS CID-10 A PARTIR DE NOTAS CLÍNICAS EM PORTUGUÊS BRASILEIRO

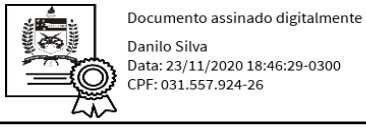
Este Trabalho de Conclusão de Curso foi julgado adequado para a obtenção do título de Bacharel em Engenharia Elétrica e aprovado em sua forma final pelo Curso de Graduação em Engenharia Elétrica.

Florianópolis, 23 de novembro de 2020.



Prof. Jean Vianei Leite, Dr.
Coordenador do Curso

Banca Examinadora:



Prof. Danilo Silva, Ph.D.
Orientador
Universidade Federal de Santa Catarina

Documento assinado digitalmente

Marcio Holsbach Costa
Data: 23/11/2020 14:05:53-0300
CPF: 572.170.680-53

Prof. Márcio Holsbach Costa, Dr.
Universidade Federal de Santa Catarina

Documento assinado digitalmente

Ricardo Faria Giglio
Data: 23/11/2020 10:25:52-0300
CPF: 304.078.978-36

Prof. Ricardo Faria Giglio, Ph.D.
Universidade Federal de Santa Catarina

Agradecimentos

Agradeço primeiramente aos meus pais e toda minha família, pelo seu inestimável apoio em todos os momentos que culminaram neste trabalho.

Agradeço também imensamente ao meu orientador, professor Danilo Silva, pelo seu suporte no decorrer de toda elaboração deste Trabalho de Conclusão de Curso.

Agradeço também aos professores Márcio Costa e Ricardo Giglio, os quais me honram com sua presença na banca examinadora.

Agradeço à Universidade Federal de Santa Catarina, especialmente a todos os professores que fizeram parte da minha graduação.

Finalmente, agradeço a todos os colegas da 3778, pelo apoio no desenvolvimento deste trabalho.

RESUMO

A rotulação de registros clínicos eletrônicos com códigos CID é uma tarefa manual, cara e lenta. A atribuição de tais códigos é, entretanto, uma importante tarefa para fins de cobrança e organização de bancos de dados. Embora muitos trabalhos tenham estudado o problema de atribuição automática de códigos CID a partir de texto livre utilizando técnicas de aprendizado de máquina, grande parte utiliza registros em língua inglesa, especialmente provenientes do conjunto de dados público MIMIC-III. Este trabalho apresenta resultados para um conjunto de dados com notas clínicas em português brasileiro. São desenvolvidos e otimizados um modelo de Regressão Logística, uma Rede Neural Convolucional (CNN), uma Rede Neural Recorrente do tipo GRU (*Gated Recurrent Unit*) e uma CNN com Atenção (CNN-Att), para predição de códigos CID de diagnóstico. São também apresentados resultados dos mesmos modelos para o conjunto de dados MIMIC-III, os quais superam trabalhos anteriores entre modelos das mesmas famílias, além do estado da arte. Comparado ao MIMIC-III, o conjunto de dados em português brasileiro contém um número muito menor de palavras por documento, quando apenas sumários de alta são utilizados. Experimentos a partir da concatenação de documentos adicionais disponíveis nesse conjunto demonstram um grande incremento em performance. O modelo CNN-Att obtém os melhores resultados em ambos os conjuntos de dados, atingindo uma F1 em ponderação micro de 0.537 no MIMIC-III e 0.485 no conjunto de dados em português com documentos adicionais.

Palavras-chave: Codificação CID. Notas clínicas. Processamento de linguagem natural. Classificação multi-rótulo. Redes neurais.

ABSTRACT

ICD coding from electronic clinical records is a manual, expensive and time-consuming process. Code assignment is, however, an important task for billing purposes and database organization. While many works have studied the problem of automated ICD coding from free text using machine learning techniques, most use records in the English language, especially from the MIMIC-III public dataset. This work presents results for a dataset with Brazilian Portuguese clinical notes. A Logistic Regression model, a Convolutional Neural Network (CNN), a Gated Recurrent Unit Neural Network and a CNN with Attention (CNN-Att) are developed and optimized for prediction of diagnosis ICD codes. Results are also reported for the MIMIC-III dataset, which outperform previous work among models of the same families, as well as the state of the art. Compared to MIMIC-III, the Brazilian Portuguese dataset contains far fewer words per document, when only discharge summaries are used. Experiments concatenating additional documents available in this dataset achieve a great boost in performance. The CNN-Att model achieves the best results on both datasets, with micro-averaged F1 score of 0.537 on MIMIC-III and 0.485 on the Brazilian-Portuguese dataset with additional documents.

Keywords: ICD coding. Clinical notes. Natural language processing. Multi-label classification. Neural networks.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Ao longo da estadia de um paciente em um hospital, uma série de documentos é redigida sobre sua situação, incluindo descrições de sintomas, evolução clínica, diagnósticos e histórico médico. Após sua alta, profissionais codificadores analisam a documentação do paciente e atribuem à sua estadia uma lista de códigos baseados na Classificação Internacional de Doenças (CID), um sistema padronizado mantido pela Organização Mundial de Saúde [1, 2]. Tais códigos identificam uma variedade de informações clínicas, as quais são úteis para fins de cobrança, comunicação com planos de saúde e organização de bancos de dados para pesquisa e análises estatísticas [3].

Atualmente, o processo de codificação de códigos CID é realizado manualmente por codificadores especificamente treinados para essa função. A alta granularidade do sistema de codificação torna diferenças entre códigos similares bastante sutis. Além disso, grande parte da informação contida em registros clínicos apresenta-se em texto livre não estruturado, contendo linguagem técnica específica do meio médico, além de conter abreviações, termos e siglas ambíguas e erros de digitação. Em conjunto, todos esses fatores tornam o processo de rotulação de códigos CID ineficiente, lento, caro e suscetível a erros.

Em consideração a este cenário, uma solução capaz de auxiliar o trabalho de codificadores torna-se bastante desejável. De fato, há mais de duas décadas diversos trabalhos têm se voltado ao estudo dessa tarefa [4], em diferentes abordagens. Recentemente, modelos baseados em Processamento de Linguagem Natural (NLP) com redes neurais avançadas têm mostrado avanços significativos em performance sobre este problema [5, 6].

É necessário ressaltar, no entanto, que a grande maioria de trabalhos na área debruça-se sobre dados em língua inglesa. Estudos em língua portuguesa são raros [7–11]. Exceto por [11], o qual estuda a codificação de códigos CID relacionados a causa de morte em certificados de óbito e [9], que foca em predição de códigos relacionados a oncologia, os demais trabalhos utilizam bases pequenas de dados para predição de conjuntos limitados de códigos. Além disso, nenhum disponibiliza comparações de seus modelos com conjuntos de dados acessíveis.

Este Trabalho de Conclusão de Curso é pautado pelo problema de atribuição automática de múltiplos códigos CID de diagnóstico à

estadia de um paciente. Os dados de entrada são registros médicos eletrônicos em texto livre. Objetiva-se atribuir a cada atendimento médico, a partir de tais notas clínicas, um conjunto de códigos CID, dentre os milhares disponíveis no conjunto de dados. Dados já rotulados por profissionais codificadores foram utilizados para treinamento e avaliação de modelos computacionais, compondo um problema de classificação multi-rótulo com treinamento supervisionado a partir de processamento de linguagem natural sobre notas clínicas.

Especificamente, são desenvolvidos e comparados modelos de Regressão Logística (LR), Rede Neural Convolucional (CNN), Rede Neural Recorrente com *Gated Recurrent Units* (GRU) e Rede Neural Convolucional com mecanismo de Atenção (CNN-Att). O estudo de caso utiliza dados do Hospital Sírio-Libanês, de São Paulo, onde o modelo de melhor performance apresentado por este trabalho será implementado para auxiliar o processo de rotulação manual. Adicionalmente, reportam-se também resultados referentes ao conjunto de dados publicamente acessível MIMIC-III [12, 13], no qual tais modelos superam trabalhos anteriores em modelos das mesma famílias e o atual estado da arte.[1] O estudo apresentado neste trabalho foi publicado em forma de artigo[2] na 9ª Conferência Brasileira de Sistemas Inteligentes (BRACIS 2020).

1.1 OBJETIVOS

1.1.1 Objetivos Gerais

O objetivo principal deste trabalho é desenvolver e avaliar um modelo computacional capaz de realizar satisfatoriamente a predição de múltiplos códigos CID-10 a partir de texto livre em língua portuguesa proveniente de notas clínicas.

1.1.2 Objetivos Específicos

- Estudar diferentes modelos de aprendizado de máquina e de redes neurais.

[1] O código para reprodução de resultados no MIMIC-III está disponível em https://github.com/3778/icd-prediction-mimic.

[2] Artigo disponível em https://arxiv.org/abs/2008.01515.

- Replicar ou superar resultados de modelos das mesmas famílias disponíveis na literatura, treinados com MIMIC-III.
- Estudar o desempenho de diferentes tipos de documentos disponíveis no conjunto de dados do estudo de caso.
- Alcançar resultados no conjunto de dados do estudo de caso tal que o modelo de melhor performance possa ser utilizado em auxílio ao processo de rotulação manual que ocorre no Hospital Sírio Libanês.

2 FUNDAMENTAÇÃO TEÓRICA

Esta seção traz um compilado de trabalhos anteriores que estudaram problemas similares, em conjuntos de dados em língua inglesa e portuguesa. Além disso, faz-se uma revisão teórica das abordagens utilizadas neste trabalho.

2.1 TRABALHOS ANTERIORES

Devido à abrangência da tarefa de codificação de códigos CID e o grande desbalanceamento entre diferentes códigos, pesquisadores frequentemente refinam suas abordagens a um contexto específico. Enquanto alguns trabalhos, tal como este, consideram apenas códigos CID de diagnóstico [14], outros utilizam todos os códigos disponíveis [15], ou ainda conjuntos limitados a determinados escopos [15] [9]. Para alimentar os diversos modelos, grande parte dos trabalhos utiliza sumários de alta, dado que este tipo de documento condensa a maior parte da informação sobre a estadia de um paciente [5]. Entretanto, [15], [11] e [9] realizaram estudos utilizando outros tipos de documentos.

Entre as diversas abordagens utilizadas, a estrutura do sistema de codificação CID é utilizada em [16] e [17] para desenvolver abordagens hierárquicas para assistir predições. Um método baseado em coocorrência de códigos CID é proposto em [18]. Já em [19], sobreposições entre palavras nas descrições de códigos CID e palavras em documentos compõem um método baseado em regra. Com maior predominância, trabalhos utilizam modelos de aprendizado de máquina, tais como SVM (*Support Vector Machine*) [17], Naive-Bayes [20, 21] e kNN (*k-Nearest-Neighbors*) [22].

Mais recentemente, Redes Neurais Convolucionais (CNN) têm sido amplamente utilizadas na literatura, alcançando bons resultados [5, 14, 23]. A vantagem desse tipo de arquitetura sobre modelos mais tradicionais de aprendizado de máquina (tais como LR e SVM) é sua capacidade de capturar atributos contextuais locais [14]. Redes Neurais Recorrentes também têm sido empregadas devido a sua habilidade de associar informações em contextos mais amplos que em redes CNN [16, 24, 25]. Em particular, redes recorrentes constituídas por unidades LSTM (*Long Short-term Memroy*) e GRU (*Gated Recurrent Units*) capturam informação contida em extensas janelas contextuais.

