Análise de Sentimentos em Publicações do Twitter utilizando Redes Neurais Recorrentes

Adriano Soares - sadrianorod@gmail.com
Adrisson Samersla - adrissonsamersla@gmail.com
Felipe Mourad - felipemourad1999@gmail.com

Abstract

This paper aims to provide an efficient and effective solution for sentiment analysing messages in Twitter. The goal is to predict overall humor of tweets written with respect to the American Presidential Candidates: Donald Trump e Joe Biden. For this purpose, 4 incremental recurrent neural networks are proposed and implemented. Techniques such as Transfer Learning and Feature Extraction were used. The architecture that extracts emojis and slang's from text ended up having the best performance, achieving about 70% of accuracy.

Index Terms

Artificial Intelligence, Machine Learning, Neural Networks, Natural Language Processing, Twitter, Sentiment Analysis.

I. Introdução

Diversas pesquisas políticas e sociais no âmbito digital atraíram olhares a respeito dos limites da influência das redes sociais em determinar o rumo político nos países, isto é, em determinar até que ponto se consegue conduzir a opinião do público até o momento do voto. Escândalos de manipulação em redes sociais, como o da Cambridge Analytica, retratados no documentário *Privacidade Hackeada*, instigam se de fato o destino político e consequentemente social podem de fato serem efeitos de acontecimentos no ambito digital.

Diante disso, este trabalho se destina a desenvolver uma rede neural capaz de captar sentimentos utilizando textos curtos publicados na rede social Twitter. Nessa lógica, utilizou-se um conjunto de publicações separadas em três classes de informações: positivas, neutras ou negativas. Com esses dados, obtidos na plataforma *Kaggle*, foi possível treinar redes neurais utilizando a metodologia de aprendizado supervisionado. Os dados utilizados após o treinamento foram obtidos da API fornecida pelo próprio Twitter e filtrados para publicações que falam explicitamente sobre os candidatos à presidência dos Estados Unidos: Donald Trump e Joe Biden.

II. Fundamentos

Redes neurais são uma técnica de aprendisado supervisionado que emprega estruturas menores chamadas neurônios em redes para aproximar uma função desejada. Apesar da estrutura básica fixa, a diversidade de combinações possíveis entre esses elementos torna essa técnica flexível. Dentre essas arquiteturas, duas classes gerais podem ser distinguidas: *feedforward* e recorrente.

Esse primeiro tipo de rede é caracterizado pela seguinte propriedade: os resultados da computação de camadas anteriores e passada à frente, sem voltar. O formato mais simples são camadas consecutivas nos quais todos os neurônios são ligados a todos, por isso chamadas completamente conectadas (no *Keras*, *Dense*). Há também um arranjo conhecido por convolucional, no qual as ligações entre as camadas assemelham-se a máscaras convolucionais, conferindo conexões esparsas, uma forma eficiente de descrever interações complicadas entre muitas variáveis por meio de blocos menores (Goodfellow et al., 2016).

A segunda arquitetura mencionada é caracterizada por passar o resultado de uma rede como entrada para ela própria, uma espécie de realimentação, proporcionando propriedades de memória de curta duração (Russel e Norvig, 2009). Por causa dessa propriedade, mostram-se bastante eficientes no processamento de sequências (Goodfellow et al., 2016).

O campo de Processamento de Linguagem Natural (NLP - *Natural Language Processing*) aumeja construir computadores que sejam capazes de comunicar-se fluidamente com humanos, inclusive por vias textuais. Dessa forma, é necessário algoritmos capazes de aprender as idiossincrasia da comunicação humana. Na escrita, há dois desafios: a representação em caracteres, que é bastante ineficiente para um computador processar, e a correlação entre elementos muitas vezes distantes no texto. Para o primeiro obstáculo, a saída empregada é representar as palavras em espaços multidimensionais que, espera-se, represente o significado desse vocábulo. Tal processo é melhor escrito na seção III-B. Para o segundo desafio, frases podem ser entendidas como sequências de palavras, portanto o uso de redes recorrentes mostra-se deveras apropriado nesse contexto, sendo de fato bastante aplicadas no domínio de processamento de texto.

