Deep Learning para auxílio ao combate a violência de gênero: Uma representação de dados públicos usando rede Autoencoder

Ana Luiza S. Jornada[1], **Aline Duarte Riva**[2]

[1]Ciência da Computação – Universidade La Salle
Canoas – RS – Brazil

[2]Ciência da Computação – Universidade La Salle
Canoas – RS – Brazil

{ana.jornada0489,aline.riva}@unilasalle.edu.br

Abstract. *This paper presents the development of a autoencoder neural network model that allows the creation of representations about the data of violence against women that contribute to the identification of patterns, helping to characterize the profile of the aggressors, aiming to assist in the analysis information in a more intuitive way and in combating violence.*

Resumo. *Este trabalho apresenta o desenvolvimento de um modelo de rede neural autoencoder que possibilita a criação de representações sobre os dados de violência à mulher que contribuem para a identificação de padrões ajudando a caracterizar o perfil dos agressores, tendo como objetivo auxiliar na análise das informações de forma mais intuitiva e no combate a violência.*

1. Introdução

Mulheres são vítimas de muitas formas de violência que violam os princípios de igualdade de direito entre pessoas. O comitê para a Eliminação da Discriminação contra a Mulher (CEDAW) identificou a violência baseada no gênero, cuja causa principal é a desigualdade de gênero [OACNUDH 2020], como um dos principais fenômenos sociais que torna desigual o nível de poder entre homens e mulheres.

O feminicídio é tipificado pela morte violenta ou assassinato de mulheres fundamentada em seu gênero e está relacionada à hierarquização social dos sexos. Sendo um mecanismo de dominação e extinção daquelas mulheres que não agem da forma esperada pela sociedade culturalmente patriarcal [OACNUDH 2020].

No levantamento feito pelo Mapa da Violência em 2015 o Brasil foi considerado o 5º país com maior número de feminicídios do mundo [Waiselfisz 2012], o crime é diretamente relacionado com a violência de gênero, neste caso quando a vítima é morta por ser mulher. Embora o Brasil tenha tomado algumas ações como: Lei do Feminicídio e a Lei Maria da Penha, ainda existem muitas dificuldades, manter as mulheres seguras depois de realizar denúncia contra o companheiro ou familiar, prover uma punição apropriada para autores de crimes de violência contra mulheres e ajudar as vítimas que não possuem renda devido às violências sofridas anteriormente, são alguns dos desafios a serem superados.

Em 2020 foi imposto o isolamento social pela COVID-19, que causa a coexistência forçada da vítima com seu agressor, o que ocasionou aumento de casos de

violência doméstica em vários países. Segundo a Ouvidoria Nacional dos Direitos Humanos (ONDH), do Ministério da Mulher, da Família e dos Direitos Humanos (MMFDH), entre os dias 1º e 25 de março no Brasil houve um aumento de 18% no números de denúncias registradas pelos serviços Disque 100 e Ligue 1808 [Vieira et al. 2020].

Dentro deste contexto e utilizando estatísticas de crimes violentos contra mulheres foi desenvolvida uma ferramenta baseada em *Deep Learning* que cria um novo espaço de representação para essas estatísticas, facilitando a identificação de padrões e ajudando a caracterizar o perfil dos agressores. A simplificação na exposição dos dados sobre violência contra mulher, gerada pela ferramenta, tem potencial de se tornar um grande aliado na criação de ações, projetos, pesquisas relacionadas a este assunto.

Um trabalho com referencial ao tema é [Subramani et al. 2019] que utilizou *Deep Learning* na classificação de situações críticas de denúncias online ou pedido de ajuda financeira em post nas redes sociais, o problema mencionado é que muitas vezes esses pedidos se perdem ao meio de outros posts na rede social e com essa tecnologia é possível fazer a categorização automática dos conteúdos auxiliando para a intervenção instantaneamente nos casos urgentes reduzindo a perda desses pedidos de ajuda. Outro artigo referencial que utilizou o mesmo conceito foi o [Yallico Arias and Fabian 2022] que propôs um modelo classificador para identificar os níveis de violência psicológica contra a mulher analisando a conversar do casal focando nas mensagens que o homem envia para a mulher e 'alertá-la' sobre o risco que ela corre nesse relacionamento.

