Automatic Identification of Postings Related to the Use Through Deep Learning Models

Hismael Costa de Oliveira[1], **Alexandre Matos Arruda**[1], **Marília Soares Mendes**[1]

[1]Universidade Federal do Ceará (UFC)
Russas – CE – Brasil

hismael.costa@alu.ufc.br, {marilia.mendes, aldufc}@ufc.br

Abstract. *Some posts in Social Systems (SS) can bring important informations about the use of that SS, this we call Postings Related to the Use (PRU). The PRU are important tolls to systems evaluate, in this case textual evaluation. There are softwares ables to identify PRU, however they have small precision when compared to manual classifications. Given this problem, the present work seeks to overcome this deficiency by using deep learning models for the automatic identification of PRU in SS, in this case the social network Twitter. In performed tests, our model has higher precision when compared to anothers used for this work.*

Resumo. *Algumas postagens em Sistemas Sociais (SS) podem trazer informações significativas sobre o uso daquele SS, estas chamamos de Postagens Relacionadas ao Uso (PRU). PRU é um conceito importante para a avaliação de sistemas. Existem ferramentas capazes de identificar as PRUs, porém possuem menor precisão quanto as classificações manuais. Tendo em vista este problema, o presente trabalho busca investigar essa deficiência utilizando modelos de aprendizagem profunda para a identificação automática de PRUs nos SS, neste caso a rede social Twitter. Nos testes realizados nosso modelo possuiu alta precisão de classificação quando comparado com outros utilizados para esta tarefa.*

1. Introdução

Comunicação é uma parte essencial da vida humana, através dela nos relacionamos com todo o universo de informação que nos rodeia. Com o advento da Web 2.0, emergiu, em uma escala gigantesca, aplicações focadas na comunicação, colaboração e interação entre usuários [Pereira et al. 2010]. Um exemplo dessas aplicações são os Sistemas Sociais (SS), eles permitem a comunicação, diversificação de ideias e informação entre os usuários, facilitando qualquer forma de relacionamento social e fazendo com que os próprios usuários produzam conteúdo. Os SS são amplos, desde sites de colaboração de conteúdo até redes sociais.

A maior forma de interação nesses sistemas é feita através de postagens, podendo ser publicas ou privadas, que podem lidar com diversos assuntos. Várias dessas postagens podem estar relacionadas ao pensamento do usuário sobre o uso do sistema naquele momento, por exemplo: "*É tão chato ter apenas 140 caracteres para postar no twitter :(*". Estas postagens são chamadas de **Postagens Relacionadas ao Uso** (PRU) [Mendes 2015].

Elas são utilizadas nas técnicas de avaliação textual do sistema [Mendes 2015], que serão detalhadas mais à frente.

Como qualquer tipo de automação, a identificação automática de PRU reduziria drasticamente o tempo, já que o computador consegue fazer essa tarefa em milissegundos, além de custos com pessoas e trabalho.

A dificuldade desta tarefa deve-se ao fato das PRUs não possuírem uma regra básica de classificação, como por exemplo um substantivo ou adjetivo que auxilie nesta tarefa. A sua identificação se dá por pequenas distinções entre outras postagens. Assim, todo o contexto da sentença deve ser levado em consideração como também características do SS em questão. Portanto, é necessário identificar características que não estão evidentes nas postagens.

Diversas estratégias foram utilizadas para identificar automaticamente as PRUs, como por exemplo: a busca booleana proposta por [Mendes 2015], utilizando técnicas de aprendizado de máquina e linguagem natural [Hedegaard and Simonsen 2013] ou através de técnicas de mineração de dados [Lima et al. 2017]. O problema deve-se ao fato destas abordagens não possuírem alta precisão quando comparada com a classificação manual [Mendes and Furtado 2017].

Aprendizagem profunda é uma área da aprendizagem de máquina capaz de extrair automaticamente características importantes das entradas. Cada arquitetura possui uma forma diferente de extração, seja através de propagação temporal, como as *Recurrent Neural Networks* (RNN), utilizando filtros nas entradas, como as *Convolutional Neural Networks* (CNN), como também através da competição entre duas ou mais redes neurais, como as *Generative adversarial network*. Um tipo de RNN interessante é a *Bidirectional Long Short-term Memory* (BLSTM), a sua diferença em comparação à uma RNN é a capacidade de propagação temporal tanto para entradas já passadas pela rede como para as que ainda irão passar aliada a sua aptidão de armazenar grandes dependências de tempo.