III. Desenvolvimento

Os itens a seguir discorrem a respeito de cada parte do desenvolvimento do projeto, desde a extração e coleta dos dados ao *deploy* da solução para predição dos ânimos na corrida presidencial americana de 2020. Foram implementadas 4 versões de redes neurais, por meios de incrementos sucessivos às arquiteturas, com o objetivo de aumentar a acurácia na predição de sentimentos de tweets, como será descrito nas próximas seções.

A. Análise do Conjunto de Dados

Os datasets usados para treinamento e validação contêm 27.481 e 3.534 entradas de dados respectivamente, já previamente separados, no formato *CSV*. Cada linha, por sua vez, contém os campos: identificador da mensagem (*textID*), texto (*text*), fragmento do texto destacado (*selected_text*) e sentimento atribuido (*sentiment*).

Antes que esses dados possam servir de input para às redes neurais a serem avaliadas, eles são pré-processados. Tal etapa compreende uma série de transfomações a seguir listadas:

- Conversão da coluna de sentimentos, que é literal, nos valores numéricos 0, 1 e 2 (negativo, neutro e positivo, respectivamente);
- Remoção de entradas nulas;
- Tokenização;
- Transformação do valor numérico relativo ao sentimento em uma variável categórica, ie., um vetor de três entradas (uma para cada classe) no formato *one-hot encoding*.

Os conjuntos de dados de treinamento e de validação, após o pré-processamento, possuem a seguinte distribuição de sentimentos:

Distribuição de Sentimentos			
Dataset	Negativo (28,3%)	Neutro (40,5%)	Positivo (31,2%)
Treinamento	7781	11117	8582
Validação	1001	1430	1103

TABLE I: Distribuição dos tweets entre as classes de sentimento. Repare que a proporção é a mesma para os datasets de treinamento e de validação.

1) Tokenização: Essa etapa do pré-processamento é bastante importante para o desempenho da rede neural e para sua arquitetura, logo merece destaque. Para que as frases sejam processadas por uma rede neural, elas precisam ser convertidas em arranjos (possivelmente multidimensionais) de valores numéricos. Para isso, o algoritmo de tokenização confere um token (número inteiro único) a cada nova palavra encontrada no texto, usando heurísticas próprias. Assim, cada sentença se transforma em uma lista de tokens. A ferramenta utilizada para isso foi o *Tokenizer*, contido no pacote de pré-processamento de texto do *Tensorflow*.

Há, contudo, variações nesse fluxo de trabalho. Por exemplo, pode-se tokenizar os caracteres, extraindo-se, assim, informações de sequências de caracteres que não constituem palavras propriamente ditas, como *hashtags* e *emojis*, extremamente importantes nesse formato de comunicação (Santos e Gatti, 2014).

B. Arquiteturas

Como o input da rede neural é uma lista de tokens, tais informações precisam ser representados em espaços multidimensionais, um formato de dados mais adequado às tecnicas de aprendizado, que usam variações da Descida de Gradiente. Para isso, as arquiteturas tradicionalmente possuem uma primeira camada de Embedding, técnica especializada em expandir a representação vetorial de um determinado conjunto de dados. Por exemplo: suponha que a palavra "*cat*" seja mapeada para o token "5". A camada de embedding da rede neural obterá um vetor multidimensional a partir desse token, de forma que palavras relacionadas possuem uma pequena distância vetorial. Assim, os tokens das palavras "*cat*" e "*dog*" gerarão vetores razoavelmente próximos. De forma muito interessante, tem-se que o vetor obtido pelo token de "*cat*", subtraído do vetor obtido pelo token de "*dog*", adicionado pelo vetor obtido pelo token de "*bark*", resulta num vetor muito próximo de "*meow*". Assim, é possível relacionar as palavras nesse espaço vetorial de forma intuitiva [3].