O problema de pesquisa que trouxe a necessidade da aplicação de uma ferramenta como aqui apresentada foi a dificuldade de criar medidas e projetos que possam prover segurança para as mulheres de forma antecipativa como também garantir ajuda as vítimas da violência.

Tendo em vista esse problema de pesquisa o objetivo deste trabalho é analisar os dados disponibilizados pelo governo, órgãos públicos e instituições de pesquisa referente a casos de violência contra mulher no território brasileiro com base em estudos disciplinados através da escolha de modelos de representação e modelagem estatística acerca destes gerar informações com maior precisão de forma representativa que possam auxiliar na construção de projetos assertivos no combate à violência contra a mulher e na criação de *insights* futuros para a solução deste problema social.

2. Metodologia

Para o desenvolvimento deste projeto foi utilizada uma metodologia exploratória visto que busca um padrão para casos de feminicídio e violência contra a mulher através de representações com base em dados para identificar os agressores. Esta pesquisa tem como fonte de dados o estudo de indicadores reportados pelo projeto EVA [Igarape 2020], a avaliação foi feita somente sobre dados do Brasil na área da saúde, sem uma comparação com resultados de outros países.

As informações presentes nos dados foram exploradas de maneira quantitativa tendo como objetivo desenvolver gráficos e relatórios para mostrar o resultado da pesquisa. É importante destacar que este trabalho foi desenvolvido considerando algumas limitações, no Brasil ainda há poucos dados disponíveis para a sociedade referentes à violência contra a mulher e feminicídio e por vezes quando disponíveis não possuem

muitas informações, portanto as representações usadas são restritas às disponíveis no momento.

O método escolhido para a representação dos dados foi uma técnica de *Deep Learning* com a qual é possível criar uma ferramenta que reduz a dimensionalidade dos dados possibilitando o fácil agrupamento dos mesmos e possibilitando encontrar um padrão de comportamento no resultado.

A ferramenta foi desenvolvida na linguagem *Python*, utilizando a técnica de aprendizado não supervisionado com o uso de modelos de redes neurais para a tarefa de aprendizado de representação [Bengio et al. 2017]. O objetivo do modelo foi criar uma representação com menor número de dimensões. Criando, assim, propriedades úteis sobre estes dados que facilitam a análise de dados sobre eles. Para instrumentar a execução do modelo foram utilizadas as bibliotecas *numpy*, *pandas* e *sklearn* que facilitam os cálculos estatísticos sobre os dados. O desenvolvimento do código e alguns testes preliminares foram realizados na ferramenta *Google Colab*, que possibilita armazenar e executar o modelo em nuvem. O *Google Colab* é de fácil uso tanto para fase de codificação quanto para a organização de comentários e fases do projeto. Porém, a versão gratuita possui limitações de memória *RAM*, tempo de execução e armazenamento. Por estas razões a execução final do modelo foi realizada em máquina local.