Essas abordagens têm obtido melhorias sobre modelos de aprendizado de máquina mais antigos, uma vez que textos livres geralmente carregam alta complexidade e sua compreensão depende de relações semânticas locais e globais entre termos e sentenças.

Adicionalmente ao uso de redes neurais, modelos inovadores incluem *ensemble* de diferentes arquiteturas [15, 26] e mecanismos de atenção por rótulo [5, 6]. Atenção por rótulo consiste em ponderar uma representação base obtida, por exemplo, por uma rede neural, diferentemente para cada rótulo.

Na tarefa específica de predição de códigos CID de diagnóstico, o trabalho apresentado em [5] detém o atual estado da arte.

2.2 EXTRAÇÃO DE ATRIBUTOS

Treinar um modelo computacional a partir de texto livre requer a aplicação de métodos de extração de atributos, pelos quais um texto pode tornar-se inteligível para um certo modelo. Entre diferentes métodos, alguns codificam documentos inteiros em vetores, levando em conta a ocorrência de palavras em cada documento, mas ignorando sua ordenação. Esta abordagem é conhecida por *Bag-of-Words* (BoW), sendo *Term Frequency – Inverse Document Frequency* (TF-IDF) [27] o método mais popular a utilizá-la. Outros métodos geram representações vetoriais latentes de palavras, como Word2Vec [28], GloVE [29] e FastText [30], o que permite que documentos sejam representados por sequências de vetores de palavras. Métodos mais avançados permitem que uma mesma palavra seja mapeada a diferentes vetores, a partir do contexto em que se insere em um documento. É o caso dos métodos ELMo [31] e BERT [32]. Finalmente, existem ainda métodos a nível de parágrafo e a nível de caracteres [33, 34].

Este trabalho utiliza atributos TF-IDF para o modelo de Regressão Logística e vetores de palavras computados a partir de Word2Vec para as redes neurais. Portanto, detalham-se esses dois métodos de extração de atributos a seguir.

2.2.1 Term Frequency — Inverse Document Frequency

O método TF-IDF [27] tem como objetivo mensurar a importância de uma palavra contida em um documento pertencente a um deter-

minado *corpus*. Tal como no método BoW, um documento é convertido a partir de uma codificação *multi-hot*, que representa as palavras de um dado vocabulário que nele estão presentes. A partir dessa representação base, cada palavra recebe um peso referente a sua importância naquele documento.

Para uma palavra w em um documento d, calcula-se o valor do atributo correspondente:

$$\text{TF-IDF}(d, w) = \frac{n_{w,d}}{n_d} \cdot \left(\log \frac{N}{N_w} + 1 \right), \tag{1}$$

em que $n_{w,d}$ é o número de ocorrências da palavra no documento, n_d é o número total de palavras no documento, N é total de documentos no *corpus* e N_w é o total de documentos que contém a palavra w. Os fatores do produto representam, respectivamente, TF e IDF. A adição de 1 evita que termos ocorrentes em todos os documentos sejam completamente ignorados.

2.2.2 Word2Vec

Word2Vec [28] é um modelo de representação que leva em conta a ordem e contexto de palavras em documentos. A partir de um vocabulário comum a um *corpus*, cada palavra é projetada em um espaço multi-dimensional, o que permite identificar relações interdependentes entre termos, a partir de similaridade de cosseno.

O modelo é composto por uma rede neural com uma única camada oculta. Duas abordagens podem ser utilizadas no treinamento dos vetores, ilustradas na Figura 1: *Continuous Bag-of-Words* (CBoW) e *Skip-gram* (SG). Na primeira, uma palavra é predita a partir de um conjunto limitado de palavras que a precedem e sucedem. Tal conjunto é convertido em atributos BoW para o treinamento da rede, o que causa perda da informação de ordenação local das palavras. Por outro lado, a abordagem SG objetiva prever, a partir de uma palavra, o conjunto de palavras que a rodeia. Nesse caso, a ordem dos termos do contexto influencia a projeção da rede, uma vez que palavras mais próximas recebem pesos maiores.

As representações vetoriais fixas a nível de palavra resultantes do treinamento de Word2Vec podem ser carregadas como uma camada de *embedding* em modelos baseados em rede neural. Uma camada de

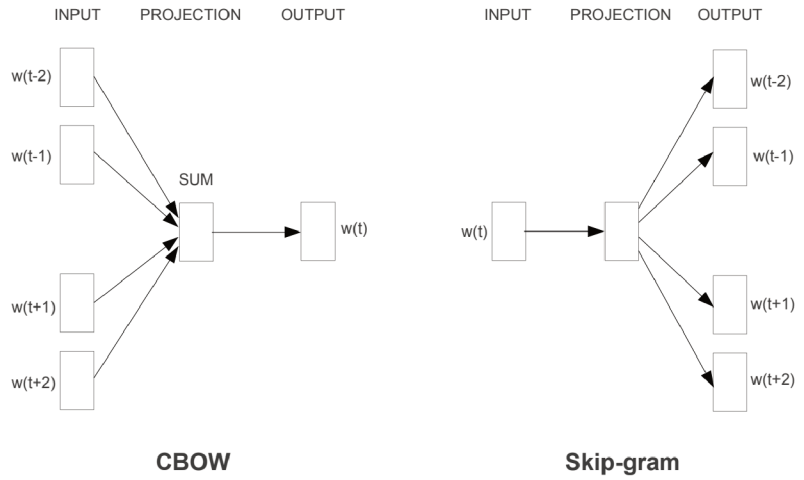


Figura 1: Arquitetura dos métodos CBoW w *Skip-gram*.

Fonte: Pathmind A.I. Wiki (2020).

embeding é um mapeamento entre variáveis discretas de entrada (por exemplo, *tokens* representando palavras) a suas correspondentes representações vetoriais.

2.3 REGRESSÃO LOGÍSTICA

Uma das abordagens mais comuns em problemas de classificação é conhecida por Regressão Logística. Para um problema binário com um vetor de entrada $\mathbf{x} \in \mathbb{R}^{d_v}$ com o tamanho d_v da dimensão definida pelo método de representação que carrega atributos de um certo documento, define-se a saída escalar $\hat{y} \in [0, 1]$ pela equação

$$\hat{y} = \sigma(\mathbf{w}^T \mathbf{x} + b), \tag{2}$$

em que $\mathbf{w}$ é um vetor de pesos treinável, b é um *bias* e σ é a função sigmoide logística, definida por

$$\sigma(z) = \frac{1}{1 + e^{-z}}. \tag{3}$$

Para um problema multi-rótulo, é comum separá-lo em um conjunto de problemas binários, um para cada rótulo, de modo que a Equação 2 escala-se para:

$$\hat{\mathbf{y}} = \sigma(\mathbf{W}^T\mathbf{x} + \mathbf{b}), \tag{4}$$

em que a saída $\hat{\mathbf{y}} \in \mathbb{R}^C$ é um vetor com tamanho igual ao número C de classes, em que cada componente é vinculado a um rótulo diferente, $\mathbf{W} \in \mathbb{R}^{C \times d_v}$ é uma matriz de pesos treinável e $\mathbf{b} \in \mathbb{R}^C$ é um vetor de *bias*. A função σ é aplicada elemento a elemento.

2.4 REDES NEURAIS

Redes neurais são estruturas computacionais organizadas em camadas, cada uma contendo um conjunto de unidades ou nós. Cada unidade mínima em uma camada consiste de uma função parametrizada simples na forma da Equação 5:

$$h_n = g(\mathbf{w}_n^T\mathbf{x} + b_n) \tag{5}$$

em que $\mathbf{x}$ é o vetor de entrada contendo as saídas de todas as unidades de uma camada precedente ou os atributos de entrada de um modelo, e $\mathbf{w}_n$ e b_n são pesos únicos da unidade n, respectivamente um vetor único de ponderação das entradas e um *bias*. A função g é denominada *função de ativação* e consiste de uma operação não-linear, que resulta na saída escalar h daquela unidade.

A grande vantagem de uma rede neural é a possibilidade de organizar essas unidades de inúmeras formas, com o objetivo de aproximar uma função — arbitrária e/ou desconhecida — $f^*(x)$ por uma função $f(x)$ decorrente da combinação de unidades.

Supondo-se que existam amostras constituídas por pares $(\mathbf{x}, \mathbf{y})$, em que $\mathbf{x}$ são entradas ou atributos de entrada e $\mathbf{y}$ são saídas tais que $\mathbf{y} \approx f^*(\mathbf{x})$, é possível realizar um treinamento dos pesos de uma rede neural a partir dos pares conhecidos, de modo a aproximar a função f^*. O treinamento eficiente de redes neurais passa por um algoritmo denominado *Backpropagation*.

2.4.1 Backpropagation

O algoritmo de *Backpropagation* busca minimizar uma determinada função custo $J(\mathbf{y}, \hat{\mathbf{y}})$ que representa a qualidade das predições $\hat{\mathbf{y}}$ da rede frente às saídas verdadeiras $\mathbf{y}$. Com um certo conjunto não-nulo de pesos iniciais, as predições da rede podem ser computadas por propagação das entradas $\mathbf{x}$ (*Foward propagation*) ao longo das unidades e camadas da rede. Os vetores de pesos $\mathbf{W}^{[\ell]}$ (considerando-se incluso o *bias*) de cada camada $\ell \in [L, L-1, ..., 1]$ são então atualizados via gradiente descendente, partindo da camada mais próxima à saída (L) tal que

$$\mathbf{W}^{[\ell]} \leftarrow \mathbf{W}^{[\ell]} - \alpha \frac{\partial J}{\partial \mathbf{W}^{[\ell]}}, \tag{6}$$

em que α é a taxa de aprendizado. Para uma camada ℓ qualquer, ao invés de calcular diretamente o gradiente da função custo em relação aos pesos $\mathbf{W}^{[\ell]}$, é possível utilizar a regra da cadeia sucessivamente de L até ℓ, propagando o gradiente até obter a atualização de pesos da respectiva camada. Essa possibilidade é o que torna o algoritmo *Backpropagation* bastante eficiente.

2.4.2 Redes Neurais Convolucionais

A simples combinação de unidades em camadas, em que cada unidade de uma camada subsequente recebe como entrada um vetor contendo a saída de todas as unidades da camada anterior, forma uma arquitetura conhecida por rede neural totalmente conectada. Em aplicações específicas, especialmente no caso de visão computacional e em séries temporais, relações espaciais entre atributos de entrada são de alta relevância para identificação de padrões. Para esse tipo de problema, redes totalmente conectadas são pouco eficientes, justamente por ignorarem a relação espacial dos componentes de entrada, uma vez que cada unidade trata as entradas de maneira independente. Redes Neurais Convolucionais (CNN) são um tipo especializado de rede neural para processamento de dados desse tipo, isto é, com topologia em rede [35].

A principal característica desse tipo de rede é a substituição da operação de combinação linear que ocorre em uma unidade padrão por uma operação de convolução entre a entrada e um certo filtro de pesos (*kernel*). É importante destacar que a operação referida como

convolucional para redes neurais é comumente apenas uma operação de correlação cruzada [35]. A diferenciação entre as duas operações se dá simplesmente pelo não espelhamento (no caso da correlação cruzada) do filtro.