Após essa primeira camada, há inúmeras arquiteturas possíveis. As propostas neste trabalho possuem uma camada recorrente com comunicação bidirecional, por meio dos *layers* Bidirectional e LSTM do Keras. A seguir, tem-se uma camada completamente conectada de 1024 neurônios, finalizando com outra camada de 3 neurônios (para as 3 classes de sentimento). Todas foram treinadas usando Descida de Gradiente Estocástica sobre todo o dataset de treinamento por 100 épocas. A função de custo escolhida foi a entropia cruzada categórica.

Tirando essa arquitetura comum, há 4 versões incrementais implementadas para comparação:

1) Embedding: versão base acima descrita;
2) GloVe: mesma arquitetura que a versão Embedding, porém carrega (e congela) os pesos do Embedding, obtidos com o conjunto pré-treinado da ferramenta GloVe, usando a técnica conhecida por Transfer Learning.

3) Extraction: acrescenta à arquitetura da versão GloVe um vetor extra de features previamente extraídos do *dataset*, concatenando com a saída da camada recorrente, usando a técnica conhecida por Feature Extraction.
4) CRNN: versão não-incremental das anteriores. Combinação das arquiteturas convolucional e recorrente.

A principal diferença da camada de Embedding puro e da GloVe [4], é que na GloVe leva-se em conta o contexto da palavra, tentando usufruir-se das palavras em volta na determinação de seu vetor, o que não acontece no Embedding. Assim, a camada de Embedding treinada possui menos potencial de aprender o contexto, mas mais flexibilidade, o que pode trazer frutos positivos em termos de aprendizado de sentimento, enquanto que a GloVe apresenta-se estática, com forte conhecimento de contexto.

As figuras 1, 2, 3 contêm os resumos das arquiteturas usadas. Nas três seções a seguir, serão discutidos os incrementos nas arquiteturas que compõem essas diferentes versões.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 32, 200)           5320000
_________________________________________________________________
bidirectional (Bidirectional (None, 128)               135680
_________________________________________________________________
dense (Dense)                (None, 1024)              132096
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 3075
=================================================================
Total params: 5,590,851
Trainable params: 5,590,851
Non-trainable params: 0
_________________________________________________________________
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 32, 200)           5320000
_________________________________________________________________
bidirectional (Bidirectional (None, 128)               135680
_________________________________________________________________
dense (Dense)                (None, 1024)              132096
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 3075
=================================================================
Total params: 5,590,851
Trainable params: 270,851
Non-trainable params: 5,320,000
_________________________________________________________________
```

Fig. 1: Sumário das duas primeiras redes neurais implementadas. À esquerda, a versão Embedding. À direita, a versão GloVe. Repare a diferença de parâmetros treináveis, pois os pesos da camada de Embedding na segunda versão foram pré-treinados pelo GloVe, logo podem ser congelados.