Neste trabalho, serão utilizados filtros de convolução aliados a propagação temporal das entradas para a identificação automática de PRU, através da combinação da CNN com a BLSTM, que é chamado de C-BLSTM.

O restante do trabalho se divide em 6 seções. Na seção 2 são apresentados os principais conceitos utilizados para comparação de modelos de aprendizado. Na seção 3 são apresentados os trabalhos relacionados. Na seção 4 apresenta de forma resumida a arquitetura do C-BLSTM. Na seção 5 são apresentados os processos de coleta, limpeza e análise dos dados. Na seção 6 são apresentados os resultados dos experimentos através da comparação de diversos modelos de aprendizagem de máquina. Por fim, na seção 7 são apresentadas as conclusões e os trabalhos futuros.

2. Postagens Relacionadas ao Uso e Avaliação de Sistemas

Usabilidade é uma medida que garante que os produtos são fáceis de usar, eficientes e agradáveis do ponto de vista do usuário [Preece et al. 2015]. A avaliação de usabilidade está intrinsecamente ligada a outro fator de extrema importância, a satisfação de uso [Preece et al. 2015]. Por satisfação de uso entendemos os sentimentos, emoções e percepções do usuário, obtidos por meio de perguntas escritas ou orais [Shaw 1996]. Existem várias maneiras de avaliar a usabilidade de um sistema, entre elas podemos destacar:

questionário, entrevistas, avaliação heurística, método de inspeção semiótica, dentre outros.

Uma outra técnica utilizada para avaliação é a Textual [Mendes 2015]. Ela consiste em usar narrativas dos usuários a fim de avaliar ou obter alguma percepção sobre o sistema por meio de suas postagens. Esse tipo de avaliação faz uso de Postagens Relacionadas ao Uso (PRUs) [Mendes 2015]. Uma PRU é um texto ou comentário feito de forma espontânea publicado pelos usuários se referindo ao uso do sistema. Elas podem estar na forma de pergunta, reclamações ou até mesmo elogios sobre o sistema. Como exemplo temos a seguinte postagem extraída do Twitter:"*Twitter fica dando erro interno de servidor sempre que tento editar meu perfil –*"'.

Em [Mendes 2015], os autores analisaram as principais características das PRU mostradas através dos SS. Nesta análise foi observado que as PRUs geralmente expressam pedidos, críticas, comparações, perguntas ou fazem sugestões sobre o sistema.

[Mendes 2015] é apresentado um modelo e metodologia para avaliação da interação a partir da linguagem textual do usuário. A metodologia chamada MALTU (Modelo para Avaliação da Interação em Sistemas Sociais a partir da Linguagem Textual do Usuário) é utilizada para avaliar a experiência de uso e usabilidade de sistemas por meio de um conjunto de PRUs. As etapas são: (1) Definição do contexto de avaliação; (2) Extração de PRUs; (3) Classificação das PRUs; (4) Interpretação dos Resultados; (5) Relato dos Resultados. A classificação pode ser feita de forma automática ou por avaliadores. Na classificação por avaliadores, deve ter no mínimo três pessoas que analisam cada postagem e a classifica de acordo com as categorias sugeridas pela metodologia, porém esse processo pode levar muito tempo e tornar-se cansativo. [Freitas et al. 2016]

Existe uma ferramenta chamada UUX-Posts[1] [Mendes 2015, Mendes and Furtado 2017] que fornece suporte às etapas 2 e 3 da metodologia MALTU. A ferramenta extrai e classifica as postagens automaticamente, a fim de fornecer um resultado de avaliação. Entretanto, a precisão deste resultado ainda não é tão eficaz quanto a classificação realizada por avaliadores, pois é necessária a classificação de muitas postagens para encontrar padrões de linguagem e melhorar a classificação automática [Mendes and Furtado 2017]. A sua dificuldade de identificação das PRUs deve-se ao fato delas não seguirem um padrão sintático ou semântico explícito. Neste trabalho tentaremos identificar estes padrões através do uso de modelos de aprendizagem profunda.