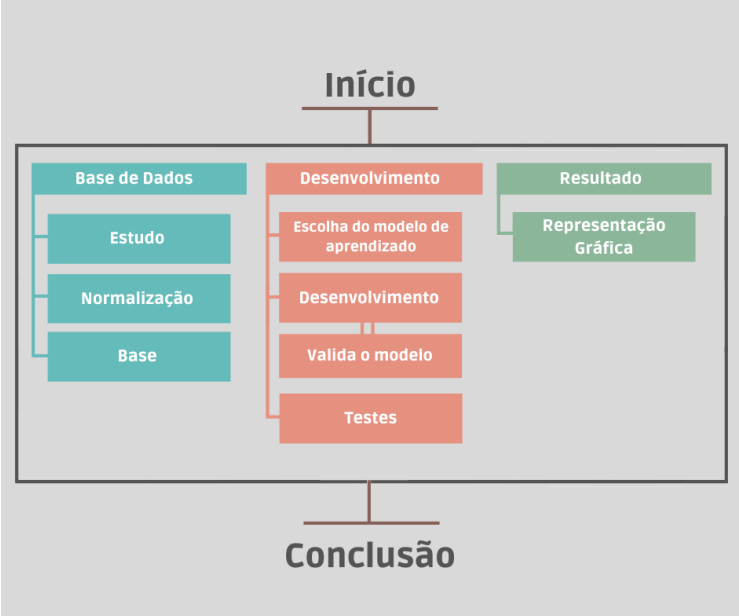


Figura 1. Fluxo de processo

O projeto seguiu as etapas conforme Figura 1 iniciando pelo estudo, normalização e geração da base final de dados utilizada no processo de treino, apresentado na seção 3, em sequência o desenvolvimento da rede neural onde foi feita a escolha do modelo, desenvolvimento, validação do modelo e testes, todos esses pontos aborda-se na seção 3.1, e por fim o resultado apresentado graficamente, que demostra-se na seção 4.

3. Desenvolvimento

O primeiro passo para o desenvolvimento da ferramenta foi a busca das informações na fonte utilizando a função *requests.get()* que possibilita fazer a requisição para uma *url* e baixar as informações necessárias em um objeto *response* e carregar a base de dados para o projeto . Como o arquivo retorna um *zip* foi feita a extração do arquivo e posterior leitura do *saude_geral.csv* que contém a base original.

Partindo dessa base foi necessário analisar quais informações seriam utilizadas. Neste ponto foi necessário iniciar o processo de limpeza dos dados que foi dividido em tarefas.

1. Remover colunas nulas: é feita a exclusão de todas as colunas que possuem todas as suas linhas nulas;
2. Remover colunas que não fazem parte do estudo: aqui somente foram mantidas as colunas que possuem dados relevantes para a pesquisa. Foram excluídas as colunas população, agressão_sexual, cod_estado, cod_município, nível_do_dados, município, tipo_base_de_dados, sexo, taxa, violência_doméstica, país;
3. Normalizar Nulos: Algumas das colunas que se mantém na base para o estudo possuem dados nulos que precisam ser normalizados. Todas as informações nulas foram substituídas pela informação 'Não Especificado' mantendo o padrão da base original.

Após o processo de limpeza e normalização dos dados as informações que serão usadas são pontuadas na Tabela 1.

Tabela 1. Informações utilizadas no estudo

Coluna	Informações
Agressor	Companheiro(a), Conhecido(a), Desconhecido(a), Ex-Companheiro(a), Irmão(a), Mãe, Não Especificado, Outro Parente, Pai
Ano	Entre 2000 e 2017
Arma	Agressão Por Outras Causas Arma de Fogo, Força Corporal, Não Especificado, Objeto Contundente, Objeto Cortante, Produtos Químicos e Substâncias Nocivas, Sem Arma
Estado	Todos os estados do Brasil
Faixa Etária	0-14, 15-29, 30-44, 45-65, 65+
Ocorrência	Homicídio Doloso, Violência Física, Violência Patrimonial, Violência Psicológica, Violência Sexual
Raça	Amarela, Branca, Indígena, Não Especificado, Parda, Preta

No arquivo original para cada linha existe a informação de quantidade de casos, esta informação foi aberta em nível de linhas, ou seja, dado o número x de recorrências

x novas linhas foram criadas e um novo arquivo criado, chamado base_estendida que foi utilizado no estudo.

Todos os dados escolhidos para a base utilizada são categóricos e para que a análise dos dados fosse utilizada no modelo todas as informações precisaram ser transformados. Isso foi feito através da função *OneHotEncoder* da biblioteca *sklearn.preprocessing*, esta função transforma as informações *string* (texto) em uma matriz *one-hot* numérica criando um vetor binário para cada categoria e retornando uma matriz esparsa.

