Detecção de Fake News com Técnicas de Aprendizado de Máquina

Lucas Alves da Cunha Parreiras

Universidade Federal de Minas Gerais – UFMG
Belo Horizonte
MG - Brasil
lucasparreiras@ufmg.com

RESUMO

A democratização da informação através da internet e das novas mídias trouxeram consigo uma ameaça à democracia política com as *Fake News*. Observando este cenário este trabalho propõe modelos através do Aprendizado de Máquina para a classificação de notícias em falsas ou reais. São explorados 17 modelos através de 4 algoritmos de aprendizado de máquina: LSTM, CNN, Adaboost e Naive Bayes.

Palavras-chave

Fake News; Democracia; Aprendizado de Máquina; LSTM, CNN, Adaboost, Naive Bayes.

1. INTRODUÇÃO

A internet e o surgimento mais recente de uma nova mídia possibilitou uma nova relação entre o emissor-receptor, inserindo-se o conceito de interação. Para além disso, as novas mídias permitiram que todos participem do processo de produção de informação. Assim, a sociedade presencia a democratização da informação [3].

Em meio à democratização da informação a sociedade está presenciando uma grande ameaça ao sistema político democrático com as chamadas *Fake News*, ou notícias falsas. Normalmente as *Fake News* são geradas para interesses comerciais para atrair espectadores e adquirir maior receita de publicidade. Porém, pessoas e grupos com agendas politicamente maliciosas são conhecidos por produzir notícias falsas para influenciar eventos e políticas em todo o mundo. Existe a crença e estudos que indicam que a circulação de notícias falsas teve impacto material no resultado das eleições presidenciais dos EUA em 2016 [1]. No Brasil a sociedade presencia a ameaça das *Fake News* e há uma grande preocupação devido ao processo eleitoral que ocorrerá em 2018.

Do ponto de vista da NLP (*Neuro-linguistic programming*), esse fenômeno oferece uma oportunidade interessante e valiosa para identificar padrões que podem ser codificados em um modelo classificador. Este trabalho propõe explorar métodos de Aprendizado de Máquina para criar modelos que possibilitem a classificação de notícias falsas. A organização deste documento está feita da seguinte forma: seção 2 descreve trabalhos relacionados, a seção 3 explica as metodologias utilizadas, a seção 4 detalha os experimentos e resultados, por fim, o trabalho é concluído na seção 5 com conclusões, limitações e trabalhos futuros.

2. TRABALHOS RELACIONADOS

A classificação de texto tem um rico histórico de pesquisa dentro da comunidade de NLP e um conjunto igualmente impressionante de aplicações práticas demonstram sua importância.

Embora existam agências de *fact check*, ferramentas e produtos para detectar fontes de notícias falsas, esse problema é abordado neste trabalho como uma instância da classificação de texto, usando o conteúdo da notícia como fonte de recursos. Isso permite concentrar nos algoritmos relacionados à NLP, o que permite explorar em profundidade o desempenho de uma variedade de modelos em uma tarefa específica.

3. METODOLOGIA

Antes de iniciar o processo de modelagem preditiva, os dados de texto requerem uma preparação especial. Os tratamentos utilizados na implementação dos modelos deste trabalho estão descritos em detalhe na seção 3.1.

Foram implementados modelos para a classificação das notícias utilizando: Naive Bayes, Adaboost, LSTM e CNN+LSTM. As seções 3.2 a 3.5 descrevem cada um dos modelos.

Para treinamento e validação foi utilizado o conjunto de dados *"Fake News: Build a system to identify unreliable news articles"* disponível na plataforma Kagle [11].

3.1. Preparação dos Dados

O primeiro passo na preparação dos dados é a limpeza do texto. No processo de limpeza do texto para os modelos de predição, são removidas pontuação e *stop words*, todos caracteres maiúsculos foram transformados para minúsculo e as palavras flexionadas foram reduzidas ao seu tronco (*stemming*).