3. Trabalhos Relacionados

Diversas pesquisas foram realizadas para ajudar na identificação automática de PRUs. Em [Hedegaard and Simonsen 2013] o autor extraíram os dados a partir de revisões de *video games* e *softwares* e utilizaram o algoritmo *Support Vector Machine* (SVM) para o processo de classificação. O autor conseguiu identificar padrões importantes das PRUs, entretanto a classificação possuiu uma baixa precisão.

Em [Mendes 2015] é apresentado uma busca booleana para extração automática de PRUs. Os dados utilizados foram da rede social Twitter e do sistema acadêmico, SIGAA. A autora realizou diversos testes utilizando técnicas de mineração de dados,

[1] http://uuxposts.russas.ufc.br/

como *bag-of-words* e *lemmatizer*, aplicando os dados gerados à algoritmos tradicionais de aprendizado de máquina, como SVM e Naive Bayes. Como no trabalho anterior, a autora conseguiu constatar padrões importantes nas PRUs e aplicou o processo de extração baseado nesses padrões. Entretanto o uso de algoritmos tradicionais de aprendizado de máquina auxiliaram para uma baixa precisão de identificação das PRUs.

Em [Mendes and Furtado 2017] é apresentada a UUX-Posts[2] ferramenta que fornece suporte às etapas 2 e 3 da metodologia MALTU. A ferramenta extrai e classifica as postagens automaticamente, utilizando a abordagem apresentada por [Mendes 2015], a fim de fornecer um resultado de avaliação. Entretanto, a precisão deste resultado ainda não é tão eficaz quanto a classificação realizada por avaliadores, pois é necessária a classificação de muitas postagens para encontrar padrões de linguagem e melhorar a classificação automática [Mendes and Furtado 2017].

Os trabalhos apresentados possuem baixa precisão de identificação de PRUs, utilizando algoritmos de aprendizado de máquina tradicionais para classificação. Este trabalho tem como objetivo fornecer um modelo de aprendizagem profunda capaz de identificar com alta precisão as PRUs.

4. Descrição do Modelo de Aprendizagem Profunda

A arquitetura do modelo C-BLSTM é mostrada em 1, consistindo de dois componentes principais: *Convolutional Neural Networks* (CNN) e *Bidirectional Long Short-term Memory* (BLSTM). As duas subseções a seguir descrevem a aplicação da CNN para extrair sequências de recursos de palavra de nível superior e BLSTM para capturar dependências de longo prazo sobre as sequências, respectivamente.

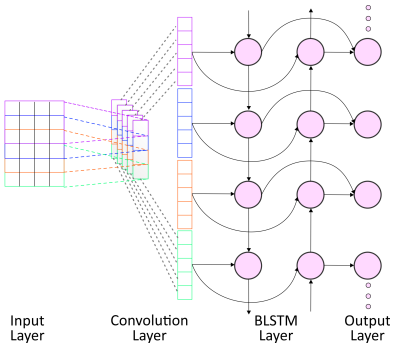

Figura 1. Arquitetura do modelo C-BLSTM
Elaborado pelos Autores (2019).

4.1. *Convolutional Neural Networks*

As CNNs são redes neurais especializadas em estruturas 2D, como texto, imagens, sons e vídeos. Em geral, sua aplicação para classificação de texto pode ser descrito como: um vetor de filtros deslizando sobre uma sequência e detectando características importantes em diferentes posições [Chunting et al. 2015].

Um conjunto de filtros convolvem com os vetores de janela em cada posição para gerar um mapa de características. Para os filtros n com o mesmo tamanho, os mapas de

[2]http://uuxposts.russas.ufc.br/

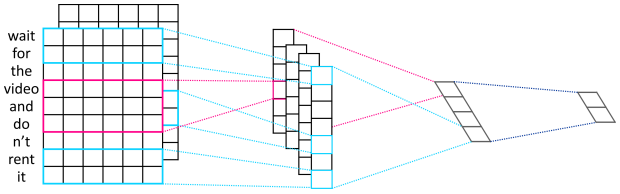

Figura 2. Arquitetura de uma CNN
Elaborado pelos Autores (2019).

características *n* gerados podem ser reorganizados como representações de recursos para cada janela. A Figura 2 ilustra este processo.