No âmbito de NLP, é razoável imaginar que as redes CNN possam ser um interessante caminho para treinamento de modelos, sobretudo devido ao compartilhamento de parâmetros ao longo de um documento. Uma CNN pode capturar informação espacial em sequências de sentenças, palavras e/ou caracteres — que compõem uma série temporal—, o que permite à rede aprender sobre as relações semânticas entre termos próximos. Diferentemente de imagens, textos apresentam-se como uma sequência em uma única dimensão, o que torna necessário que o operador convolucional atue apenas na direção do texto, resultando em uma rede CNN unidimensional. Computa-se a correlação cruzada unidimensional de um elemento i de uma entrada 1-D I por um filtro $K \in \mathbb{R}^M$ [35]:

$$S(i) = (I * K)(i) = \sum_m I(i+m)K(m). \tag{7}$$

Quando passa pela camada de *embedding*, o texto de um documento de entrada é convertido em uma sequência de vetores $\mathbf{x}_n \in \mathbb{R}^{d_e}$ representativos de palavras, em que d_e é a dimensão de *embedding*. Os vetores são concatenados horizontalmente de modo a compor a matriz de entrada da camada convolucional $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N]$.

A camada convolucional unidimensional realiza então a combinação de vetores próximos a partir de um conjunto de filtros convolucionais que compõem $\mathbf{W} \in \mathbb{R}^{k \times d_c \times d_e}$, em que k é o tamanho do filtro convolucional e d_c é o número de filtros. A cada passo n computa-se:

$$\mathbf{h}_n = g\left(\sum_{j=1}^{k} \mathbf{W}_j \mathbf{x}_{n+j} + \mathbf{b}\right), \tag{8}$$

em que g é a função não-linear de ativação aplicada elemento a elemento, $\mathbf{b} \in \mathbb{R}^{d_c}$ é um vetor de *bias*, e $\mathbf{W}_1, \mathbf{W}_2, ..., \mathbf{W}_k \in \mathbb{R}^{d_e \times d_c}$. Adicionalmente, preenchem-se o início e fim da matriz de entrada com zeros, de modo a obter uma matriz resultante $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_N]$, tal que $\mathbf{H} \in \mathbb{R}^{d_c \times N}$.

2.4.3 Redes Neurais Recorrentes

Como já colocado, redes neurais totalmente conectadas ignoram a sequência de atributos de entrada e ademais, exigem um tamanho fixo de entrada. Redes Neurais Recorrentes (RNNs) compõem uma família de redes especializadas em processamento de dados sequenciais. Similarmente a CNNs, RNNs têm a capacidade de capturar atributos locais e também se utilizam de compartilhamento de parâmetros para processar entradas de diferentes formas e relacionar diferentes partes das sequências de entrada, o que é altamente relevante no caso em que certa parte de informação pode ocorrer em múltiplas posições em uma sequência [35].

Supondo-se uma sequência de entrada $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N]$, em que $\mathbf{x}_n \in \mathbb{R}^{d_e}$ é a representação vetorial correspondente à n-*ésima* palavra de uma sequência, uma rede recorrente é aquela em que seu estado atual representado por um vetor $\mathbf{h}_n \in \mathbb{R}^{d_c}$ em uma posição n, em que d_c é o número de unidades da rede, é obtido a partir de uma certa função g com parâmetros $\mathbf{W}$, que toma como entradas $\mathbf{x}_n$ e o estado anterior $\mathbf{h}_{n-1}$, de modo que

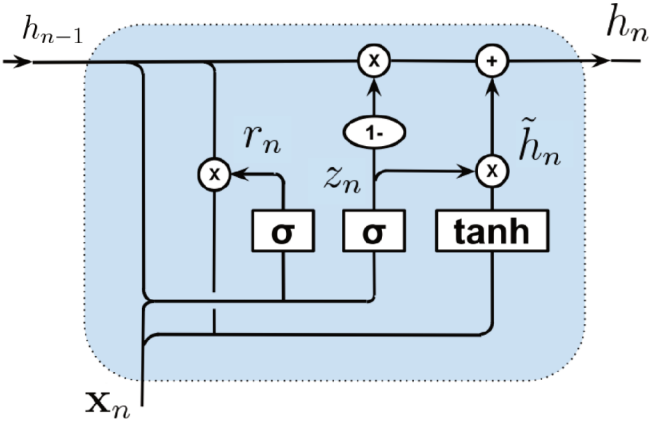
$$\mathbf{h}_n = g(\mathbf{h}_{n-1}, \mathbf{x}_n), \tag{9}$$

em que a função g e a matriz de pesos $\mathbf{W}$ são compartilhados entre os estados.

Na arquitetura mais clássica de uma RNN, a função g é simplesmente uma operação não-linear aplicada a uma combinação linear entre $\mathbf{h}_{n-1}$ e $\mathbf{x}_n$. Entretanto, essa forma padrão apresenta problemas como memória de curto prazo e explosão e/ou desvanecimento de gradiente ao longo de atualizações, o que torna seu treinamento pouco eficiente em determinadas aplicações [35]. RNNs mais eficientes utilizam modelos sequenciais conhecidos por *gated RNNs*, como LSTM (*Long Short-Term Memory*) e GRU (*Gated Recurrent Unit*). Tais redes realizam um conjunto de operações com diferentes objetivos, como permitir ao modelo propagar estados a uma distância/tempo maior ao longo da rede, aprender a esquecer estados quando a informação já foi utilizada e permitir um fluxo ininterrupto de gradiente para aumentar a eficiência de treinamento [35].

A arquitetura GRU foi a RNN escolhida para este trabalho, por

Figura 2: Diagrama de uma unidade GRU. Note que h_n é um elemento de $\mathbf{h}_n$.



Fonte: Elaborado pelo autor (2020).

possuir em geral maior eficácia que LSTM e poder ser treinada com mais eficiência [36]. A Figura 2 apresenta o diagrama de uma unidade GRU. A unidade GRU possui duas portas, uma de atualização (*update gate*) e outra de esquecimento (*reset gate*), as quais realizam operações distintas e são então combinadas para determinar um novo estado. Definem-se os vetores de saída no n-*ésimo* estado $\mathbf{z}_n$ e $\mathbf{r}_n$, respectivamente para as portas de atualização e esquecimento:

$$\mathbf{z}_n = \sigma(\mathbf{W}_{xz}\mathbf{x}_n + \mathbf{W}_{hz}\mathbf{h}_{n-1} + \mathbf{b}_z), \tag{10}$$

$$\mathbf{r}_n = \sigma(\mathbf{W}_{xr}\mathbf{x}_n + \mathbf{W}_{hr}\mathbf{h}_{n-1} + \mathbf{b}_r), \tag{11}$$

em que $\mathbf{W}_{xz}, \mathbf{W}_{xr} \in \mathbb{R}^{d_c \times d_e}$ e $\mathbf{W}_{hz}, \mathbf{W}_{hr} \in \mathbb{R}^{d_c \times d_c}$ são matrizes de pesos, $\mathbf{b}_z, \mathbf{b}_r \in \mathbb{R}^{d_c}$ são vetores de *bias* e σ é a operação sigmoide elemento a elemento.

A partir das saídas das portas, define-se a equação de atualização do vetor de estado $\mathbf{h}_n$:

$$\tilde{\mathbf{h}}_n = \tanh(\mathbf{W}_{xh}\mathbf{x}_n + \mathbf{W}_{hh}(\mathbf{r}_n \odot \mathbf{h}_{n-1}) + \mathbf{b}_h), \tag{12}$$

$$\mathbf{h}_n = (1 - \mathbf{z}_n) \odot \mathbf{h}_{n-1} + \mathbf{z}_n \odot \tilde{\mathbf{h}}_n, \tag{13}$$

em que $\mathbf{W}_{xh} \in \mathbb{R}^{d_c \times d_e}$ e $\mathbf{W}_{hh} \in \mathbb{R}^{d_c \times d_c}$ são matrizes de pesos, $\mathbf{b}_h \in \mathbb{R}^{d_c}$ é um vetor de *bias* e $\odot$ é a operação de produto elemento a elemento.

Concatenando as saídas de cada estado de atualização, compõe-se a matriz $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_N]$, tal que $\mathbf{H} \in \mathbb{R}^{d_c \times N}$.

2.4.4 Métodos de Pooling

A saída de uma camada ou sequências de camadas de uma rede neural convolucional ou recorrente gera uma representação matricial $\mathbf{H} \in \mathbb{R}^{d_c \times N}$ do documento. Para criar uma saída compatível com a representação vetorial dos rótulos, tipicamente reduz-se a matriz $\mathbf{H}$ a um vetor, aplicando-se uma operação de *pooling* global a cada linha da matriz. Assim, define-se um vetor $\mathbf{v}$ comum a todos rótulos, que pode ser computada pelo maior valor a cada linha da matriz $\mathbf{H}$, tal que

$$v_j = \max_n h_{n,j}, \tag{14}$$

em uma operação denominada *Global Max Pooling*; ou pela média dos valores ao longo de cada linha, de modo que

$$v_j = \frac{\sum_{n=1}^{N} h_{n,j}}{N}, \tag{15}$$

pelo método de *Global Average Pooling*. Independentemente do método empregado, o vetor base $\mathbf{v}$ passa então por uma camada linear de ativação sigmoide σ com tantas unidades quanto o número de rótulos, no caso de um problema multi-rótulo. Em cada unidade, computa-se a predição $\hat{y}_l \in [0, 1]$ sobre um rótulo l por:

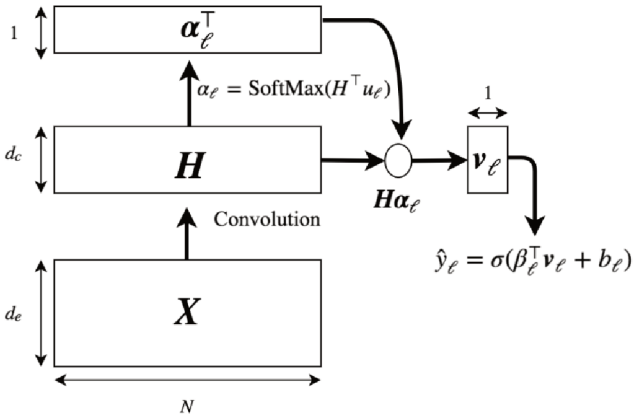
$$\hat{y}_l = \sigma(\boldsymbol{\beta}_l^T \mathbf{v} + b_l), \tag{16}$$

em que $\boldsymbol{\beta}_l \in \mathbb{R}^{d_c}$ é um vetor de pesos e b_l é um *bias* da unidade.

2.4.5 Atenção por rótulo

Em substituição à abordagem mais comum de aplicação de um método de *pooling*, pode-se fazer uso de um mecanismo de atenção por rótulo. A aplicação desse mecanismo consiste em ponderar a represen-

Figura 3: Diagrama de um mecanismo de atenção por rótulo.



Fonte: Mullenbach et al. (2018) [5].

tação base gerada por uma rede neural diferentemente para cada classe. Essa estratégia é apresentada em [37] e utilizada pelo atual estado da arte [5]. Os pesos de ponderação são treináveis e permitem que a importância de cada parte da representação base criada na saída da rede seja maior ou menor a depender de cada rótulo.