Após o processo de limpeza do texto, para utilizar o texto nos algoritmos de aprendizado de máquina é realizada a tokenização, ou seja, dada uma sequência de caracteres e uma unidade de documento definida, a tokenização é a tarefa de cortá-la em pedaços, chamadas de tokens. Um token é uma instância de uma sequência de caracteres em algum documento específico que é agrupado como uma unidade semântica útil para processamento [10].

3.2. Naive Bayes

Naive Bayes é um algoritmo de classificação para problemas de classificação binária (duas classes) e multi-classe baseado no teorema de Bayes.

No Naive Bayes o cálculo das probabilidades para cada hipótese é simplificado para tornar seu cálculo tratável. Ao invés de tentar calcular os valores de cada valor de atributo P (d1, d2, d3 | h), eles são assumidos como condicionalmente independentes, dado o valor alvo e calculados como P (d1 | h) * P (d2 | H).

3.3. Adaboost

Adaboost consiste de um conjunto de modelos mais simples (conhecidos como "modelos fracos") que, embora não sejam muito eficazes individualmente, são combinados para um desempenho ótimo.

O processo pelo qual esses modelos fracos são combinados é, no entanto, mais complexo do que simplesmente a média dos resultados. Muito brevemente, o processo de treinamento do Adaboost pode ser descrito da seguinte forma:

Para cada modelo fraco:

1) Treine o modelo fraco para que a soma de erros quadráticos seja minimizada

2) Atualize os pesos, de modo que os casos classificados corretamente tenham seu peso reduzido e os casos de classificação incorreta tenham seus pesos aumentados.

3) Determine o peso do modelo fraco, ou seja, a contribuição total do resultado do modelo fraco para a pontuação geral. Isso é conhecido como alfa e é calculado como 0,5 * ln ((1-error.rate) /error.rate))

À medida que o peso é atualizado em cada iteração, cada modelo fraco tenderá a se concentrar mais nos casos que foram classificados incorretamente em instâncias anteriores.

3.4. Long Short-Term Memories (LSTM)

A LSTM é um tipo de rede neural recorrente. Foi inicialmente proposta por Hochreiter e Schmidhuber [4], e desde então várias modificações na unidade original foram feitas. Ao contrário da unidade recorrente, que simplesmente calcula uma soma ponderada do sinal de entrada e aplica uma função não linear, cada unidade LSTM mantém uma memória ct no tempo t, que é usada subsequentemente para determinar a saída, ou a ativação, ht, da célula .

$$h_t = u_t \odot \tilde{h}_t + (1 - u_t) \odot h_{t-1}$$
$$\tilde{h}_t = \tanh(x_t W + (r_t \odot h_{t-1})U + b)$$
$$u_t = \sigma(x_t W_u + h_{t-1} U_u + b_u)$$
$$r_t = \sigma(x_t W_r + h_{t-1} U_r + b_r)$$

As equações acima podem ser vistas nos quatro estágios operacionais fundamentais da GRU.

Na figura 1 tem-se a arquitetura da LSTM utilizada neste trabalho.

3.5. Convolutional Neural Network (CNN) with LSTM

O principal módulo de uma CNN é calcular a convolução entre entrada e saída. Assim como a CNN é utilizada na visão computacional, para processamento de texto uma matriz é necessária como a entrada da CNN. Redes neurais convolucionais se destacam em aprender a estrutura espacial em dados de entrada.

Os dados das notícias têm uma estrutura espacial unidimensional na sequência de palavras e a CNN pode ser capaz de identificar características invariantes para classificação se uma notícia é ou não falsa. Estas características espaciais aprendidas podem então ser aprendidas como sequências por uma camada LSTM.

Uma camada unidimensional CNN e max pooling após a camada Embedding é adicionada, que depois alimenta os recursos consolidados à LSTM cuja saída é enviada para uma não-linearidade sigmóide para prever o rótulo.

Na figura 2 tem-se a arquitetura da rede CNN+LSTM utilizada neste trabalho.

4. EXPERIMENTOS E RESULTADOS

4.1. Implementação

Todo o código foi escrito em Python 3.5 usando o TensorFlow[9], NumPy[7], scikit-learn[8], Keras[5] e Natural Language Toolkit[6]. O dropout foi empregado como um mecanismo de regularização para as redes neurais LSTM e CNN+LSTM.