As operações de *Max-Pooling* podem ser aplicadas após o processo de convolução, a fim de selecionar os recursos mais importantes sem comprometer o processo de aprendizado da BLSTM.

4.2. *Bidirectional Long Short-term Memory*

A LSTM [Hochreiter and Schmidhuber 1997] é um tipo especial de Rede Neural Recorrente, capaz de aprender longas dependências de curto prazo. A arquitetura LSTM consiste em um conjunto de sub-redes recorrentes conectadas, conhecidas como blocos de memória. Cada bloco contém uma ou mais células auto-conectadas e três unidades multiplicativas -**input**, **output** e **forget gates**. Essas unidades permitem que as células de memória da LSTM armazenem e acessem informações por longos períodos de tempo, evitando assim o problema do gradiente de fuga.

Uma desvantagem das LSTMs convencionais é que eles só são capazes de fazer uso do contexto anterior. LSTMs Bidirecionais (BLSTM) [Schuster and Paliwal 1997] podem ser treinados usando todas as entradas disponíveis no passado e no futuro de um período de tempo específico. Isso é feito pela imobilização do neurônio de um LSTM regular em uma parte responsável pela direção positiva do tempo (estados de avanço) e uma parte pela direção de tempo negativa (estados de retrocesso), como mostrado na Figura 3. Em seguida, uma BLSTM calcula a sequência oculta para a frente, a sequência oculta para trás e a sequência de saída, iterando a camada anterior de t = T para 1, a camada direta de t = 1 para T e atualizando a camada de saída.

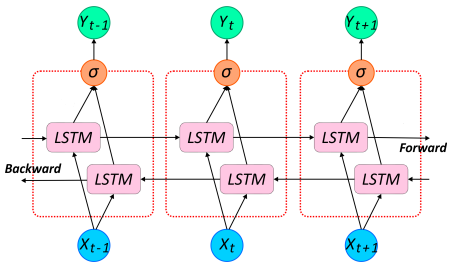

Figura 3. Arquitetura de uma BLSTM
Elaborado pelos Autores (2019).

Estas duas arquiteturas formam a base para o C-BLSTM, permitindo assim uma extração automática de características, e com resultados satisfatórios, bem como o compartilhamento delas através do tempo e com longa dependência de curto prazo.

5. Base de Dados

Os processos que precedem a classificação de dados são complexos e trabalhosos devido à falta de base de dados sobre PRUs na internet. Portanto, foram realizadas atividades de *coleta*, *limpeza* e *análise* dos dados através da frequência das palavras, estas etapas serão detalhadas nas seções 5.1, 5.2 e 5.3, respectivamente.

5.1. Coleta de Dados

Para a coleta de dados foi utilizado o sistema UUX-Posts[3][Mendes and Furtado 2017], mencionado anteriormente. Entretanto, a classificação de postagens desse sistema não possui boa precisão. Como forma de resolver esse problema foram extraídas postagens que continham a palavra *twitter*, através da classificação automática, onde foi delegado a 3 pessoas a validação da classificação.

Nesse processo uma pessoa classificava a postagem, enquanto a segunda verificava a classificação. Quando tinham ambiguidade na classificação uma terceira pessoa era chamada como critério de desempate. As pessoas eram pesquisadores da área de Interação Humano-Computador. No fim deste processo foram extraídas e verificadas 3865 postagens, sendo 2020 PRUs e 1845 Não PRUs. Nenhuma postagem foi descartada.

A base de dados coletada, as análises e os modelos de aprendizado, que serão posteriormente apresentados, estão disponíveis no repositório do *GitHub* [4].