3.1. Autoencoder

O modelo de rede neural escolhido é conhecido pelo nome *autoencoders* (AE) e tem como objetivo replicar os dados de entrada para a sua saída, tendo como intuito aprender representações dos dados e reduzir dimensionalidades. Ele é dividido em duas partes um *encoder* que tem como função comprimir a informação da entrada e um *decoder* que faz o trabalho inverso e reconstrói a informação original [Kusner et al. 2017] sendo:

- *encoder*: uma função f(x) que transforma o *input* para uma representação h.
- *decoder*: uma função g(x) que transforma a representação h em sua reconstrução r [Bengio et al. 2017].

Na Figura 2 é possível ver a relação da entrada, representação e sua reconstrução passando por toda a rede.

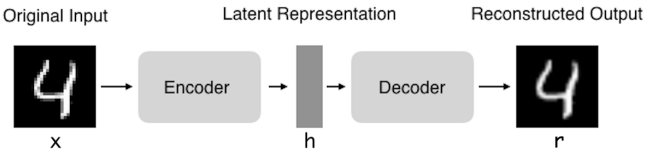


Figura 2. *Deep Learning Book* [Bengio et al. 2017]

Neste trabalho utilizou-se o *autoencoder* para gerar informações que possam representar os dados do *input*. A rede AE é treinada de maneira não supervisionada para que não tenha a necessidade de um *target*, podendo construir a informação com bases em suas *features*, visto que aqui objetivo é criar representações da entrada em um espaço-latente diferente que assuma propriedades úteis e não gerar uma informação preditiva sobre uma variável.

Na Figura 3 é apresentada a visão de como essa rede neural funciona. Sempre tem-se os dados de entrada, cada nodo da entrada está densamente conectado a todos os nodos da próxima camada e cada uma dessas conexões possui um peso diferente e próprio, sendo que todas as camadas entre a entrada e saída são chamados de camadas escondidas (*hidden nodes*) justamente porque é o que não conseguimos enxergar de maneira clara. A primeira parte tem-se o *encoder* onde está o dado codificado até o meio da rede onde fica o espaço latente (menor camada), e depois disso ele começa a ser decodificado até gerar a saída que é uma representação do dado de entrada.

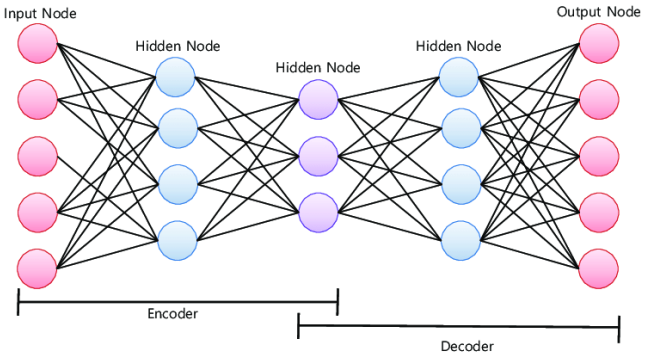


Figura 3. *Deep Learning Book* [Bengio et al. 2017]

3.2. Construção do *Autoencoder*

Algumas variáveis importantes são definidas no início da criação do modelo. No Código Fonte 1 tem-se a representação do código que define estas variáveis. O *enconding_dim* que representa o número de dimensões de h , define a dimensionalidade do novo espaço de representação. Para obter recursos úteis é comum restringir as dimensões de h para que sejam menores que x (*input*), neste caso o *autoencoder* é chamado de incompleto, para este cenário optou-se pelo modelo aprender apenas os recursos relevantes, removendo informações vazias, colunas repetidas e informações incompletas evitando que ele apenas copie a entrada para a saída [Bengio et al. 2017]. O *num_colums* especifica o número de itens da tabela de entrada, que se refere ao número total de colunas que representa este número. E o *input_data* é o marcador de entrada que gera o tensor que especifica quantos dados a rede deve esperar.