As experiências foram executadas em máquinas locais somente utilizando CPU, bem como com Google Colab habilitado para GPU fornecidos pela Google.

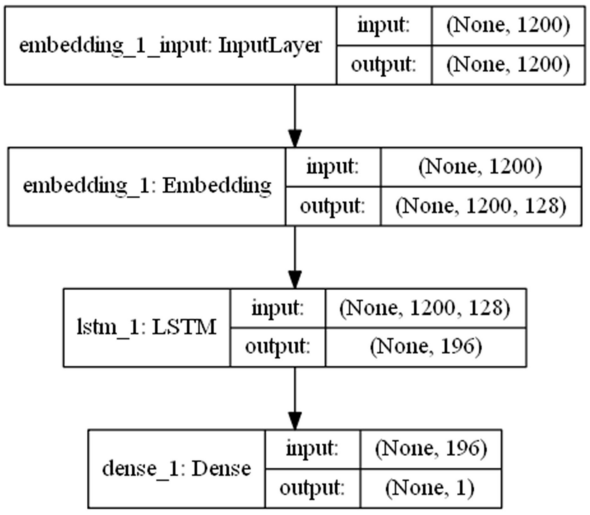

Figura 1: Arquitetura LSTM

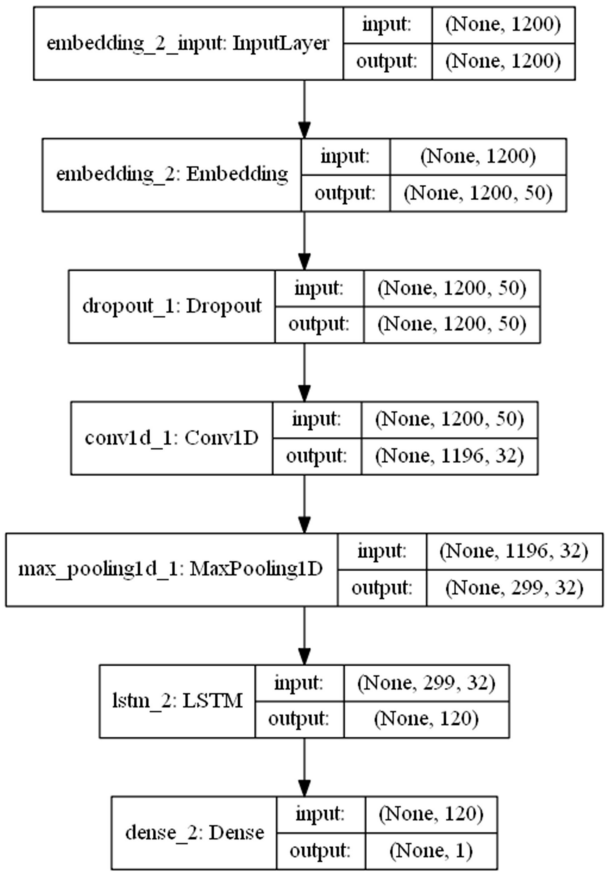

Figura 2: Arquitetura CNN+LSTM

4.2. Experimentos

Ao todo foram projetados 17 modelos para classificar as *Fake News*. Para Naive Bayes e Adaboost, primeiro foi utilizado o método TF-IDF, com unigram, bigram e trigram e utilizado todas as *features*, que somaram 10.429.999. Depois foi utilizado a contagem de palavras com o método CountVectorizer, com *unigram*, *bigram* e *trigram* utilizando 20k *features*, 18k *features*, 15k *features*, 10k *features* e 8k features. Para as redes neurais LSTM e CNN+LSTM foi utilizado como input a matriz construída pelo *word embeddings* utilizando para cada exemplo de treino 1200, 800, 400, 297, 200 termos utilizando como *padding* os próprios termos da matriz mantendo-se a sequência da primeira posição para a última, depois foi utilizado como input a mesma quantidade de termos porém completando a matriz com zero.