5.2. Limpeza dos Dados

Para auxiliar neste processo foi utilizada a linguagem de programação *Python* e algumas bibliotecas especializadas para processamento textual e de dados, como pro exemplo *Pandas* [McKinney et al. 2010], e *NLTK* [Loper and Bird 2002]. Fazendo uso dessas bibliotecas e de ferramentas fornecidas pela própria linguagem de programação, foram aplicadas as seguintes técnicas de limpeza textual:

1. **Remoção de todas as menções à usuários ou *hastags***.
2. **Remoção dos *hyperlinks***.
3. **Remoção dos caracteres de pontuação**.
4. **Correção Ortográfica**. Para a realização desta etapa não foi considerado o contexto da palavra na frase, apenas sua acentuação e completude.
5. **Remoção de *stop-words***. Uma *stop-word* é uma palavra comumente usada que um algoritmo de classificação foi programado para ignorar, tanto no processo de treinamento quanto quando realiza uma classificação real. Na língua portuguesa são consideradas *stop-wors* artigos, conjunções, preposições e alguns tipos de verbo, como por exemplo, ser e estar.
6. **Padronização das palavras**. Algumas vezes as palavras podem estar em formatos inapropriados, como por exemplo *amooooooooo*. Para aumentar a precisão do nosso modelo, foi aplicado uma padronização das palavras. Após esse processo o exemplo acima iria aparecer apenas como *amo*.
7. **Redução ao radical ou base da palavra**.

[3]http://uuxposts.russas.ufc.br/

[4]http://github.com/hismael17/C-BLSTMPRU

5.3. Análise de Dados

Nesta Seção será analisada a distribuição das palavras pela frequência de apresentação nas classes PRU e Não-PRU. Primeiro será investigada a distribuição geral do comprimento das palavras em cada entrada. Para isso foi utilizado o diagrama de caixa, como pode ser visto na Figura 4.

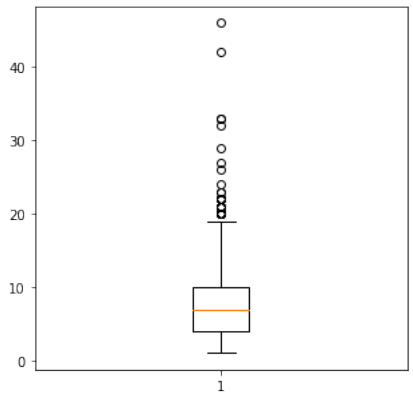

Figura 4. Diagrama de caixa do comprimento das palavras
Elaborado pelos Autores (2019).

Como visto anteriormente os valores de máximo e mínimo estão situados entre 0 e 10, e alguns *outliers* são detectados quando o tamanho é maior ou igual a 20.

Posteriormente foram comparadas as frequências de apresentação das palavras nas sentenças PRU e Não-PRU, mostradas na Figura 5, pode ser observado que são praticamente as mesmas para ambas as classes. Ou seja, palavras com alta aparição em sentenças PRU também possuem alta aparição em sentenças Não-PRU e vice-versa. Nosso objetivo é buscar métricas e análises que separe melhor estas palavras.

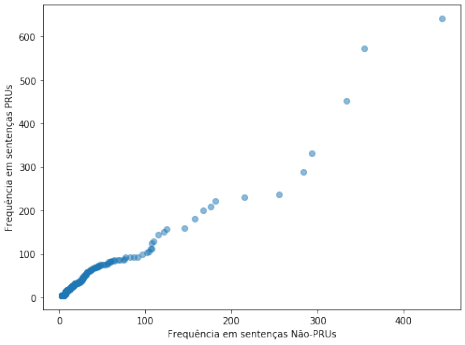

Figura 5. Aparição em sentenças PRU vs Aparição em sentenças Não-PRU
Elaborado pelos Autores (2019).