Código Fonte 1. Código que define as variáveis.

```
encoding_dim = 2
num_columns = 86
input_data = Input(shape=(num_columns,))
```

Para o desenvolvimento do modelo foram criadas algumas camadas (*layers*) densas, que são as camadas de neurônios em uma rede neural. Foram criadas 6 camadas para o *encoder* que comprimem as informações e 5 camadas *decoder* que reconstroem as informações, essas camadas foram definidas de forma empírica com base na experiencia prática durante a execução do algoritmo, onde foi avaliada sua performance após apresentação do resultado. No Código Fonte 2 é apresentado o código e estrutura usada para criar as camadas. Para a função *Dense()* é definida a função de ativação dos neurônios *relu* que aplica a função de ativação da unidade linear retificada. Com a relu é feito o mapeamento do resultado da soma ponderada das entradas do neurônio para sua saída, seguindo uma função que para valores menores ou iguais a zero a saída assume valor zero e para valores maiores que zero o resultado é utilizado como saída. Para a última camada do *decoder* a função de ativação *sigmoid* que faz a conversão dos números reais para a valores na faixa 0 a 1 [Hassan1&2 and Akamatsu 2004].

Código Fonte 2. Gera os *layers*.

```
encoded = Dense(num_columns, activation='relu')(input_data)
encoded = Dense(40, activation='relu')(encoded)
encoded = Dense(20, activation='relu')(encoded)
encoded = Dense(10, activation='relu')(encoded)
encoded = Dense(4, activation='relu')(encoded)

encoded = Dense(encoding_dim, activation='relu')(encoded)

decoded = Dense(4, activation='relu')(encoded)
decoded = Dense(10, activation='relu')(encoded)
decoded = Dense(20, activation='relu')(decoded)
decoded = Dense(40, activation='relu')(decoded)
decoded = Dense(num_columns, activation='sigmoid')(decoded)
```

Após a criação das camadas da rede é criado o modelo *encoder* e o *autoencoder* que modela a rede completa. A função compile é usada para definir a otimização, perda e a métrica a ser usada na rede. No Código Fonte 3 apresentada a estrutura do algoritmo desenvolvido para esta parte da rede.

Código Fonte 3. Criação dos modelos.

```
encoder = Model(input_data, encoded)
autoencoder = Model(input_data, decoded)

autoencoder.compile(  optimizer='adamax',
                      loss='CategoricalCrossentropy',
                      metrics=['categorical_accuracy'])
```

Os parâmetros utilizados para o *compile* são:

- *Optimizer*: otimiza os pesos de entrada ao comparar a representação e a função de perda, o *adamax* foi escolhido pois ele é mais robusto no tratamento de ruídos durante a atualização dos pesos, como também possui uma estabilidade numérica maior [Bilogur 2020].
- *Loss*: é usado para identificar erros ou desvios no processo de aprendizado. A *Categorical Crossentropy* é usada em tarefas de classificação de várias classes, isso significa que quando o algoritmo precisa escolher entre muitas categorias esse modelo irá decidir qual delas escolher [Koidl 2013].
- Metrics: para avaliar o desempenho da rede foi adicionado o *categorical accuracy* que calcula a taxa média de precisão toda a vez que o *input* passa pela rede [Faroit 2020].

Neste ponto tem-se a rede *autoencoder* definida e criada para ser utilizada no processo de treino.

3.3. Reprocessando os dados

Para realizar o treinamento do modelo é necessário transformar os dados novamente em uma matriz esparsa, formato esperado pela rede.

No Código Fonte 4 encontra-se o código onde o x *train* e o x *test* são transformados em um *dataframe* da posição 0 até 100% do conteúdo da base. Tendo esse *dataframe* é possível usar a função *todense* para gerar a matriz esparsa do dado.