```
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
main_input (InputLayer)         [(None, 32)]         0
__________________________________________________________________________________________________
embedding (Embedding)           (None, 32, 200)      5320000     main_input[0][0]
__________________________________________________________________________________________________
bidirectional (Bidirectional)   (None, 128)          135680      embedding[0][0]
__________________________________________________________________________________________________
emoji_feature_input (InputLayer [(None, 7)]          0
__________________________________________________________________________________________________
concatenate (Concatenate)       (None, 135)          0           bidirectional[0][0]
                                                                 emoji_feature_input[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 1024)         139264      concatenate[0][0]
__________________________________________________________________________________________________
dense_2 (Dense)                 (None, 3)            3075        dense_1[0][0]
==================================================================================================
Total params: 5,598,019
Trainable params: 278,019
Non-trainable params: 5,320,000
__________________________________________________________________________________________________
```

Fig. 2: Sumário da versão Extraction. Repare que há 2 *input*: *main_input* (para os tokens do texto) e *emoji_feature_extraction* (para as informações extraídas do texto).

C. Transfer Learning

Como pode ser visto na figura 1, a camada de Embedding contém muito mais parâmetros do que as outras camadas. Além disso, o dataset utilizado não é suficientemente grande para treinar adequadamente esse *layer*. Como essa estrutura é comum a todas as arquiteturas que processam textos, há ferramentas que já fornecem tais pesos previamente treinados. Dentre esses recursos, encontra-se GloVe: um algoritmo de aprendizado não-supervisionado com base em estatísticas de co-ocorrência entre palavras (Pennington et al., 2014). No site desse projeto estão disponíveis conjuntos pré-treinados de vetores que podem ser usados como pesos para a camada Embedding das redes neurais implementadas. A opção escolhida foi a treinada em uma massa de dados do Twitter, o que deve ajudar na transferência de conhecimento. As versões 2, 3 e 4 utilizam essa técninca.

D. Feature Extraction