Intuitivamente, sabe-se que, se uma palavra aparece com mais frequência em uma classe em comparação a outra, isso pode ser uma boa medida do quanto a palavra é significativa para caracterizar a classe. Tendo por base isto, foi calculado, seguindo a fórmula abaixo, a **Taxa de classe** (TC) para cada palavra.

$$TC = \frac{aparicaoPalavra_{\mathrm{i}}}{\sum_C aparicaoPalavra_{\mathrm{i}}}, \qquad onde \quad C \in \{PRU, Nao - PRU\}.$$

Palavras com alto TC em uma determinada classe possuem aproximadamente aparição 0 em outras. Juntamente com o TC pode-se utilizar uma outra técnica que considere a frequência de aparição de uma palavra em sentenças de uma determinada classe. Será chamada esta técnica de **Frequência de Classe** (FC) e calcular-se-á da seguinte forma:

$$FC = \frac{classFreq_{\mathrm{i}}}{\sum_i classFreq_{\mathrm{i}}}$$

Pode-se combinar a TC e a FC, entretanto se calcular apenas a média desses valores, a TC será muito dominante e não refletirá as duas métricas de maneira eficaz. Para resolver este problema será calculada a **Média Harmônica** desse valores. Ela tende a mitigar o impacto de grandes *outliers* e agravar o impacto dos pequenos. Sua fórmula pode ser vista abaixo:

$$H = \frac{n}{\sum_{i=1}^{n} \frac{1}{x_{\mathrm{i}}}}$$

Ao calcular a média harmônica, o impacto do valor pequeno (neste caso, a frequência da classe) é muito agravado e acabou dominando o valor médio. Como forma de encontrar padrões entre as palavras e distribuição nas classes, pode-se aplicar sobre os valores da média harmônica a **função de distribuição cumulativa** (FDC). A FDC é função de distribuição de X, avaliada em x, é a probabilidade de que X terá um valor menor ou igual a x. Calculando o valor de FDC, pode-se ver onde o valor está na distribuição em termos de forma cumulativa.

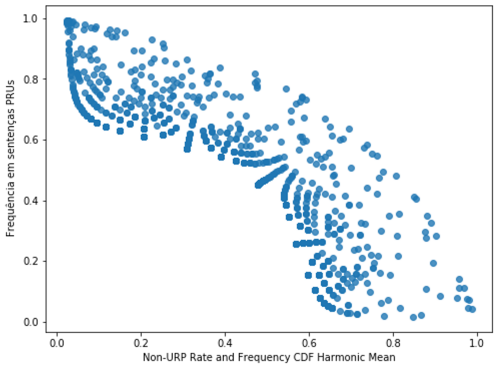

Figura 6. PRU FDC com Media Harmônica vs Não-PRU FDC com Media Harmônica

Elaborado pelos Autores (2019).

A Figura 6 mostra a distribuição dos dados quando aplicada a FDC sobre os valores da média harmônica, pode ser visto um interessante padrão foi observado com esta operação. Se um ponto de dados estiver próximo ao canto superior esquerdo, esta palavra será mais da classe PRU e, se estiver mais perto do canto inferior direito, será mais da classe Não-PRU.

6. Experimentos e Análise do Modelo

Para realizar os experimentos apresentados nesta seção, foi utilizada a base de dados, sendo 75% para treino (2899 dados) e 25% (966 dados) para teste e validação,

apresentada na seção anterior, bem como a linguagem de programação *Python* e algumas bibliotecas ofertadas por ela, como *Pandas* [McKinney et al. 2010], *Numpy* [van der Walt et al. 2011], *Sklearn* [Pedregosa et al. 2011], *Tensorflow* [Abadi et al. 2015] e *Keras* [Chollet et al. 2015].

Primeiro foi realizado uma transformação dos comentários textuais para números inteiros, esse procedimento é chamado de *Embedding*. Os ajustes dos parâmetros foram feitos utilizando busca gulosa, onde são dados aos parâmetros de maior relevância do algoritmo um conjunto de valores e é retornado o conjunto de valores de parâmetros com melhor precisão. Os parâmetros obtidos para o C-BLSTM são mostrados na Tabela 1.

Tabela 1. Parâmetros utilizados para o C-BLSTM

Hyper-Parameters	
Número de filtros de convolução	350
Tamanho do kernel de convolução	35
Dimensão de Memória do BLSTM	350
Taxa de Dropout	0.15

Após diversos testes e comparações de Acurácia, Precisão, *Recall* e F1-Score, o modelo final possui uma Camada Convolutiva, uma camada Bidirecional de LSTM, com implementação suportada *CuDNN*, uma Biblioteca de Aprendizagem Profunda NVIDIA CUDA, e duas Camadas Densas finais.