Código Fonte 4. Reprocessa os dados.

```
p = enc.transform(x_train[0:round(len(x_train)*1)])
x = p.todense()

p2 = enc.transform(x_test[0:round(len(x_test)*1)])
xteste = p2.todense()
```

3.4. Treinamento do modelo

É utilizada a função *fit* para realizar o treinamento da rede. Como mostra o Código Fonte 5 existe um número de parâmetros necessários para que o modelo seja treinado de maneira correta.

Código Fonte 5. Treinamento dos dados.

```
history = autoencoder.fit(x, x,
                epochs=100,
                batch_size=64,
                shuffle=True,
                validation_data=(xteste, xteste)
               )
```

Passamos o dado que será treinado e o queremos como saída, neste caso o próprio x, as *epochs* (épocas) que representam o número de vezes que o dado irá passar pela rede, o *batch size* que define a frequência, baseada no número exemplos de treinamento (linhas do dataset), em que os pesos são atualizados. O *shuffle* que garante que os dados serão misturados entre si, não seguindo sempre a mesma sequência nas diversas épocas de treinamento e por último a *validation* data que define os dados sobre quais a avaliação será feita e qualquer outra métrica do modelo no final de cada época executada [Rstudio 2020].

Após treinar o modelo, usamos para criar a representação dos dados a função *predict* passando nossa variável que possui os dados de teste (xteste), ela retorna uma matriz com as representações que foi capaz de criar conforme mostra o Código Fonte 6.

Código Fonte 6. Dados de representação.

```
encoded_data = encoder.predict(xteste)
decoded_data = autoencoder.predict(xteste)
```

É aplicado tanto o *predict* da parte do *encoder* como de toda a rede *autoencoder*. Para a representação usaremos o *encoder* onde os dados estão representados com dimensionalidade reduzida. O código desenvolvido pode ser encontrado no link: *bit.ly/bjsR1*.

4. Resultados

Para encontrar um melhor resultado para a base de dados da representação foi necessário realizar testes na tentativa de encontrar valores em que a perda reduzisse e a precisão aumentasse durante a execução da rede neural e que também trouxesse dados representativos consistentes. Estes testes aconteceram com a execução da rede neural n vezes mudando as métricas do compilador e do *fit,* conforme o resultado obtinha melhora as alterações eram apenas refinadas até um ponto de finalização.

Ao escolher o melhor resultado foi desenvolvida a versão que possibilita a análise. Para a construção desse *dashboard* foi utilizada a ferramenta *Power BI Desktop* onde foram criados gráficos que demonstram os resultados e em que é possível detectar informações relevantes com a violência contra mulheres. O processo de desenvolvimento do *dashboard* começa na criação do documento que o algoritmo gera, neste projeto foi gerado um arquivo de extensão .csv que compacta as informações que são facilmente lidas pelo *Power BI*.

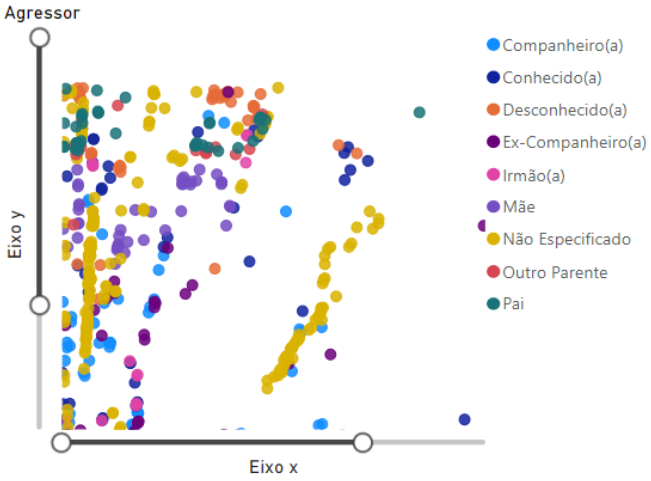
No Código Fonte 7 o código utilizado para gerar as informações necessárias para o desenvolvimento visual é demonstrada. Após, é realizado um tratamento dentro da ferramenta: ajuste do tipo da coluna de texto para numérico, remoção da primeira linha (contém apenas número da coluna) e alteração na nomenclatura das colunas conforme necessidade.