Para validação dos modelos foi utilizado o método de validação cruzada, para NB e Adaboost utilizando 10 grupos e para as redes neurais 5.

4.3. Comparação e Análises

A performance dos modelos foi medido pela acurácia e podem ser observadas nas tabelas 2 e 3.

Os modelos demonstram que a decisão da quantidade de *features* é um fator de extrema importância quando está se trabalhando com classificação de textos. Para os modelos utilizando Naive Bayes e Adaboost, pode-se observar que Adaboost tem um comportamento superior à medida que são consideradas maiores quantidades de *features*, já para o Naive Bayes, foi observado uma melhor performance para os valores intermediários.

Era esperado que os modelos desenvolvidos com Naive Bayes tivesse performance inferior aos demais métodos, porém NB é um método extremamente rápido e com uma boa performance, o que possibilita ter um modelo base para ser superado. Porém, dependendo da quantidade de *features* escolhidas, a performance do modelo com NB se aproxima ao modelo com Adaboost, sendo que o segundo espera-se sempre uma performance superior ao primeiro, reforçando a importância da condição de escolha da quantidade de *features*.

É importante observar que NB e Adaboost utilizam-se da frequência de contagem de palavras diferentemente das LSTMs, onde a sequência de termos é importante para o aprendizado da rede.

Nas redes neurais podemos observar que para as duas arquiteturas, no geral, há um melhor aprendizado conforme pode-se observar a curva de perda dos modelos nas figuras de figura 4 à figura 22, quando é utilizando o complemento da matriz com palavras do próprio texto, consequentemente observa-se uma melhor acurácia. Observa-se também uma melhor performance da arquitetura que utiliza CNN e LSTM. Outro aspecto interessante observado, é que em modo geral a performance da CNN+LSTM, especialmente utilizando-se zero como complemento da matriz, aumenta quando diminui-se a quantidade de palavras. Analogamente ao estudo realizado por Barbisan e Machado [2] sobre o tópico no texto argumentativo, onde demonstram que 73,8% dos novos tópicos dos textos analisados no estudo, estão ancorados em tópicos anteriores. Dessa forma, poderíamos entender que os textos que foram truncados para serem treinados pelas redes neurais, não perderam sua essência e ao contrário disto, possibilitaram que a rede tivesse um melhor aprendizado. Desta forma, poder-se-ia explicar a arquitetura da rede CNN+LSTM como sendo a CNN um extrator de tópicos e a LSTM o classificador que considera o aspecto temporal dos tópicos.

Tabela 2: Acurácia dos modelos NB e Adaboost

	NB ACC (%)	Adaboost ACC (%)
TF-IDF	84,92	**93,74**
20k features	**91,72**	93,09
18k features	91,63	93,09
15k features	91,26	93,09
10k features	90,13	93,09
8k features	89,74	93.09

Tabela 2: Acurácia dos modelos NB e Adaboost

	LSTM ACC (%)	CNN+LSTM ACC (%)
1200 termos com word-padding	**91,43**	94,08
1200 termos com zero-padding	50,28	78.15
800 termos com word-padding	89,16	90,76
800 termos com zero-padding	66,51	76,26
400 termos com word-padding	89,13	93,98
400 termos com zero-padding	89,30	90,21
297 termos com word-padding	90,39	93,26
297 termos com zero-padding	89,50	96,08
200 termos com word-padding	89,04	**96,62**
200 termos com zero-padding	89,88	95,25