A Figura 3 apresenta o diagrama do mecanismo de atenção. Para cada rótulo l define-se um vetor de parâmetros $\mathbf{u}_l \in \mathbb{R}^{d_c}$ correspondente àquele rótulo. O vetor de atenção gerado para cada rótulo é computado a partir do vetor de parâmetros e da representação base $\mathbf{H}$, tal que

$$\alpha_l = SoftMax(\mathbf{H}^T \mathbf{u}_l), \tag{17}$$

$$SoftMax(\mathbf{x}) = \frac{\exp(\mathbf{x})}{\sum_i \exp(x_i)}, \tag{18}$$

em que $\exp(\mathbf{x})$ é a operação exponencial elemento a elemento do vetor $\mathbf{x}$. A partir de cada vetor de atenção α_l define-se a representação vetorial correspondente ao rótulo l como

$$\mathbf{v}_l = \sum_{n=1}^{N} \alpha_{l,n} \mathbf{h}_n. \tag{19}$$

A representação vetorial de cada rótulo passa finalmente por uma unidade com função de ativação sigmoide σ, tal que

$$\hat{y}_l = \sigma(\boldsymbol{\beta}_l^T \mathbf{v}_l + b_l), \tag{20}$$

em que $\boldsymbol{\beta}_l \in \mathbb{R}_c^d$ é um vetor de pesos, b_l é um *bias* e $\hat{y}_l \in [0, 1]$ é a predição do modelo em relação à amostra l.

3 ANÁLISE DE DADOS

Esta seção apresenta a descrição, comparação, análise e pré-processamento dos dados utilizados neste trabalho.

3.1 TIPOS DE DOCUMENTOS MÉDICOS

Os documentos estudados neste trabalho são redigidos por médicos e profissionais da saúde, contendo informações relativas a um paciente, seu histórico clínico e sua estadia em um hospital. Os documentos médicos estudados são de três tipos: Anamnese/Exame Físico, Evolução Clínica e Sumário de Alta. Descrevem-se esses documentos como:

- **Anamese/Exame Físico**: documento gerado a partir de um pronto atendimento. Contém uma análise inicial do paciente, incluindo condições sintomáticas e informação sobre seu histórico clínico.

- **Evolução Clínica**: documento gerado a partir do acompanhamento da condição clínica de um paciente em sua estadia. Apresenta sintomas, respostas a medicamentos e resultados de exames. Uma única estadia pode gerar vários documentos desse tipo.

- **Sumário de Alta**: documento gerado após a alta de um paciente, isto é, com o término de um atendimento. É o tipo de documento mais completo, compilando histórico e fatos ocorridos durante todo o atendimento.

3.2 SISTEMA DE CODIFICAÇÃO CID

A Classificação Internacional de Doenças e Problemas Relacionados à Saúde (CID) é publicada pela Organização Mundial da Saúde, com o objetivo de padronizar mundialmente a codificação de doenças. A classificação compreende uma extensa variedade de sintomas, sinais, queixas, históricos do paciente, circunstâncias sociais, diagnósticos e causas externas de doenças e ferimentos. O sistema de classificação é periodicamente revisado e atualizado, além de possuir flexibilidade

para modificações e adendos segundo necessidades específicas. Os códigos estão organizadas em capítulos, grupos, categorias e subcategorias, respectivamente do menor para o maior grau de especificidade.

No sistema CID-10 os códigos são compostos por até 4 dígitos. Tomemos o código CID I700 como exemplo. A letra "I" indica que o código faz parte do Capítulo IX, "Doenças do aparelho circulatório". Os três primeiros dígitos compõem a categoria da CID. "I70" indica a categoria "Aterosclerose". Categorias entre I70 e I79 enquadram-se no grupo "Doenças das artérias, das arteríolas e dos capilares". Finalmente, o último dígito identifica o código com maior especificidade, indicando a subcategoria I700, "Aterosclerose da aorta". Encontra-se também a notação com um ponto separando o dígito referente à subcategoria (e.g., I70.0).

Um sistema paralelo de codificação CID contém adicionalmente códigos associados a procedimentos médicos. Esses códigos estão presentes em um dos conjuntos de dados analisado, mas não no conjunto de interesse. Grande parte da literatura utiliza apenas os códigos do sistema padrão, conhecido por códigos CID de diagnóstico. Por esses fatores, apenas códigos CID de diagnóstico foram utilizadas neste trabalho.

3.3 DATASET MIMIC-III

O banco de dados MIMIC-III — a terceira revisão do MIMIC, na versão 1.4 — contém dados publicamente acessíveis em língua inglesa relativos a pacientes do Centro Médico *Beth Israel Deaconess*, nos Estados Unidos [12]. Cada entrada de um paciente no hospital está associada a diversos documentos médicos, além de uma lista ordenada de códigos CID em seu nível mais específico (i.e., subcategoria), em classificação CID-9-CM (em que CM refere-se a Modificação Clínica).

Tal como a maior parte dos trabalhos relacionados que fazem uso do MIMIC-III, apenas sumários de alta em texto livre foram selecionados, sendo um único documento selecionado por admissão. Um total de 52722 admissões correspondentes a 41127 pacientes foram selecionadas. Nesses documentos, encontraram-se 6918 códigos CID únicos. A Tabela 1 apresenta a descrição dos códigos CID mais ocorrentes neste conjunto de dados.

O pré-processamento dos dados foi realizado tal como visto em

Tabela 1: Definições dos 10 códigos CID mais ocorrentes no *dataset* MIMIC-III

CID	Definição	[% de amostras]
4019	*Hypertension*	38,02
4280	*Congestive heart failure*	24,36
42731	*Atrial fibrillation*	23,88
41401	*Coronary atherosclerosis*	23,10
5849	*Acute kidney failure*	16,89
25000	*Diabetes Type II*	16,66
2724	*Hyperlipidemia*	16,13
51881	*Acute respiratory failure*	13,75
5990	*Urinary tract infection*	12,22
53081	*Esophageal reflux*	11,67

Fonte: Elaborado pelo autor (2020).

trabalhos anteriores [24], removendo padrões de data/hora, caracteres especiais e tornando caracteres em minúsculos. Os dados foram separados em três conjuntos exatamente como em [5], com 47719 (90,5%) amostras no subconjunto de treinamento, 1631 (3,1%) no subconjunto de validação, e 3372 (6,4%) no subconjunto de teste. Nenhum paciente é listado em mais de um subconjunto.

3.4 DATASET HSL

O banco de dados HSL contém documentos associados a pacientes do Hospital Sírio-Libanês. Coletados entre 2016 e 2018, os textos são redigidos em português brasileiro. Diversos tipos de documentos médicos em texto livre estão disponíveis. Cada documento está associado a uma ID de admissão no hospital, a partir da qual diferentes documentos podem ser agrupados.

Para o uso neste trabalho, todas as admissões que não possuíam um sumário de alta relacionado foram descartadas, resultando em 77005 admissões de 51298 pacientes únicos. Cada admissão está vinculada a uma lista de códigos CID, os quais foram atribuídos por codificadores

Tabela 2: Definições dos 10 códigos CID mais ocorrentes no *dataset* HSL

CID	Definição	[% de amostras]
I10	Hipertensão essencial (primária)	34,37
E788	Outros distúrbios do metabolismo de lipoproteínas	24,67
Z871	História pessoal de doenças do aparelho digestivo	21,17
Z720	Uso do tabaco	30,07
E039	Hipotireoidismo não especificado	14,70
Z874	História pessoal de doenças do aparelho geniturinário	13,74
E669	Obesidade não especificada	13,53
Z873	História pessoal de doenças do sistema osteomuscular e tecido conjuntivo	12,14
I700	Aterosclerose da aorta	10,95
Z721	Uso de álcool	10,71

Fonte: Elaborado pelo autor (2020).

profissionais utilizando o códigos CID-10 de diagnóstico, em seu nível mais específico (i.e., subcategoria). Um total de 5360 diferentes códigos está presente nos dados selecionados. A Tabela 2 apresenta as definições dos dez códigos mais frequentes no conjunto de dados.

Inicialmente, apenas sumários de alta (S) foram selecionados, com um único documento vinculado a cada admissão. Esse conjunto será referenciado por HSL-S. Entretanto, após análises mais profundas e comparações objetivas com os sumários de alta presentes no MIMIC-III, optou-se por por incluir documentos adicionais com alta disponibilidade no banco de dados, em especial: Evoluções Clínicas (E) e Anamneses/Exames Físicos (A). Diferentemente dos sumários de alta, cada admissão pode estar associada a números variados de documentos de tipos E e A, entre nenhum e diversos, especialmente no caso de documentos tipo E, uma vez que são gerados a partir do acompanhamento médico do paciente durante sua estadia.

Tabela 3: Estatísticas referentes a tipos de dcoumentos nos conjuntos de dados MIMIC-III e HSL.

Dataset	Pacientes únicos	Admissões	Total de documentos	Média de palavras por amostra[a]
MIMIC-III[b]	41127	52722	52722	1327,5
HSL-S	51298	77005	77005	94,6
HSL-E	50899	76159	919713	1483,0
HSL-A	42153	59249	63423	155,4
HSL-SEA	51298	77005	1060141	1730,4

[a]Concatenação de todos os documentos correspondentes a uma mesma amostra.

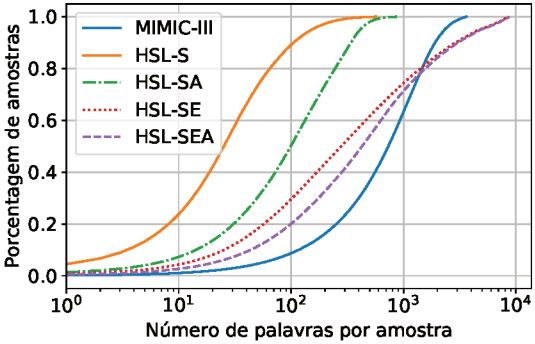
[b]Apenas sumários de alta.

Fonte: Elaborado pelo autor (2020).

Os documentos adicionais foram concatenados aos documentos de tipo S com a mesma ID de admissão, a fim de criar uma única amostra pertencente àquela ID. Essa abordagem com documentos adicionais melhor reproduz o processo de rotulação manual que acontece no Hospital Sírio-Libanês, uma vez que os codificadores observam todos os documentos de uma estadia para determinar os códigos CID correspondentes. O conjunto com documentos S, E e A concatenados será referenciado por HSL-SEA. A Tabela 3 apresenta o total de documentos por tipo, as admissões e pacientes únicos vinculados. Cabe ressaltar que, devido à concatenação, o mesmo número de amostras está presente no HSL-S e HSL-SEA e as mesmas IDs de admissão foram utilizadas.

O pré-processamento dos dados é realizado da mesma maneira que no conjunto MIMIC-III. Dado que mais de uma admissão pode estar relacionada a um único paciente, os dados foram separados certificando-se que um paciente não está presente em mais de um único subconjunto. O subconjunto de treinamento contém um total de 69309 (90,0%) amostras, o de validação 2313 (3,0%), e o de teste 5383 (7,0%).

Figura 4: Distribuição cumulativa de contagem de palavras por amostra em todos os conjuntos de dados.