Foi realizada a comparação direta do C-BLSTM com diversos outros de aprendizado de máquina e aprendizado profundo. Os algoritmos que utilizados para comparação foram: Máquinas Vetores de Suporte (SVM), Árvores de Decisão, Perceptron de Múltiplas Camadas (MLP), Naive Bayes Gaussiano, Rede Neural Convolutiva (CNN), Rede Neural Recorrente (LSTM), Rede Neural Recorrente Bidirecional (BLSTM). Os resultados de Acurácia, Precisão, *Recall* e *F1-Score* para cada algoritmo são mostrados na Tabela 2.

Foram realizados testes para comprovar a eficiência do nosso modelo. Para avaliar o desempenho do modelo num conjunto de dados balanceado, no nosso caso, é utilizada a curva **Receiver Operating Characteristic**, ou **Curva ROC**. Ela é um gráfico da taxa de falsos positivos (eixo x) versus a taxa positiva verdadeira (eixo y) para um número de

Tabela 2. Comparação de Resultados

	Acurácia	***Precisão***	***Recall***	***F1-Score***	***AUC***
SVM	0.65	0.71	0.66	0.64	0.61
Árvore de Decisão	0.73	0.74	0.74	0.74	0.75
Naive Bayes Gaussiano	0.54	0.58	0.54	0.43	0.49
Perceptron de Múltiplas Camadas	0.65	0.66	0.66	0.65	0.67
CNN	0.85	0.86	0.85	0.86	0.89
LSTM	0.82	0.83	0.83	0.83	0.79
BLSTM	0.84	0.85	0.84	0.85	0.82
C-BLSTM	**0.88**	**0.87**	**0.87**	**0.87**	**0.94**

diferentes valores de limite candidatos entre 0,0 e 1,0. Em outras palavras, traça a taxa de alarme falso versus a taxa de acerto. A *Area Under the Curve* (AUC) pode ser usada como um resumo da habilidade do modelo. A figura 7 mostra nossa curva ROC para esta tarefa de classificação e a AUC para cada modelo pode ser encontrada na Tablhe 2.

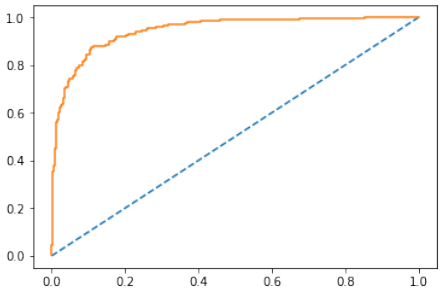

Figura 7. Curva ROC do C-BLSTM
Elaborado pelos Autores (2019).

Para esta tarefa, foi atingida uma AUC igual a ***0.94519***. Também foi encontrado o **limiar ótimo**, que é o limite que melhor discrimina entre as duas classes diferentes, pois maximiza a especificidade e a sensibilidade, seu valor é ***0.25502***.

Os métodos anteriores avaliaram o desempenho do modelo em toda a população, a próxima técnica, o **Gráfico de Ganhos Cumulativos** (GGC), avalia o desempenho do classificador em cada parte dos dados. O GGC mostra a porcentagem do número total de casos em uma determinada categoria ”ganhos”, visando uma porcentagem do número total de casos. Quanto mais uma curva estiver acima da linha de base, maior será o ganho. Nosso GGC é mostrado na figura 8, como pode-se observar as duas curvas estão situadas acima da curva da linha de base, garantindo assim um alto desempenho do nosso modelo de classificação.

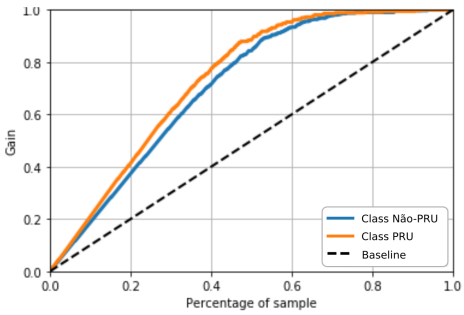

Figura 8. Gráfico de Ganhos Cumulativos do C-BLSTM
Elaborado pelos Autores (2019).