Para análise de *features* em publicações do Twitter, utilizou-se como métrica gírias encontradas recorrentemente. A lógica utilizada foi criar uma rede neural paralela responsável por receber este *input* e concatená-lo com a rede principal a fim de que estas gírias se tornem métricas extras para auxiliar a rede na compreensão se determinada publicação é de fato algo positivo, negativo ou neutro.

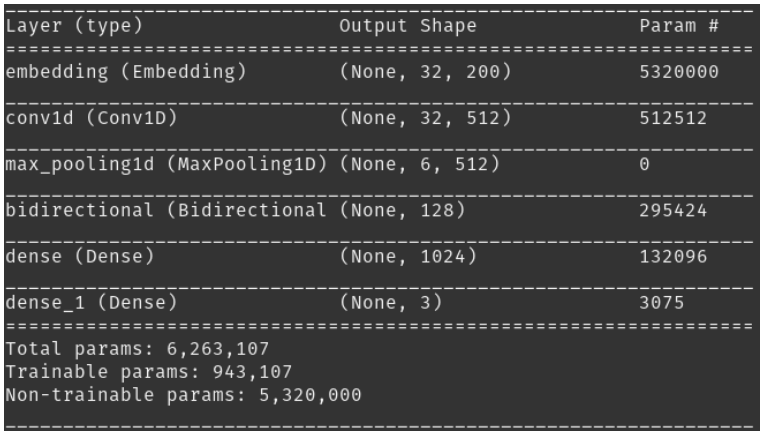

Fig. 3: Sumário da última versão implementada: CRNN. Nessa arquitetura, há uma camada de convolução e de MaxPolling (típico de redes convolucionais), seguidas por uma camada recorrente com Bidirectional e LSMT (característico de redes recorrentes).

A rede foi estruturada em um *input*, paralelo com o principal, onde o tamanho da entrada depende do número de gírias mapeadas, cada coluna recebe 0 caso não possua a gíria correspondente no texto analisado ou 1 caso o contenha. Este processamento de dados é realizado previamente, e todo o código relativo a tarefa encontra-se no arquivo *preprocess.py*.

Após os input, a rede responsável pela análise de *feature* concatena com a saída da rede principal após esta vetorizar os tokens do texto. Seguidamente, utiliza uma rede densa de 64 neurônios com a função linear responsável pela ativação. Após sair dessa camada, a camada de saída baseia-se numa estrutura densa com três neurônios ativados pela função *softmax* a fim de garantir melhor poder de classificação.

E. CRNN - Convolutional Recurrent Neural Network

Wang et al. (2016) trazem resultados bastante importantes para a compreensão de pequenos textos (como *tweets*, que possuem até 280 caracteres). Os autores argumentam que a extração de features locais proporcionadas por redes convolucionais combinada com a capacidade que redes recorrentes possuem para aprender relações entre elementos distantes em um texto podem resultar em uma arquitetura de performance superior a redes exclusivamente recorrentes. Com base nesse resultado, a quarta versão implementada combina essas arquiteturas, como pode ser visto na figura 3.

F. Busca de Tweets para avalição final

Para a extração dos tweets, criou-se uma conta de desenvolvedor no Twitter [8], seguindo-se os passos recomendados em um tutorial na internet [9]. Utilizou-se da biblioteca Tweepy [10] para a manipulação da API de requisições. Com ela, é possível extrair uma certa quantidade de tweets a cada 15 minutos (cerca de 2600). Assim, após algumas horas, foi possível extrair 20000 Tweets, sendo 10000 sobre Donald Trump, e 10000 sobre Joe Biden. Para pesquisar sobre um dado assunto, basta filtrar na query da API o que deseja-se buscar. É possível filtrar os dados recebidos por idioma (automaticamente reconhecido pelo Twitter). Assim, filtrou-se apenas Tweets em inglês. Finalmente, foi possível avaliar os sentimentos dos Tweets, com os métodos apresentados, expondo se são positivos, neutros ou negativos, na seção seguinte.