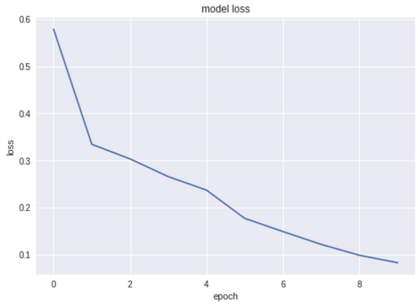

Figura 3: Perda para LSTM - 1200 termos com word-padding

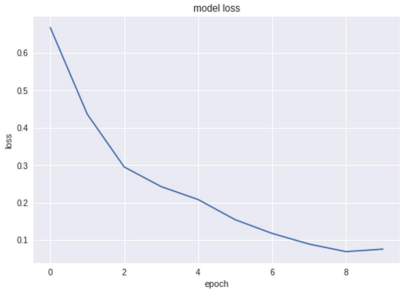

Figura 4: Perda para CNN+LSTM - 1200 termos com word-padding

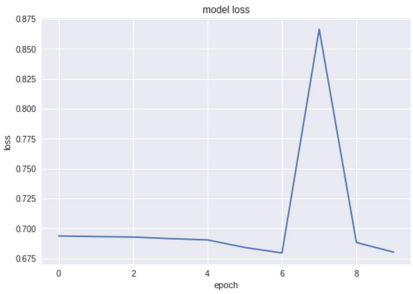

Figura 5: Perda para LSTM - 1200 termos com zero-padding

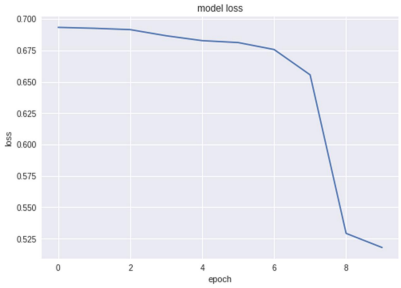

Figura 6: Perda para CNN+LSTM - 1200 termos com zero-padding

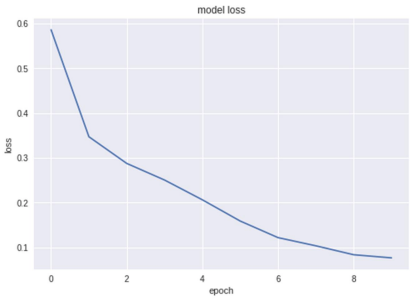

Figura 7: Perda para LSTM - 800 termos com word-padding

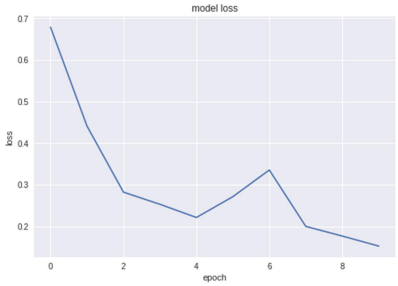

Figura 8: Perda para CNN+LSTM - 800 termos com word-padding

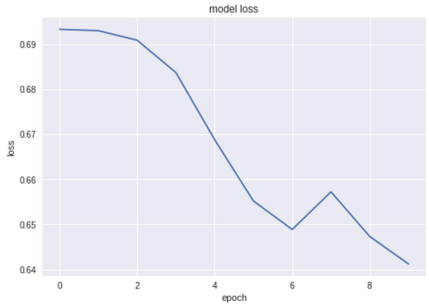

Figura 9: Perda para LSTM - 800 termos com zero-padding

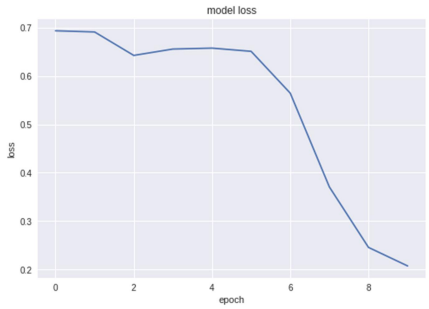

Figura 10: Perda para CNN+LSTM - 800 termos com zero-padding

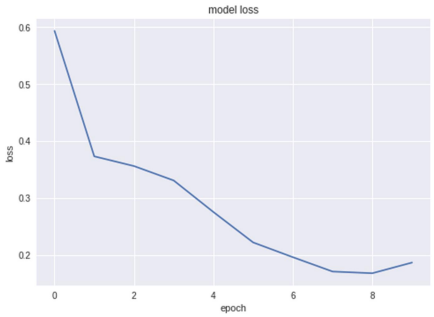

Figura 11: Perda para LSTM - 400 termos com word-padding

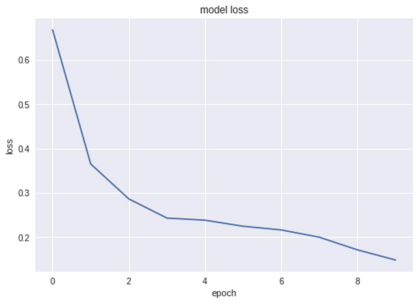

Figura 12: Perda para CNN+LSTM - 400 termos com word-padding

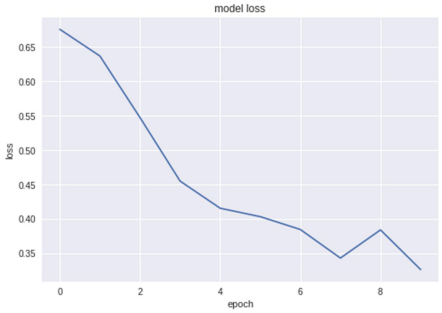

Figura 13: Perda para LSTM - 400 termos com zero-padding

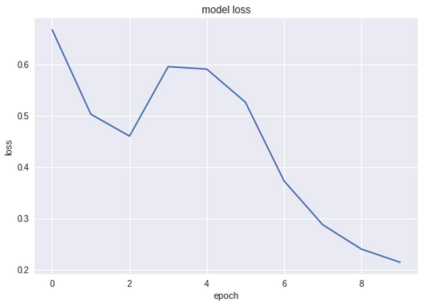

Figura 14: Perda para CNN+LSTM - 400 termos com zero-padding

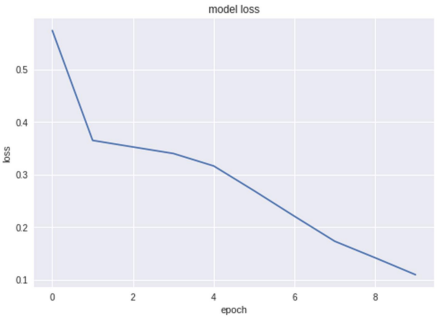

Figura 15: Perda para LSTM - 297 termos com word-padding