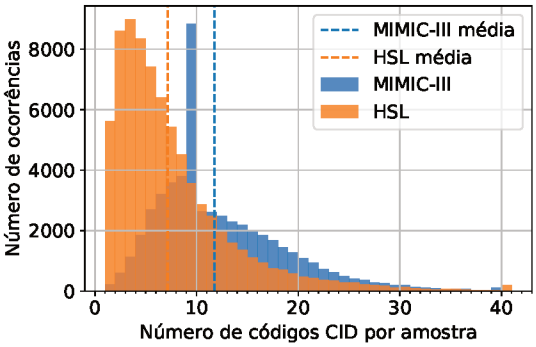


Fonte: Elaborado pelo autor (2020).

3.5 COMPARAÇÃO ENTRE OS DATASETS

Além da língua utilizada, os conjuntos de dados MIMIC-III e HSL apresentam outras diferenças relevantes. A Figura 4 mostra a distribuição cumulativa da contagem de palavras por amostra em todos os conjuntos de dados estudados, após o pré-processamento. A Tabela 3 apresenta diferenças no número de documentos selecionados para cada conjunto. Como também apresentado pela Tabela 3, os sumários de alta presentes no MIMIC-III possuem uma média de palavras por amostra muito superior que o que se observa no HSL-S. Já, após a concatenação de documentos tipo S, E e A essa média torna-se mais comparável, como se vê no conjunto HSL-SEA. De fato, o estudo dessas médias foi importante na decisão por utilizar documentos adicionais no HSL, uma vez que, pelos dados apresentados, é possível verificar que os sumários de alta do MIMIC-III contêm, objetivamente, muito mais informação que o mesmo tipo de documento no HSL, ao passo que apresenta uma média de palavras e distribuição cumulativa mais próxima a HSL-SEA. Além disso, a partir de uma amostragem aleatória de documentos, nota-se, subjetivamente, que os sumários de alta do MIMIC-III são melhor escritos, construídos de maneira mais dissertativa e detalhada.

Figura 5: Histogramas de contagem de códigos CID por amostra nos *datasets* MIMIC-III e HSL.



Fonte: Elaborado pelo autor (2020).

Os sistemas de codificação empregados nos conjuntos de dados são também adversos. Enquanto o MIMIC-III adota uma modificação clínica do sistema CID-9, HSL utiliza o mais recente CID-10. A Figura 5 apresenta histogramas da contagem de códigos CID por amostra em ambos os conjuntos de dados. Ainda que o máximo, mínimo e desvio padrão sejam similares, a média de códigos CID atribuídos por amostra é inferior no HSL. É notável que as classes (i.e., códigos CID) são extremamente desbalanceadas em ambos os conjuntos de dados, como ilustrado por alguns exemplos na Tabela 4. Cabe mencionar também

Tabela 4: Porcentagem de amostras rotuladas com o primeiro, décimo, centésimo e milésimo códigos CID mais frequentes.

Ranking	MIMIC-III	HSL
1	38,02%	34,37%
10	11,67%	10,71%
100	2,23%	1,26%
1000	0,15%	0,06%

Fonte: Elaborado pelo autor (2020).

que, respectivamente, 4,47% e 4,48% dos códigos CID presentes nos subconjuntos de teste do MIMIC-III e HSL não estão presentes no conjunto de treinamento, o que torna sua predição impossível por todos os modelos estudados.

4 MÉTODOS

Esta seção dedica-se à apresentação de métricas de avaliação e modelos desenvolvidos neste trabalho.

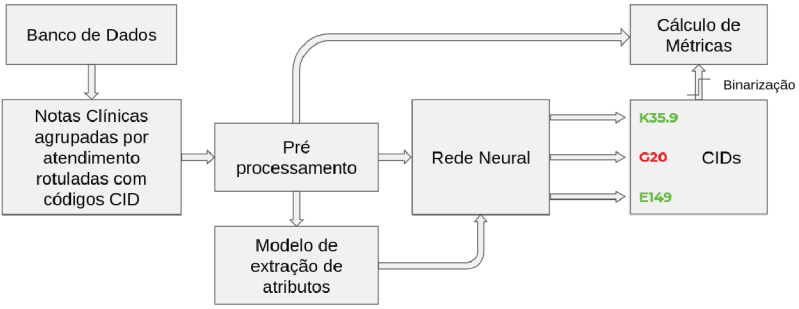
4.1 HARDWARE DE IMPLEMENTAÇÃO

Os dados do MIMIC-III e HSL foram disponibilizados via Amazon S3. Todo o desenvolvimento deste trabalho foi realizado em uma máquina virtual AWS EC2, com 8 núcleos virtuais de computação, 30 GB de memória RAM e uma placa gráfica NVIDIA T4 com 16 GB de memória dedicada.

4.2 PIPELINE PARA O PROBLEMA DE CLASSIFICAÇÃO MULTI-RÓTULO A PARTIR DE REDES NEURAIS

O problema de classificação multi-rótulo a partir de texto livre segue um *pipeline* comum para todos os modelos baseados em rede neural estudados neste trabalho. A Figura 6 apresenta um diagrama que ilustra os passos para o treinamento de um modelo. Dados de interesse são selecionados a partir de um banco de dados e separados em conjuntos definidos: um para treinamento da rede; outro para validação de arquitetura e otimização de hiperparâmetros; e um para teste final do modelo. Cada amostra carrega um par entrada/saída, isto é, um texto de entrada associado a uma lista de códigos CID que corretamente rotulam aquela amostra (*ground-truth*). As amostras selecionadas são então pré-processadas. Cada texto é manipulado conforme apresentado nas Seções 3.3 e 3.4, cada palavra é convertida para um *token* (no caso, um número inteiro que representa a sua posição em um vocabulário comum a todo o *corpus*), e finalmente cada texto é truncado ou preenchido a fim de atender a um tamanho fixo de entrada. Já as listas de códigos associados são transformadas a partir de uma codificação *multi-hot* — ou seja, cada amostra passa a ter um vetor booleano associado de dimensão igual ao total de classes disponíveis no banco de dados, com valores verdadeiros para as posições que correspondem aos rótulos atribuídos àquela amostra.

Figura 6: Pipeline do problema de classificação multi-rótulo a partir de texto não-estruturado.



Fonte: Elaborado pelo autor (2020).

A partir do texto pré-processado treinam-se representações vetoriais a nível de palavra, de modo a projetar cada palavra de um texto de amostra em um espaço multidimensional de dimensão reduzida inteligível para redes neurais. Neste trabalho, o algoritmo Word2Vec (veja Seção 2.2.2) foi utilizado para tal treinamento.

Após essas etapas, as condições para treinamento e avaliação das redes neurais estão estabelecidas. As representações vetoriais de palavras são carregadas em uma camada de *embedding*, a fim de possibilitar o mapeamento dos *tokens* de entrada a seus respectivos vetores. A rede neural é então treinada, atribuindo na saída, para cada amostra, um vetor $\hat{\mathbf{y}}$ de dimensão igual ao número total de classes com valores reais $\hat{y}_i \in [0, 1]$ correspondentes à confiança ou *score* do modelo a cada classe. O treinamento do modelo se dá pela atualização de pesos, a partir de uma função custo, obtida a partir da função perda de entropia cruzada

$$L(\mathbf{y}, \hat{\mathbf{y}}) = -\mathbf{y} \log \hat{\mathbf{y}} - (1 - \mathbf{y}) \log(1 - \hat{\mathbf{y}}), \tag{21}$$

em que $\mathbf{y}$ e $\hat{\mathbf{y}}$ são respectivamente, o vetor de saída verdadeira de uma amostra conhecido durante o treinamento e o vetor de predição do modelo.

A saída da rede deve então ser binarizada (Seção 4.3), para finalmente permitir o cálculo das métricas de avaliação (veja a Seção

4.4), as quais serão indicativas do desempenho do modelo em um certo subconjunto avaliado.

A partir desse *pipeline*, é possível treinar e avaliar os modelos baseados em redes neurais que serão apresentados a seguir. É importante mencionar que, de modo geral, esse fluxo de processamento é padrão para o problema estudado, embora diversas importantes modificações quanto a pré-processamento, métodos de extração de atributos e arquiteturas de rede são possíveis.

4.3 QUANTIZAÇÃO DA SAÍDA

Os modelos desenvolvidos neste trabalho possuem unidades com ativação sigmoide na saída, o que implica que, para cada amostra, um vetor $\tilde{\mathbf{y}}_n$ de dimensão C (número de classes) com entradas em valores reais entre 0 e 1 correspondentes à confiança do modelo de que aquela classe está presente naquela amostra. Como o problema estudado é multi-rótulo, faz-se necessária uma interpretação sobre esses vetores para que sejam convertidos em um conjunto de códigos CID preditos para cada amostra. Neste trabalho, optou-se por utilizar duas abordagens distintas: otimizar um limiar de binarização das saídas dos modelos; e computar saídas com números fixos de rótulos por amostra.

4.3.1 Baseada em limiar

Nesta abordagem, define-se que o modelo prediz uma classe em uma amostra se o valor correspondente à confiança é maior que um certo limiar t. Assim, para cada predição $\tilde{y}_n^c \in [0, 1]$ da classe c sobre amostra n, define-se o valor binário de predição $\hat{y}_n^c$:

$$\hat{y}_n^c = \begin{cases} 1 & \text{, se } \tilde{y}_n^c > t \\ 0 & \text{, caso contrário} \end{cases}, \tag{22}$$

o que resulta em um número variado de CIDs preditas para cada amostra.

Essa abordagem é corrente na literatura [14] — de fato, essa é a abordagem padrão se o limiar é fixado em 0,5 — e mostra-se interessante do ponto de vista de análise geral de desempenho do modelo.

4.3.2 Baseada em saída de tamanho fixo

Nesta abordagem, supõe-se um número fixo de códigos CID preditos para cada amostra. Para cada amostra n, os k maiores valores de confiança do vetor de predições $\tilde{\mathbf{y}}_n \in \mathbb{R}^C$ de um modelo são definidos como predições positivas, enquanto as demais classes são definidas como não preditas, compondo um vetor de entradas binárias $\hat{\mathbf{y}}_n \in \mathbb{R}^C$ para cada amostra. Tal abordagem é interessante do ponto de vista da aplicação estudada neste trabalho, como será discutido na Seção 4.4.

4.4 MÉTRICAS DE AVALIAÇÃO

A escolha de uma métrica de avaliação é fundamental para o entendimento acerca do desempenho dos modelos estudados. Entre as métricas populares em problemas multi-rótulo estão *Precision*, *Recall* (taxa de verdadeiro positivo) e F1 [38], segundo ponderações macro e micro — especialmente esta última. Essas métricas foram as escolhidas neste trabalho.

A ponderação macro compreende em computar a métrica individualmente para cada classe e, em seguida, tomar a média simples entre todas as classes — i.e., cada classe contribui igualmente para a métrica final. Em um problema com grande quantidade classes altamente desbalanceadas, verifica-se um impacto muito alto de classes raras no desempenho reportado dos modelos. Na ponderação micro, por sua vez, somam-se globalmente os verdadeiros positivos, falsos positivos e falsos negativos, para apenas então computar a métrica desejada. Assim, o impacto de cada classe é proporcional a sua ocorrência.