IV. Resultados e Discussões

Assim, fez-se o treinamento das redes com o dataset especificado, com dados de treinamento de validação. A seguir, expõe-se os dados de acurácia de cada uma das 4 redes, com duas taxas de aprendizado: 0,01 e 0,001. As imagens são vistas nas figuras 4 a 5. Na figura 6, exibe-se um comparativo entre as redes para cada taxa de aprendizado. Mais informações sobre como manipular as redes implementadas pode ser encontrada no readme do repositório no github [11].

Entre os gráficos apresentados, percebe-se que para a taxa de aprendizado menor, demora-se mais para ocorrer o overfit, em que o conjunto de treinamento continua a subir sua acurácia, enquanto o de validação estabiliza. Para a rede que usa apenas Embedding, vemos qe para a taxa de aprendizado 0,001 não há tempo suficiente para que haja de fato um aprendizado, o que poderia ser melhorado com mais épocas. Isto ocorre apenas nesta arquitetura pois é a que requer mais treinamento, por não usar parâmetros pré-treinados do GloVe. Percebe-se claramente na figura 6 que o Embedding é a que mais demora a crescer, por ter muito mais parâmetros em treinamento, como esperado.

Percebe-se que a rede que utiliza de CRNN é a que mais rapidamente possui overfit. Isto provavelmente se deve a maior quantidade de parâmetros, podendo ser melhor treinada, e a mistura de técnicas diferentes (convolucional e recorrente). Entretanto, percebe-se que há um grande viés para o set de treinamento, não apresentando significativas melhoras no set

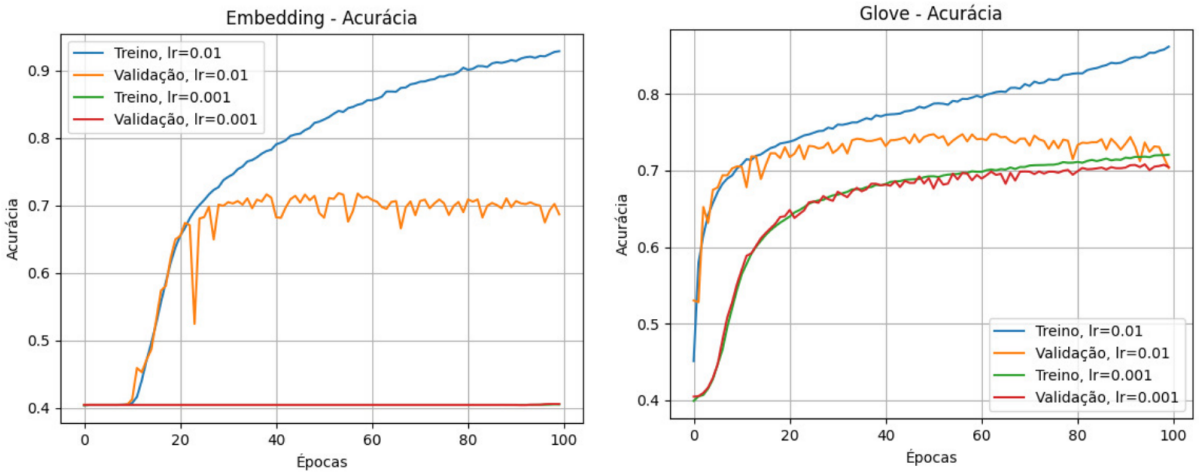

Fig. 4: Dados de acurácia para a rede que se utiliza somente de Embedding, e de GloVe.

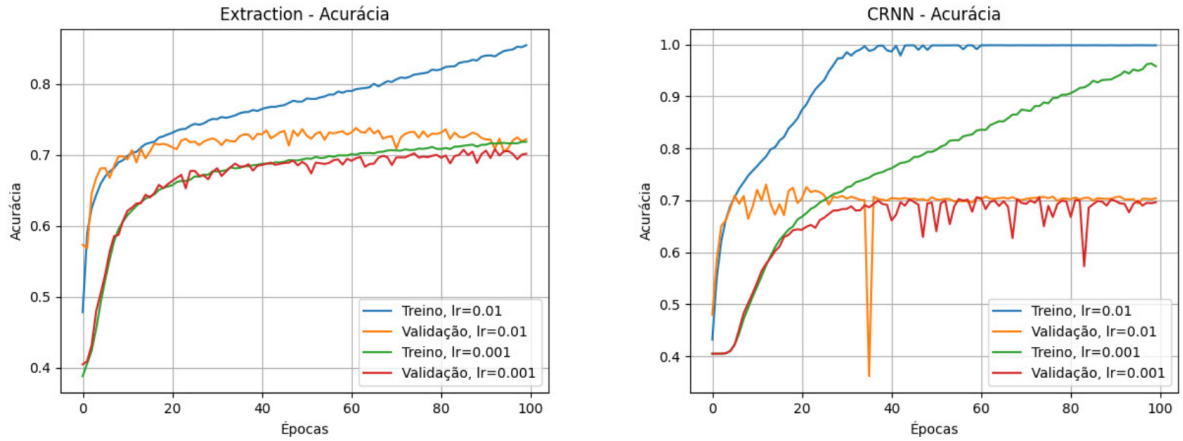

Fig. 5: Dados de acurácia para a rede que se utiliza de feature extraction, e de CRNN.

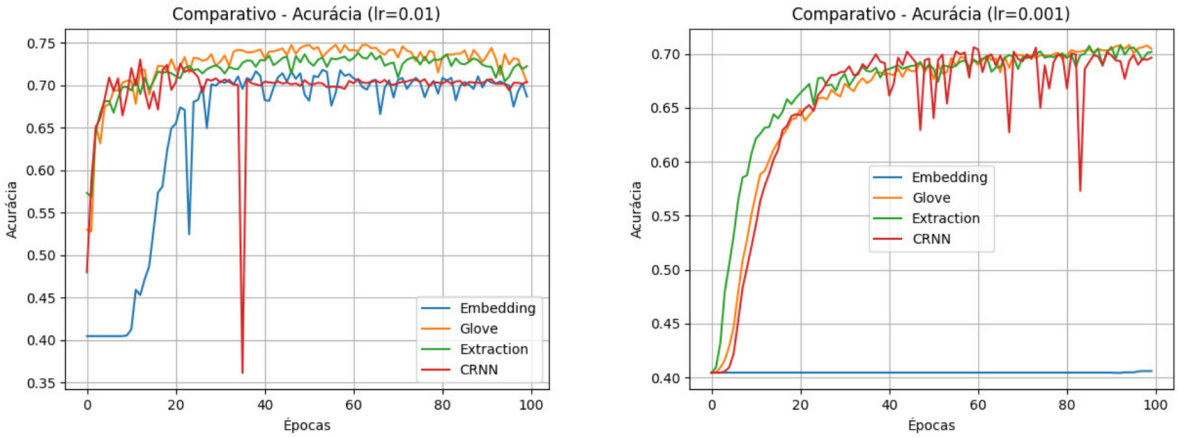

Fig. 6: Comparação das acurácias para learning rate = 0,01 e para 0,001.

de validação. Além disso, percebe-se que é a rede mais instável, o que deve ser devido ao caráter convolucional da rede, que possui um grande viés de localidade das palavras em sua aferição. Talvez com mais modificações e testes, seria possível obter um resultado interessante usando-se da mescla das arquiteturas.