Código Fonte 7. Gera arquivo com os resultados.

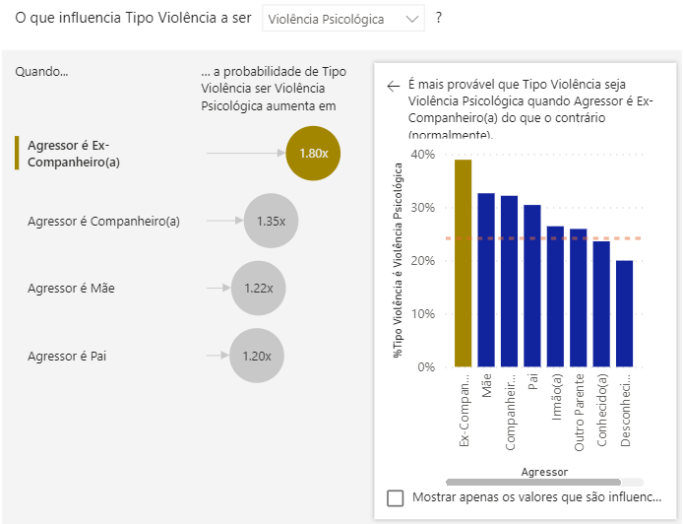
```
r = pd.DataFrame(
                    np.concatenate(
                        (enc.inverse_transform(xteste),encoded_data)
                    ,axis=1)
                )
r.to_csv('Dados/base_analise.csv',index=False)
```

Após esses passos, foi desenvolvido o *dashboard*, ele foi dividido em duas partes, uma com gráficos de dispersão nos quais é possível estudar os agrupamentos formados pelos dados de saída da rede AE e identificar a relação entre os agressores levando em consideração o perfil de cada um, por exemplo identifica-se que existe a proximidade ou agrupamento de pai e irmão dentro do eixo x e y, portanto existem propriedades(*features*) que definem sua proximidade quanto ao perfil de agressor, neste cenário teríamos uma indicação de que os dois possuem propriedades relacionadas em que o analista poderia interpretar o que ocasiona essa proximidade. Com esta visualização podemos diagnosticar

quais seriam os próximos passos em termos de pesquisa, projeto ou desenvolvimento de novas análises.



(a) Representação dos dados em gráfico de dispersão.



(b) Gráfico de principais influenciadores.

Figura 4. Representação dos dados em gráfico de dispersão.

A segunda parte possui um gráfico para otimizar a análise e demonstra-se a probabilidade de uma informação ser influenciada por outra, no caso demostrado tem-se a possibilidade de encontrar estatísticas de probabilidade em relação ao tipo de violência para cada agressor, podemos entender com base no gráfico a probabilidade do tipo de violência x ser o agressor y dentre todos os tipos de agressores, sendo mais fácil provar e argumentar sobre o agressor em potencial e para a melhorar a projeção de uma solução que evite tais agressões. Na Figura 4 é possível ver os dois gráficos com os dados gerados, conforme explicado anteriormente nesta seção.

É importante ressaltar que a demonstração dos dados pode ser alterada dependendo da necessidade do analista, pode-se ter outros gráficos e mais detalhes de informações. É importante manter em mente que o *dashboard* deve atender as necessidades do analista que ficara responsável pela gestão e utilização dos dados.

É importante também mencionar o desempenho da rede, a Figura 5 mostra o gráfico que compara o *val_loss* (perda de treinamento) e o *loss* (perda de validação) sendo os valores de perda da rede, é possível perceber, portanto que a perda do treinamento está reduzindo de maneira suave que indica que o modelo está melhorando à medida que está sendo treinado.