Para o problema deste trabalho optou-se pela ponderação micro, por duas razões primordiais: pelo fato de apresentar um resultado mais representativo sobre o alto número de classes, conferindo mais importância a classes mais ocorrentes; e por permitir comparações com a literatura, a qual utiliza majoritariamente esse tipo de ponderação.

Definem-se *Precision*, *Recall* e F1 em ponderação micro:

$$Precision_{micro} = \frac{\sum_{c=1}^{C} \sum_{n=1}^{N} y_n^c \hat{y}_n^c}{\sum_{c=1}^{C} \sum_{n=1}^{N} \hat{y}_n^c}, \tag{23}$$

$$Recall_{micro} = \frac{\sum_{c=1}^{C} \sum_{n=1}^{N} y_n^c \hat{y}_n^c}{\sum_{c=1}^{C} \sum_{n=1}^{N} y_n^c}, \tag{24}$$

$$F1_{micro} = \frac{2 \cdot Precision_{micro} \cdot Recall_{micro}}{Precision_{micro} + Recall_{micro}}, \tag{25}$$

em que C é o número total de classes (códigos CID); N é o número de amostras do *subset* avaliado; e y_n^c e $\hat{y}_n^c$ são entradas com valores binários — respectivamente, verdadeiros e preditos — indicativos de cada classe c de uma amostra n. Em termos de matriz de confusão, podem-se também definir as métricas de avaliação em ponderação micro:

$$Precision_{micro} = \frac{TP}{TP + FP}, \tag{26}$$

$$Recall_{micro} = \frac{TP}{TP + FN}, \tag{27}$$

$$F1_{micro} = \frac{2TP}{2TP + FP + FN}, \tag{28}$$

em que TP, FP e FN são, respectivamente, a soma global dos verdadeiros positivos, falsos positivos e falsos negativos em todas as classes e amostras. O Apêndice A apresenta uma interpretação sobre as métricas utilizadas, incluindo os fundos de escala e um exemplo de cálculo.

Após o treinamento de cada modelo as métricas foram computadas a partir da abordagem baseada em limiar, considerando limiares entre 0 e 1, em incrementos de 10^{-2}, utilizando o conjunto de validação. O limiar com o melhor resultado em termos de F1-micro é elegido para o cálculo de métricas no subconjunto de teste.

Adicionalmente, sob o ponto de vista de uma aplicação em que codificadores recebem de um modelo, para cada amostra, uma lista de códigos selecionáveis, a análise a partir da abordagem de lista fixa torna-se bastante representativa do desempenho dos modelos. O codificador seleciona da lista de predições os rótulos verdadeiros positivos, enquanto descarta manualmente rótulos falsos positivos. Assim, idealmente todos os rótulos verdadeiros devem estar presentes em uma lista limitada de rótulos preditos. No contexto dessa aplicação, *Recall* apresenta-se como a métrica de maior relevância. Desse modo, para este trabalho foram selecionados tamanhos fixos $k = 10$ e $k = 50$ para computar *Recall*@10 e *Recall*@50.

4.5 MODELOS

Os modelos desenvolvidos neste trabalho são apresentados a seguir. Para sua implementação utilizou-se ambiente Keras[3] baseado em Tensorflow[4]. Cada modelo foi treinado por um máximo de 10 épocas. Ao invés de utilizar uma abordagem de *Early Stopping* [35], a métrica F1 no conjunto de validação com limiar otimizado foi computadada ao final de cada época. Após a conclusão do treinamento, os pesos correspondentes à época com melhor resultado em termos dessa métrica foram restaurados.

4.5.1 Constante (top-k)

O objetivo deste modelo é prover uma métrica base, a fim de determinar se e por quanto a performance dos modelos reais supera àquela de uma implementação puramente baseada nas saídas conhecidas, isto é, que não utiliza os textos de entrada.

O modelo constante prevê uma lista fixa de k códigos CID para todas as amostras. Os códigos CID selecionados são os k mais frequentes no subconjunto de treinamento. O parâmetro k — o tamanho da lista — foi otimizado para obter a melhor métrica F1 no conjunto de validação, resultando em $k = 15$ para o conjunto MIMIC-III e $k = 8$ para o conjunto HSL. Cabe ressaltar que, uma vez que este modelo não utiliza os textos de entrada, os resultados para HSL-S e HSL-SEA são os mesmos.

4.5.2 Regressão Logística

O modelo de Regressão Logística (LR) trata o problema multirótulo como um conjunto de problemas de classificação binária, um para cada classe. As entradas do modelo LR são atributos TF-IDF computados sobre os subconjuntos de treinamento de cada conjunto de dados.

O algoritmo para cálculo de TF-IDF foi implementado utilizando Scikit-learn[5]. *Stopwords*[6] nas respectivas línguas foram retiradas dos

[3] http://keras.io/
[4] https://tensorflow.org/
[5] https://scikit-learn.org/
[6] *Stopwords* são palavras comuns em qualquer linguagem que ocorrem em alta frequência e em geral carregam isoladamente pouca informação [39]. Exemplos

textos, utilizando os pacotes padrões em português e inglês providos pelo Natural Language Toolkit[7]. O tamanho do vocabulário, isto é, o número de atributos TF-IDF de entrada foi limitado às 20000 palavras mais frequentes no *corpus* de treinamento.

Os hiperparâmetros do modelo LR foram otimizados para MIMIC-III por busca em grade considerando diferentes otimizadores, taxas de aprendizado entre 10^{-4} e 10^{-1}, em múltiplos de 10 e fatores de regularização L2 entre 0 (i.e., sem regularização) e 10, também em múltiplos de 10. A implementação final utiliza otimizador Adam [40] com taxa de aprendizado 10^{-1} e não realiza regularização. Os demais parâmetros usam valores padrão do módulo.

Cada época levou 50 segundos para treinamento no MIMIC-III e 60 segundos para HSL-S e HSL-SEA.

4.5.3 Rede Neural Convolucional

A CNN implementada consiste em uma camada de *embedding* com mapeamento para vetores treinados com Word2Vec, seguida por uma única camada convolucional unidimensional e *Batch Normalization* [41]. Na saída, uma operação de *Global Average Pooling* precede uma camada totalmente conectada com tantas unidades quanto o número de classes para cada conjunto de dados. A implementação é baseada em [5], com algumas importantes modificações: o *Dropout* [35] da camada de *embedding* foi removido, adicionou-se *Batch Normalization* e o tamanho do *kernel* dos filtros da camada convolucional foi aumentado de 4 para 10. Além disso, a implementação original utilizava a operação *Global Max Pooling* na saída, a qual foi substituída por *Global Average Pooling*. Testes em implementações similares à utilizada mostraram que cada uma dessas modificações resulta individualmente em um ganho incremental de performance.

As camadas e respectivos parâmetros estão dispostos na Tabela 5. Utilizou-se otimizador Adam com taxa de aprendizado 10^{-3} para MIMIC-III e 3×10^{-1} para HSL-SEA.

Devido ao fato de que o modelo CNN utiliza *Batch Normalization*, é mandatório às entradas que possuam tamanhos fixos [41]. Entretanto, as amostras em texto livre apresentam-se com uma enorme

incluem preposições, artigos e conjunções.

[7] https://www.nltk.org/

Tabela 5: Arquiteturas e parâmetros dos modelos de redes neurais.

CNN	GRU	CNN-Att
Entrada	Entrada	Entrada
Embedding (tamanho 300)	Embedding (tamanho 300)	Embedding (tamanho 300)
Conv1D (500 filtros, kernel 10, tanh)	GRU (500 unidades, tanh)	Conv1D (500 filtros, kernel 10, tanh)
Batch Normalization	Batch Normalization	Batch Normalization
Global Average Pooling 1D	Global Average Pooling 1D	Attention
Saída (sigmoide)	Sada (sigmoide)	Saída (sigmoide)

Fonte: Elaborado pelo autor (2020).

variedade de tamanhos (leia-se número de palavras). Para garantir uma entrada de tamanho fixo, textos de entrada com poucas palavras foram preenchidos por *tokens* de preenchimento ao final do documento. Por outro lado, textos muito extensos foram truncados a partir do final. Observando as distribuições do número de palavras por documento nos conjuntos de dados utilizados, os tamanhos de entrada foram fixados em: 2000, para MIMIC-III; 300, para HSL-S; e 4000 para HSL-SEA.

Os vetores carregados na camada de *embedding* foram treinados por um algoritmo Word2Vec provido pelo módulo Gensim[8]. Os *embeddings* foram treinados a partir dos próprios conjuntos de treinamento. A opção por não utilizar conjuntos pré-treinados se deu devido à especificidade da linguagem clínica apresentada nos conjuntos, contendo termos médicos, abreviações e acrônimos [10]. Palavras com menos de 10 ocorrências no *corpus* foram descartadas. Foram experimentados vetores com dimensões entre 100 e 600, algoritmos de treinamento CBoW e *Skip-gram*, e remoção ou não de *stopwords*. Esses parâmetros foram otimizados para HSL e MIMIC-III, resultando em vetores de dimensão 300 treinados por Skip-gram, sem remoção de *stopwords*.

As entradas dos modelos de redes neurais são textos preprocessados com palavras substituídas por números inteiros que permitem sua

[8] https://radimrehurek.com/gensim/

vinculação ao seu respectivo vetor carregado na camada de *embedding*. Palavras desconhecidas e *tokens* de preenchimento são mapeadas para vetores nulos.

Cada época de treinamento do modelo CNN levou 310 segundos para o MIMIC-III e 820 para HSL-SEA.

4.5.4 Rede Neural Recorrente

O modelo de RNN consiste de uma camada de *embedding* carregada com vetores Word2Vec, seguida por uma camada de rede recorrente do tipo *Gated Recurrent Neural Network* (GRU). Em seguida, realiza-se *Batch Normalization* e *Global Average Pooling*. Na saída utiliza-se uma camada totalmente conectada de ativação sigmoide com tantas unidades quanto o número de classes de cada *dataset*. O uso de unidades GRU se justifica por seus resultados superiores aos de redes recorrentes tradicionais em tarefas de NLP, ao mesmo tempo em que apresentam uma arquitetura mais simples que unidades LSTMs [36]. Tal como no modelo CNN, as amostras foram processadas para atender a um tamanho fixo de entrada, neste caso para também permitir treinamento mais rápido por GPU.

Para a rede GRU, três arquiteturas base foram testadas com MIMIC-III: a primeira é tal como apresentada pela Tabela 5; a segunda possui uma camada adicional com unidades GRU; a última possui uma única camada, mas com unidades GRU bidirecionais, ao invés das GRU comuns. A primeira arquitetura obteve os melhores resultados em ambos os conjuntos de dados. Deste modo, cada parâmetro foi individualmente otimizado a partir dessa arquitetura base.

Os parâmetros foram otimizados para MIMIC-III e HSL-SEA, incluindo: otimizadores e taxas de aprendizado entre 4×10^{-4} e 10^{-2} em passos de 10^{-4}; número de unidades GRU entre 100 e 500 em incrementos de 200; mascaramento de *tokens* de preenchimento, para prevenir sua influência na predição do modelo; ponderação por amostra, com pesos inversamente proporcionais ao seu número de ocorrências no *corpus*; *fine-tuning* da camada de *embedding* no treinamento da rede neural; e métodos de Pooling. Os melhores resultados foram obtidos com otimizador Adam, *fine-tuning* da camada de *embedding* e *Average Pooling*. A arquitetura final é apresentada na Tabela 5 e será referenciada apenas por GRU nas próximas seções.