Vê-se, no geral, uma estabilização próxima de 70% de acurácia no conjunto de validação, independentemente da acurácia no conjunto de treinamento, sendo que algumas chegam a 100% praticamente. Entretanto, com algumas das melhoras implementadas, vê-se um pequeno acréscimo acima de 70%, como na rede que usa de feature extraction, com taxa de aprendizado 0,01. Assim, essa rede será utilizada para o aferimento dos dados dos candidatos a presidência norte-americana.

Após a descrição de todo o processo desenvolvido desde o processamento, estudo de estruturas de redes neurais utilizando diversas ferramentas bem como o seu treinamento com publicações da rede social Twitter, finalmente utilizou-se da API fornecida pelo próprio Twitter para captação de publicações atuais a respeito dos dois candidatos a presidente dos Estados Unidos: Donald Trump e Joe Biden. Com a ferramenta de visualização conhecida como *WordCloud* verificou-se as palavras mais citadas nas bases de dados de ambos os candidatos. O resultado é representado pela Figura 7.

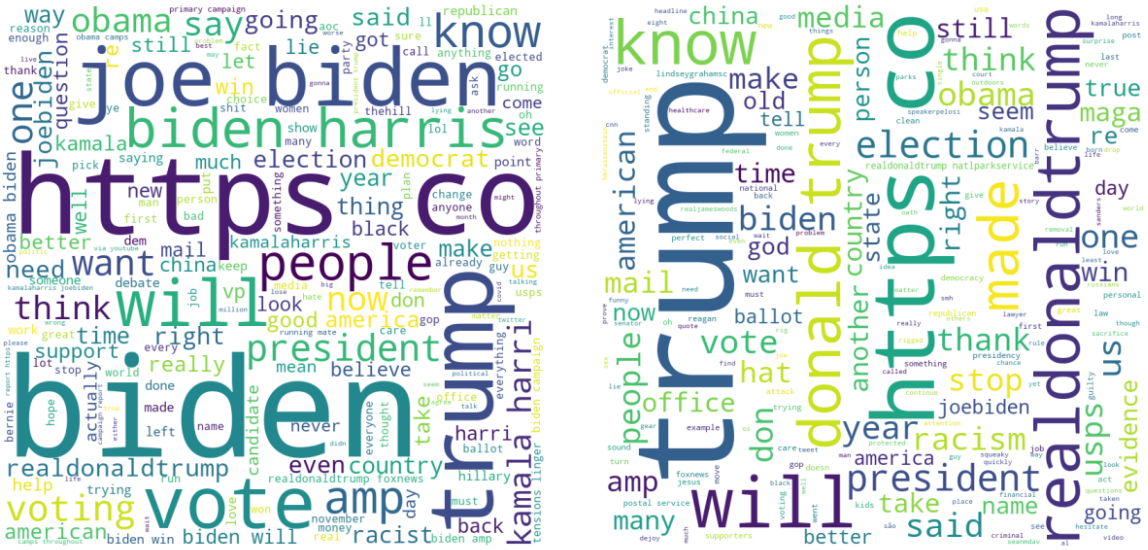

Fig. 7: Palavras mais escritas em publicações do Twitter relacionadas ao candidato Joe Biden (à esquerda) e Donald Trump (à direita).

Por fim, utilizou-se a melhor rede dentre as expostas anteriormente, sendo essa a rede com feature extraction, com learning rate de 0,01, para a aferição dos sentimentos dos tweets extraídos. Obteve-se os seguintes resultados:

	Donald Trump	Joe Biden
Negativos	32,4%	32,6%
Neutros	44,4%	44,9%
Positivos	23,2%	22,4%

TABLE II: Predição dos sentimentos de cerca de 10000 Tweets para cada candidato a presidência.

Percebe-se um predominante equilíbrio entre os resultados expostos. Dentre os motivos principais, aponta-se a acurácia relativamente baixa (70%), o que pode ser devido a uma tendência de neutralidade da rede. Assim, predominando-se tweets neutros, a rede erra menos para os extremos, o que minimizaria sua loss. Além disso, percebe-se empiricamente que parte dos tweets contém conteúdo relativo a ambos os candidatos, o que dificulta o trabalho da rede de determinar o sentimento do Tweet para um deles. Além disso, a grande presença de bots conhecida em redes sociais pode dificultar uma aferição real do sentimento da população através das fontes utilizadas. Além disso, utilizou-se uma relativamente pequena quantidade de Tweets, apenas 10000 para cada candidato, o que corresponde a cerca de 15 minutos corridos no Twitter (surpreendente!).

V. Conclusão