No último teste foram coletadas e classificadas, utilizando o modelo, 100 sentenças do *Twitter*. Foram convocados 3 pesquisadores de IHC para analisar as predições onde eles chegaram a conclusão que o modelo classificou corretamente 98 das 100 sentenças.

Esses testes serviram para mostrar o alto desempenho do modelo C-BLSTM em identificar sentenças PRU em postagens do Twitter. Como foi verificado, o modelo possui alto grau de confiabilidade e precisão na identificação de tais sentenças.

Através da comparação dos nossos resultados com os obtidos em [Mendes 2015, Mendes and Furtado 2017, Lima et al. 2017], este é o melhor resultado para identificação de postagens relacionados ao uso em sistemas sociais utilizando modelos de aprendizagem automática. A comparação dos nossos resultados com os modelos exclusivos CNN e LSTM mostra que o LSTM aprende melhor as dependências de longo prazo nas sequências de representações de nível superior.

7. Conclusão

Foram buscadas técnicas para automatizar a identificação de PRUs. Uma das melhores técnicas, como mostrado, foi usar o aprendizado profundo combinando uma CNN com uma BLSTM e usando o modelo que chamamos de C-BLSTM para identificação automática de PRUs. O modelo foi validado com uma base de postagens em português extraídos da rede social Twitter, onde obteve excelentes resultados.

Pode-se explorar em trabalhos futuros uma otimização para este modelo explorando as várias arquiteturas do aprendizado profundo descritas na literatura, bem como verificando sua adoção para classificar uma sentença PRU nas diversas facetas de Usabilidade e Experiência do Usuário, onde acreditamos que o C-BLSTM pode trazer excelentes resultados.

Referências

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Chollet, F. et al. (2015). Keras. `https://keras.io`.

Chunting, Z., Chonglin, S., Zhiyuan, L., and Francis, C. L. (2015). A c-lstm neural network for text classification. *arXiv preprint arXiv:1511.08630*.

Freitas, L., Silva, T., and Mendes, M. (2016). Evaluation of spotify: an evaluation textual experience using the maltu methodology. pages 1–4.

Hedegaard, S. and Simonsen, J. G. (2013). Extracting usability and user experience information from online user reviews. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 2089–2098, New York, NY, USA. ACM.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Lima, A. M., Silva, P. B., Cruz, L. A., and Mendes, M. S. (2017). Investigating the polarity of user postings in a social system. In *International Conference on Social Computing and Social Media*, pages 246–257. Springer.

Loper, E. and Bird, S. (2002). Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Lan-*

guage Processing and Computational Linguistics - Volume 1, ETMTNLP '02, pages 63–70, Stroudsburg, PA, USA. Association for Computational Linguistics.

McKinney, W. et al. (2010). Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX.

Mendes, M. S. (2015). *MALTU – um modelo para avaliação da interação em sistemas sociais a partir da linguagem textual do usuário*. PhD thesis, Federal University of Ceará.

Mendes, M. S. and Furtado, E. S. (2017). Uux-posts: a tool for extracting and classifying postings related to the use of a system. In *Proceedings of the 8th Latin American Conference on Human-Computer Interaction*, page 2. ACM.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, 12:2825–2830.

Pereira, R., Baranauskas, M. C. C., and da Silva, S. R. P. (2010). Softwares sociais: uma visão orientada a valores. In *Proceedings of the IX Symposium on Human Factors in Computing Systems*, pages 149–158. Brazilian Computer Society.

Preece, J., Rogers, Y., and Sharp, H. (2015). *Interaction design: beyond human-computer interaction*. John Wiley & Sons.

Schuster, M. and Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Shaw, D. (1996). Handbook of usability testing: How to plan, design, and conduct effective tests. *Journal of the American Society for Information Science*, 47(3):258–259.

van der Walt, S., Colbert, S. C., and Varoquaux, G. (2011). The numpy array: a structure for efficient numerical computation. *CoRR*, abs/1102.1523.