Os tempos de treinamento do modelo GRU foram de 268 segundos por época para o MIMIC-III e 785 segundos para HSL-SEA.

4.5.5 Rede Neural Convolucional com Atenção

O modelo de CNN com Atenção (CNN-Att) é baseado em CAML (*Convolutional Attention for Multi-Label Classification*) [5] — o atual estado da arte para o problema de classificação de códigos CID de diagnóstico no MIMIC-III —, com algumas modificações resultantes de testes realizados sobre o modelo. O modelo é também similar ao modelo de CNN deste trabalho, com a única diferença sendo a substituição do *Global Average Pooling* na saída da camada convolucional pelo mecanismo de atenção por rótulo. A operação de atenção é um produto interno escalado [37] e usa um vetor treinável separado para cada rótulo (veja Seção 2.4.5).

Em comparação ao modelo CAML original, removeu-se o Dropout aplicado sobre a camada de *embedding*, pois seu uso não apresentou ganho em performance quando comparado ao mesmo modelo sem Dropout. Além disso, adicionou-se *Batch Normalization* após a camada convolucional, uma vez que ele tipicamente permite uma mais rápida convergência durante o treinamento. O número de filtros foi aumentado de 50 para 500. Todas essas modificações mostraram ganho em performance. Adicionalmente, para obter uma convergência mais rápida do modelo agendou-se um degrau de decaimento na taxa de aprendizado, começando em 10^{-3} nas primeiras 2 épocas e assumindo um valor menor de 10^{-4} a partir da terceira época. Esse degrau apresentou os melhores resultados para ambos os conjuntos de dados. A Tabela 5 apresenta a arquitetura e parâmetros utilizados no modelo CNN-Att. Após substituir os *tokens* de entrada pelos respectivos vetores treinados por Word2Vec, os atributos passam por uma única camada convolucional. A saída da camada é então normalizada por *Batch Normalization* e a representação base gerada é enviada ao mecanismo de atenção para ponderação específica por classe, gerando o vetor de saída do modelo.

Tal como os demais modelos baseados em rede neural, foram utilizados vetores Word2Vec carregados na camada de *embedding*, e as amostras foram preenchidas e truncadas para atender a um tamanho fixo de entrada. O treinamento do modelo CNN-Att levou 1600 segundos por época para o *dataset* MIMIC-III e 3700 no HSL-SEA.

Tabela 6: Performance dos diferentes modelos no *dataset* MIMIC-III. Entradas sem citação correspondem a modelos deste trabalho.

Modelo	Limiar	F1	Precision	Recall
Constante	-	0,192	0,188	0,196
LR [5]	-	0,242	-	-
flat-SVM [14]	-	0,253	0,635	0,158
LR	0,19	0,406	0,425	0,388
CNN [5]	-	0,402	-	-
CNN [14]	-	0,399	0,440	0,366
CNN	0,30	0,423	0,467	0,387
Bi-GRU [5]	-	0,393	-	-
GRU	0,32	0,468	0,543	0,412
CAML [5]	-	0,524	-	-
CNN-Att	0,28	**0,537**	0,590	0,492

Fonte: Elaborado pelo autor (2020).

5 RESULTADOS E DISCUSSÃO

Esta seção apresenta os resultados obtidos após o treinamento de todos os modelos em ambos os *datasets*, HSL e MIMIC-III. Resultados provenientes de experimentos com documentos adicionais no HSL também são discutidos.

5.1 RESULTADOS PARA MIMIC-III

As Tabelas 6 e 7 apresentam os resultados obtidos para todos os modelos, treinados com o conjunto de dados MIMIC-III. Para fins de comparação, métricas de trabalhos anteriores foram incluídas.

Como esperado, o modelo Constante obtém métricas bastante baixas. O modelo LR com hiperparâmetros otimizados supera consideravelmente modelos similares de LR [5] e SVM [14], apresentados como *baselines*[9] nesses trabalhos. Tal resultado sugere uma falta de otimiza-

[9]No contexto de aprendizado de máquina, *baselines* são modelos de implementação em geral mais simples, cujo objetivo é prover uma base de desempenho esperado em uma tarefa para posterior comparação com modelos de maior complexidade.

Tabela 7: Performance a partir da abordagem baseada em lista fixa dos diferentes modelos no *dataset* MIMIC-III.

Modelo	Recall@10	Recall@50
LR	0,324	0,610
CNN	0,346	0,632
GRU	0,374	0,665
CNN-Att	0,415	**0,743**

Fonte: Elaborado pelo autor (2020).

ção de parâmetros nesses modelos, o que leva a *underfitting*[10] durante o treinamento; de fato, nota-se que a LR de [5] usa um fator de regularização padrão de 1, enquanto a otimização indica a retirada de qualquer regularização como uma melhor abordagem. O modelo LR atinge resultados comparáveis a implementações de redes CNN com atributos Word2Vec encontradas em [14] e [5], enquanto o modelo de CNN deste trabalho mostra uma evolução em termos das métricas utilizadas.

O modelo GRU apresenta um salto considerável sobre todos os modelos anteriores, além de um modelo similar apresentado em [5]. Finalmente, o modelo CNN-Att supera todos os demais modelos, incluindo a sua implementação original apresentada em [5].

5.2 RESULTADOS PARA HSL

Os primeiros experimentos sobre o conjunto HSL foram realizados a partir de HSL-S, isto é, considerando apenas documentos de sumário de alta, a fim de permitir uma comparação mais direta com MIMIC-III, que utiliza apenas esse tipo de documento. Considerando as diferenças discutidas na Seção 3.5, não havia expectativa por resultados similares. Ainda assim, após o treinamento do modelo LR sobre HSL-S, observaram-se métricas muito abaixo do esperado: encontrou-se F1-micro de 0.316 após otimização de limiar, o que se traduz em um desempenho 20% inferior ao mesmo modelo treinado com MIMIC-III.

[10] *Underfitting* ocorre quando um modelo é incapaz de obter baixo erro durante o treinamento [35], o que comumente acarreta em baixo desempenho nos subconjuntos de treinamento e teste.

Tabela 8: Métricas de validação do modelo LR treinado com o *dataset* HSL considerando diferentes tipos de documentos concatenados.

Documentos	Limiar	F1	Precision	Recall
S (HSL-S)	0,26	0,316	0,320	0,312
S e A	0,25	0,347	0,359	0,336
S e E	0,27	0,357	0,382	0,336
S, E e A (HSL-SEA)	0,25	**0,367**	0,390	0,346

Fonte: Elaborado pelo autor (2020).

Tabela 9: Performance dos diferentes modelos no *dataset* HSL-SEA.

Modelo	Limiar	F1	Precision	Recall
Constante	-	0,203	0,183	0,228
LR	0,25	0,368	0,400	0,340
CNN	0,26	0,374	0,386	0,363
GRU	0,29	0,441	0,508	0,390
CNN-Att	0,29	**0,485**	0,543	0,438

Fonte: Elaborado pelo autor (2020).

Tais resultados iniciais, somados ao fato de que HSL-S possui uma média de palavras por amostra significativamente inferior àquela do MIMIC-III, levaram este estudo a considerar o uso de documentos adicionais disponíveis no conjunto de dados. Para investigar o impacto em termos de métricas, o modelo de LR foi treinado em diferentes combinações de concatenação de documentos: tipos S e A; tipos S e E; e tipos S, E e A (veja a Seção 3.4 para uma explicação sobre os diferentes tipos de documento disponíveis). A Tabela 8 apresenta resultados computados sobre o conjunto de validação. Claramente, a concatenação de documentos E e A aos sumários de alta mostra melhorias nas métricas de avaliação, com uma grande diferença de performance entre HSL-S e HSL-SEA.

Considerando os resultados desse experimento, os demais modelos foram treinados com HSL-SEA. Como os modelos de CNN e RNN são sensíveis à ordem de concatenação dos documentos, foram expe-

Tabela 10: Performance a partir da abordagem baseada em lista fixa dos diferentes modelos no *dataset* HSL-SEA.

Modelo	Recall@10	Recall@50
LR	0,410	0,729
CNN	0,424	0,716
GRU	0,467	0,773
CNN-Att	0,499	**0,816**

Fonte: Elaborado pelo autor (2020).

rimentadas as ordens S-A-E e S-E-A. Esta última foi a adotada, por atingir resultados levemente superiores. Em comparação ao conjunto HSL-S, todos os modelos quando treinados com HSL-SEA obtiveram resultados consistentemente superiores. As métricas de avaliação computadas sobre o conjunto de teste para todos os modelos treinados com HSL-SEA são apresentadas nas Tabelas 9 e 10. Tal como observado no MIMIC-III, o modelo CNN obtém métricas um pouco superiores ao LR, enquanto o GRU supera significativamente ambos os modelos anteriores. Novamente, o modelo CNN-Att apresenta os melhores resultados em todas as métricas computadas.

Observa-se que cada modelo treinado com HSL-SEA alcança uma performance comparável (até 10% inferior) ao mesmo modelo treinado com MIMIC-III. Quando consideradas apenas as métricas sob a abordagem de lista fixa (vide Tabelas 7 e 10), mais relevantes para a aplicação de auxílio ao processo manual de codificação, os resultados no *dataset* HSL-SEA superam os do MIMIC-III para os dois tamanhos fixos de saída analisados. Isso torna evidente que o conjunto de dados HSL-SEA tem qualidade comparável ao que se observa nos sumários de alta do MIMIC-III, ao menos na tarefa de predição de códigos CID.

6 CONCLUSÃO

Este Trabalho de Conclusão de Curso apresentou um estudo sobre a codificação automática de códigos CID a partir de documentos médicos não estruturados, em texto livre. Para o conjunto de dados MIMIC-III, resultados de trabalhos anteriores baseados em modelos similares foram reproduzidos e superados através do aprimoramento de modelos, permitindo superar o atual estado da arte na tarefa de predição de códigos CID de diagnóstico a partir de sumários de alta em língua inglesa. Os resultados apresentados mostram como um modelo de CNN com mecanismo de atenção por rótulo ultrapassa modelos convencionais de LR, CNN e RNN, obtendo uma métrica Micro-F1 de 0,537.

Para o conjunto de dados HSL, observou-se que o uso apenas de documentos do tipo sumário de alta foi insuficiente para atingir resultados similares ao MIMIC-III. Além dos diferentes sistemas de codificação e língua, estatísticas relativas à contagem de palavras por amostra e diferentes níveis de detalhe presentes nos documentos podem justificar o decaimento observado nas métricas de avaliação na abordagem baseada em limiar. Após a concatenação de tipos adicionais de documentos encontrados no HSL, observou-se melhoria significativa nos resultados. Nesse conjunto o modelo CNN-Att atingiu novamente as melhores métricas, com Micro-F1 de 0,485.

No contexto da aplicação específica de auxílio ao processo de codificação manual, o modelo CNN-Att obteve os melhores resultados em termos de *Recall*@k em ambos os conjuntos de dados, atingindo *Recall*@50 de 0,743 no MIMIC-III e 0,816 no HSL-SEA, sendo a métrica notavelmente superior nesse último *dataset*.