O trabalho aqui exposto evidencia o grande potencial de tais tecnologias, com a utilização de Transfer Learning, e modelos recentes de aprendizado de linguagem natural, como GloVe. Além do imenso aprendizado obtido em seu desenvolvimento, abre-se portas para toda uma outra frente de aprendizado de máquina, o não-supervisionado. Tal forma de aprendizado é muito visada, por se tratar de um dos caminhos para uma inteligência ”geral”, evitando-se a supervisão de humanos.

Além disso, é surpreendente a facilidade com o qual é possível por mãos em dados sensíveis e que podem aferir resultados surpreendentes. A manipulação das redes neurais pelo Tensorflow e a extração de Tweets pelas APIs de desenvolvedor são de fácil acesso, permitindo a praticamente qualquer um interessado no assunto se aprofundar e aproveitar das tecnologias apresentadas.

De toda forma, apesar das imperfeições dos modelos e das limitações apresentadas, os resultados obtidos permitem inferir um grande equilíbrio esperado para as eleições americanas, como também prevê-se pela mídia. Entretanto, a falta de precisão fina apresentada ainda indica que os modelos apresentados não devem ser levados como tomadores de decisão, e mais como ferramentas secundárias. Apesar de tratar-se apenas de um trabalho inicial no assunto, o resultado mostra que os modelos de inteligência artificial aplicados a extração de sentimentos e linguagem natural ainda conferem muito espaço para estudos e aprimoramentos.

REFERENCES

[1] Ian Goodfellow and Yoshua Bengio and Aaron Courville. 2016. Deep Learning. MIT Press.
[2] Stuart J. Russell and Peter Norvig. 2009. Artificial Intelligence: A Modern Approach. 3rd edition. Prentice Hall.
[3] https://www.youtube.com/watch?v=gQddtTdmG_8
[4] https://stats.stackexchange.com/questions/335793/what-is-difference-between-keras-embedding-layer-and-word2vec
[5] Cícero N. dos Santos e Maíra Gatti. 2014. Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts. *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*.
[6] Jeffrey Pennington and Richard Socher and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. *Empirical Methods in Natural Language Processing (EMNLP)*.
[7] Xingyou Wang and Weijie Jiang and Zhiyong Luo. 2016. Combination of Convolutional and Recurrent Neural Network for Sentiment Analysis of Short Texts. *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*.
[8] https://developer.twitter.com/en
[9] https://www.youtube.com/watch?v=RssGfmtyn4A&feature=youtu.be
[10] https://readthedocs.org/projects/tweepy/downloads/pdf/latest/
[11] https://github.com/mouradfelipe/twitter_analysis