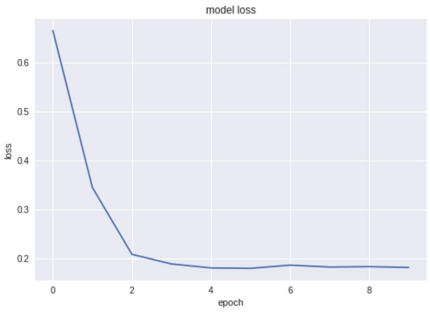

Figura 16: Perda para CNN+LSTM - 297 termos com word-padding

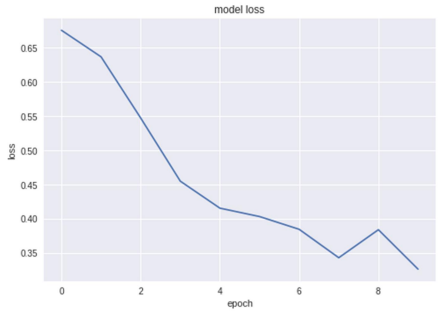

Figura 17: Perda para LSTM - 297 termos com zero-padding

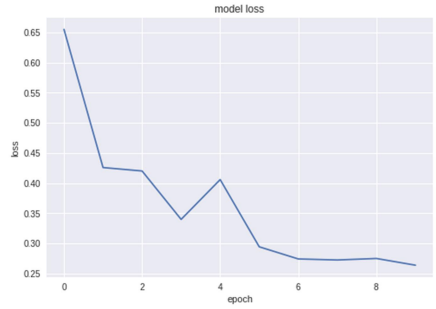

Figura 18: Perda para CNN+LSTM - 297 termos com zero-padding

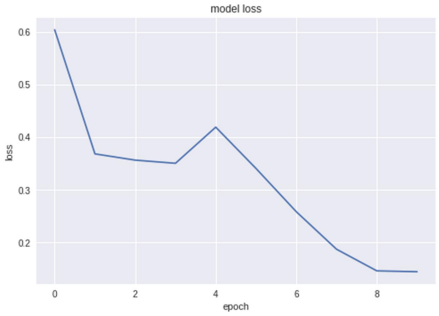

Figura 19: Perda para LSTM - 200 termos com word-padding

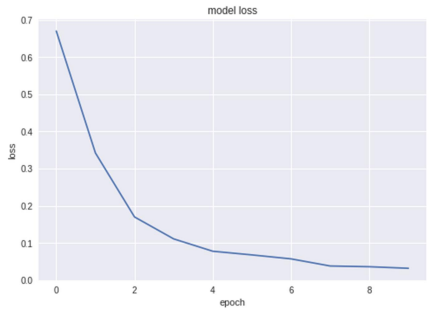

Figura 20: Perda para CNN+LSTM - 200 termos com word-padding

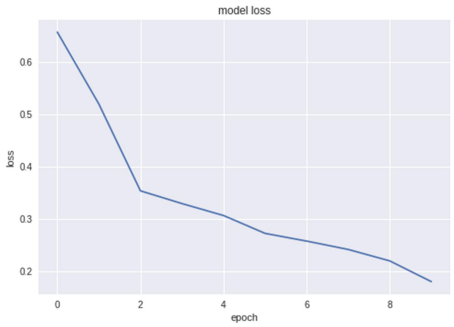

Figura 21: Perda para LSTM - 200 termos com zero-padding

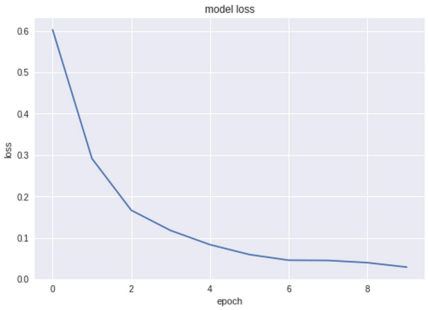

Figura 22: Perda para CNN+LSTM - 200 termos com zero-padding

5. CONCLUSÃO

Neste artigo, Detecção de Fake News com Técnicas de Aprendizado de Máquina é apresentado. O trabalho se propôs a criar modelos para classificação de notícias tendo como base para treinamento dos modelos um conjunto de dados disponível na plataforma Kaggle.

Os resultados experimentais demonstram que a escolha de quantidade de *features* é essencial para uma boa performance do modelo. Observou-se que a utilização de métodos de boosting são tão fortes quanto redes neurais em aprendizado profundo. Porém o método proposto com a combinação de métodos de aprendizado profundo, mais especificamente CNN e LSTM, ficam mais robustos para NLP.