Dados esses resultados, acredita-se que o melhor modelo treinado com os dados em português é adequado para auxiliar codificadores trabalhando a partir de registros clínicos em português brasileiro, permitindo ganhos em eficiência e redução de erros no processo de rotulação de códigos CID.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] ORGANIZATION, W. H. *International classification of diseases: [9th] ninth revision, basic tabulation list with alphabetic index.* World Health Organization, 1978.

[2] ORGANIZATION, W. H. *ICD-10 : international statistical classification of diseases and related health problems : tenth revision.* World Health Organization, 2004.

[3] JENSEN, P. B.; JENSEN, L. J.; BRUNAK, S. R. Mining Electronic Health Records: Towards Better Research Applications and Clinical Care. *Nature Reviews Genetics*, v. 13, p. 395–405, 2012.

[4] LARKEY, L. S.; CROFT, W. B. Automatic Assignment of ICD9 Codes To Discharge Summaries. Technical report, University of Massachusetts, Amherst, MA, 1995.

[5] MULLENBACH, J. et al. Explainable Prediction of Medical Codes from Clinical Text. In: Proceedings of the 2018 NAACL-HLT. c2018. v. 1. p. 1101–1111.

[6] LI, F.; YU, H. ICD Coding from Clinical Text Using Multi-Filter Residual Convolutional Neural Network. In: Proceedings of he 34th AAAI Conference on Artificial Intelligence. c2020.

[7] DE LIMA, L. R. S.; LAENDER, A. H. F.; RIBEIRO-NETO, B. A. A hierarchical approach to the automatic categorization of medical documents. In: Proceedings of the 7th CIKM. c1998. p. 132–139.

[8] FERRÃO, J. et al. Using Structured EHR Data and SVM to Support ICD-9-CM Coding. In: Proceedings of the 2013 IEEE ICHI. c2013. p. 511–516.

[9] OLEYNIK, M.; PATRÃO, D. F. C.; FINGER, M. Automated Classification of Semi-Structured Pathology Reports into ICD-O Using SVM in Portuguese. *Studies in Health Technology and Informatics*, v. 235, p. 256–260, 2017.

[10] DOS SANTOS, A. B. V.; GUMIEL, Y. B.; CARVALHO, D. R. Using Deep Convolutional Neural Networks with Self-Taught Word

Embeddings to Perform Clinical Coding. *Iberoamerican Journal of Applied Computing*, v. 8, 2018.

[11] DUARTE, F. et al. Deep neural models for ICD-10 coding of death certificates and autopsy reports in free-text. *Journal of Biomedical Informatics*, v. 80, p. 64–77, 2018.

[12] JOHNSON, A. E. W. et al. MIMIC-III, a freely accessible critical care database. *Scientific Data*, v. 3, p. 160035, 2016.

[13] JOHNSON, A.; POLLARD, T.; MARK, R. The MIMIC III Clinical Database, 2016.

[14] LI, M. et al. Automated ICD-9 Coding via A Deep Learning Approach. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, v. 16, p. 1193–1202, 2019.

[15] XU, K. et al. Multimodal Machine Learning for Automated ICD Coding. In: Proceedings of the 4th Machine Learning for Healthcare Conference. c2019.

[16] BAUMEL, T. et al. Multi-label classification of patient notes a case study on icd code assignment. In: AAAI Workshops. c2017.

[17] PEROTTE, A. et al. Diagnosis Code Assignment: Models and Evaluation Metrics. *JAMIA*, v. 21, p. 231–237, 2014.

[18] SUBOTIN, M.; DAVIS, A. R. A Method for Modeling Co-Occurrence Propensity of Clinical Codes with Application to ICD-10-PCS Auto-Coding. *JAMIA*, v. 23, p. 866–871, 2016.

[19] CRAMMER, K. et al. Automatic code assignment to medical text. In: . c2007. p. 129.

[20] PAKHOMOV, S. V. S.; BUNTROCK, J. D.; CHUTE, C. G. Automating the assignment of diagnosis codes to patient encounters using example-based and machine learning techniques. *JAMIA*, v. 13, p. 516–525, 2006.

[21] MEDORI, J.; FAIRON, C. Machine learning and features selection for semi-automatic ICD-9-CM encoding. In: Proceedings of the NAACL HLT 2010 Second Louhi Workshop on Text and Data

Mining of Health Documents. Los Angeles, California, USA: Association for Computational Linguistics, c2010. p. 84–89.

[22] RUCH, P. et al. From Episodes of Care to Diagnosis Codes: Automatic Text Categorization for Medico-Economic Encoding. In: Proceedings of the AMIA Annual Symposium. c2008. p. 636–640.

[23] LI, C. et al. Convolutional Neural Networks for Medical Diagnosis from Admission Notes. *arXiv:1712.02768 [cs]*, 2017.

[24] HUANG, J.; OSORIO, C.; SY, L. W. An Empirical Evaluation of Deep Learning for ICD-9 Code Assignment using MIMIC-III Clinical Notes. *Computer Methods and Programs in Biomedicine*, v. 177, p. 141–153, Aug. 2019.

[25] AYYAR, S.; BEAR DON'T WALK IV, O. Tagging Patient Notes With ICD-9 Codes. In: Proceedings of the 29th NIPS. c2016.

[26] XIE, P.; XING, E. A Neural Architecture for Automated ICD Coding. In: Proceedings of the 56th ACL. Association for Computational Linguistics, c2018. v. 1. p. 1066–1076.

[27] SALTON, G.; BUCKLEY, C. Term-Weighting Approaches in Automatic Text Retrieval. *Information Processing & Management*, v. 24, p. 513–523, 1988.

[28] MIKOLOV, T. et al. Efficient Estimation of Word Representations in Vector Space. In: Proceedings of the ICLR Workshop. c2013.

[29] PENNINGTON, J.; SOCHER, R.; MANNING, C. Glove: Global Vectors for Word Representation. In: Proceedings of the 2014 EMNLP. Association for Computational Linguistics, c2014. p. 1532–1543.

[30] BOJANOWSKI, P. et al. Enriching word vectors with subword information. *TACS*, 2016.

[31] PETERS, M. E. et al. Deep Contextualized Word Representations. In: Proceedings of the 2018 NAACL-HLT. Association for Computational Linguistics, c2018. v. 1.

[32] DEVLIN, J. et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: Proceedings of the NAACL-HLT 2019. c2019.

[33] ZHANG, X.; ZHAO, J.; LECUN, Y. Character-level Convolutional Networks for Text Classification. In: Proceedings of the 28th NIPS. c2015. v. 1. p. 649–657.

[34] LE, Q. V.; MIKOLOV, T. Distributed Representations of Sentences and Documents. In: Proceedings of the 31st ICML. c2014.

[35] GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[36] CHUNG, J. et al. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. In: . c2014.

[37] VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, L. U.; POLOSUKHIN, I. Attention Is All You Need. In: Proceedings of the 31st NIPS. Long Beach, California, USA: Curran Associates Inc., c2017. p. 6000–6010.

[38] Christopher D. Manning, Prabhakar Raghavan, H. S. *Introduction to information retrieval*. 2008. ISBN: 0521865719.

[39] LANE, H.; HAPKE, H.; HOWARD, C. *Natural language processing in action: Understanding, analyzing, and generating text with python*. Manning Publications, 2019.

[40] KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization, 2017.

[41] IOFFE, S.; SZEGEDY, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In: Proceedings of the 32nd ICML. c2015. v. 37.

APÊNDICE A — INTERPRETAÇÃO DE MÉTRICAS

Os valores possíveis para as métricas apresentadas na Seção 4.4 partem de 0 até um fundo de escala de 1. Uma *Precision* de 0 indica que todas as predições positivas feitas por um modelo foram incorretas, e 1 que todas as predições positivas feitas foram corretas (isto é, nenhum rótulo falso foi predito como verdadeiro, embora possam existir ainda rótulos verdadeiros que foram preditos como falsos). Um valor de 0 para *Recall* indica que todas as predições positivas foram incorretas, enquanto 1 indica que o modelo foi capaz de incluir nas predições positivas todos os rótulos reais (ainda que possivelmente tenha também incluído nas predições positivas rótulos incorretos). Finalmente, um valor de 0 para F1 indica novamente que todas as predições feitas por um modelo foram incorretas, enquanto o fundo de escala 1 indica não apenas que todas as predições positivas foram corretas, mas também que todos os rótulos verdadeiros foram preditos como positivos pelo modelo (ou seja, o modelo acertou todas as predições positivas e negativas).

Tomemos como exemplo um problema com $N = 5$ amostras e $C = 3$ classes possíveis. Sejam os rótulos conhecidos representados pela matriz $\mathbf{Y} \in \mathbb{R}^{\mathbb{N}\times\mathbb{C}}$ em que cada linha é uma amostra diferente e cada coluna uma classe, e as saídas de um certo modelo representadas por $\tilde{\mathbf{Y}} \in \mathbb{R}^{\mathbb{N}\times\mathbb{C}}$:

$$\mathbf{Y} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix}, \; \tilde{\mathbf{Y}} = \begin{bmatrix} 0,45 & 0,12 & 0,01 \\ 0,60 & 0,02 & 0,50 \\ 0,38 & 0,07 & 0,23 \\ 0,01 & 0,51 & 0,55 \\ 0,40 & 0,49 & 0,09 \end{bmatrix}. \tag{29}$$

Para a quantização de $\tilde{\mathbf{Y}}$ (veja Seção 4.3), tomemos um limiar de binarização de 0,30 para obter $\hat{\mathbf{Y}}_{t=0,30}$, e um número fixo de k=2 predições positivas por amostra para obter $\hat{\mathbf{Y}}_{k=2}$. Podemos construir matrizes $\mathbf{M} \in \mathbb{R}^{N\times C}$ classificando cada predição $\hat{y}_n^c$ como verdadeiro postiva (TP), falso positiva (FP), verdadeiro negativa (TN) e falso negativa (FN):

$$\hat{\mathbf{Y}}_{t=0,30} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix}, \mathbf{M}_{t=0,30} = \begin{bmatrix} TP & TN & TN \\ TP & FN & FP \\ FP & TN & FN \\ FN & FP & FP \\ TP & TP & TN \end{bmatrix}, \tag{30}$$

$$\hat{\mathbf{Y}}_{k=2} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix}, \mathbf{M}_{k=2} = \begin{bmatrix} TP & FP & TN \\ TP & FN & FP \\ FP & TN & TP \\ FN & FP & FP \\ TP & TP & TN \end{bmatrix}. \tag{31}$$

Para computar as métricas em ponderação micro, basta somar globalmente as diferentes classificações da tabela $\mathbf{M}_{t=0,30}$, para obter *Precision*, *Recall* e F1 segundo as Equações 26, 27 e 28:

$$Precision_{micro} = \frac{4}{4+4} = 0,50, \tag{32}$$

$$Recall_{micro} = \frac{4}{4+3} = 0,571, \tag{33}$$

$$F1_{micro} = \frac{2 \cdot 0,50 \cdot 0,571}{0,50 + 0,571} = 0,533, \tag{34}$$

e fazer o mesmo a partir de $\mathbf{M}_{k=2}$ para obter *Recall*@2:

$$Recall@2_{micro} = \frac{5}{5+2} = 0,714. \tag{35}$$