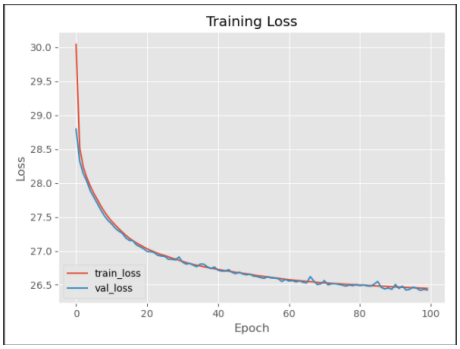


Figura 5. Gráfico comparativo train loss e val loss

Destaca-se que a rede passou por 100 épocas, caso continuasse a ser executada com mais épocas a tendência é de que o *train loss* continue a reduzir até ficar o mais próximo de 0. Os resultados obtidos alcançaram uma representação satisfatória para o uso em análises deste tema, podendo ser utilizado como ferramenta que possibilita a análise de forma mais intuitiva para pessoas das áreas sociais.

5. Considerações Finais

Com o desenvolvimento da ferramenta apresentada neste trabalho, almeja-se que as análises, por ela viabilizada, sobre o tema proposto possam auxiliar tanto no desenvolvimento de medidas assertivas no combate à violência contra a mulher, como também tornar os dados visíveis para uma maior parte da sociedade, sendo assim criando oportunidade para que ações necessárias possam acontecer.

Através deste trabalho pessoas que tenham interesse em otimizar o combate a esta violência tão cruel poderão acessar mais facilmente as estatísticas e mapear caminhos para uma melhor aplicação de práticas e políticas voltadas ao combate da violência de gênero.

Referências

Bengio, Y., Goodfellow, I., and Courville, A. (2017). *Deep learning*, volume 1. MIT press Massachusetts, USA:.

Bilogur, A. (2020). Keras optimizers. Disponível em: `https://www.kaggle.com/residentmario/keras-optimizers` Acesso em: 10 Set. 2020.

Faroit (2020). Usage of metrics. Disponível em: `https://faroit.com/keras-docs/1.2.0/metrics/` Acesso em: 11 Set. 2020.

Hassan1&2, N. and Akamatsu, N. (2004). A new approach for contrast enhancement using sigmoid function.

Igarape (2020). Eva. Disponível em: https://eva.igarape.org.br/ Acesso em: 05 Ago. 2020.

Koidl, K. (2013). Loss functions in classification tasks. *School of Computer Science and Statistic Trinity College, Dublin*.

Kusner, M. J., Paige, B., and Hernández-Lobato, J. M. (2017). Grammar variational autoencoder. In *International Conference on Machine Learning*, pages 1945–1954. PMLR.

OACNUDH (2020). Modelo de protocolo latino-americano de investigaçao das mortes violentas de mulheres por razões de gênero (femicídio/feminicídio). Disponível em: http://www.onumulheres.org.br/wp-content/uploads/2015/05/protocolo_feminicidio_publicacao.pdf Acesso em: 05 Mai. 2020.

Rstudio, K. (2020). Train a keras model. Disponível em: https://keras.rstudio.com/reference/fit.html Acesso em: 08 Set. 2020.

Subramani, S., Michalska, S., Wang, H., Du, J., Zhang, Y., and Shakeel, H. (2019). Deep learning for multi-class identification from domestic violence online posts. *IEEE Access*, 7:46210–46224.

Vieira, P. R., Garcia, L. P., and Maciel, E. L. N. (2020). Isolamento social e o aumento da violência doméstica: o que isso nos revela? *Revista Brasileira de Epidemiologia*, 23:e200033.

Waiselfisz, J. J. (2012). Mapa da violência. *CEP*, 2722:000.

Yallico Arias, T. and Fabian, J. (2022). Automatic detection of levels of intimate partner violence against women with natural language processing using machine learning and deep learning techniques. In Lossio-Ventura, J. A., Valverde-Rebaza, J., Díaz, E., Muñante, D., Gavidia-Calderon, C., Valejo, A. D. B., and Alatrista-Salas, H., editors, *Information Management and Big Data*, pages 189–205, Cham. Springer International Publishing.