Futuramente, mais trabalhos serão explorados para melhorar o desempenho do modelo CNN+LSTM, combinando-se mais CNNs e agregando à LSTM inspirando-se no estudo de tópicos de textos narrativos e argumentativos.

REFERÊNCIAS

[1] Allcott H., Gentzkow M.”Social Media and Fake News in the 2016 Election”. 2017, Journal of Economic Perspectives, vol: 31: 211-236.

[2] Barbisan, L., Machado, R.2000. “O Tópico no Texto Argumentativo”. Letras de Hoje, Porto Alegre, p. 69-106

[3] Escobar J.2005. “A Internet e a Democratização da Informação – proposta para um estudo de caso”.UERJ, Rio de Janeiro.

[4] Hochreiter, S., and Schmidhuber, J.1997. “Long Short-Term Memory”, Neural Computation.

[5] Keras: The Python Deep Learning library. Disponível em <https://keras.io/>
[6] Natural Language Toolkit. Disponível em <https://www.nltk.org/>
[7] Numpy. Disponível em <http://www.numpy.org/>
[8] Scikit-learn: Machine Learning in Python. Disponível em <http://scikit-learn.org/stable/index.html>
[9] TensorFlow. Disponível em <https://www.tensorflow.org/>
[10] ”Tokenization”.Disponível em <https://nlp.stanford.edu/IR-book/html/htmledition/tokenization-1.html >
[11] UTK Machine Learning Club. “Fake News: Build a system to identify unreliable news articles”. Disponível em <https://www.kaggle.com/c/fake-news