Deteção de Fake News

1.

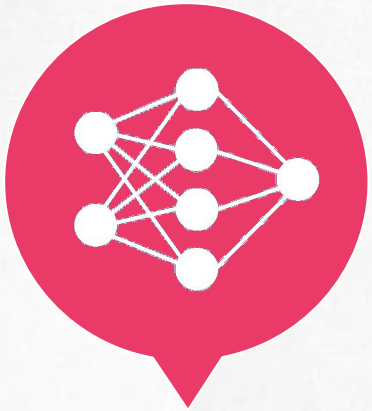
Motivação

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.

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

2.

Objetivo



Modelo Classificador

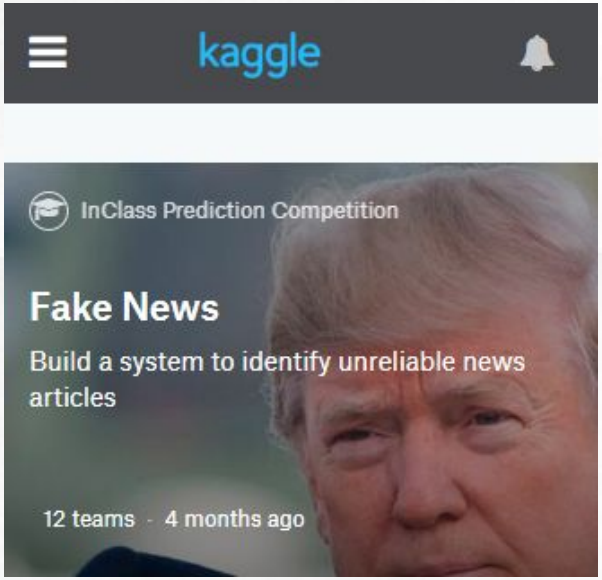
Do ponto de vista da NLP (Neuro-linguistic programming), o fenômeno Fake News oferece uma oportunidade interessante e valiosa para identificar padrões que podem ser codificados em um modelo classificador.

3.

Os Dados

https://www.kaggle.com/c/fake-news

Build a system to identify unreliable news articles



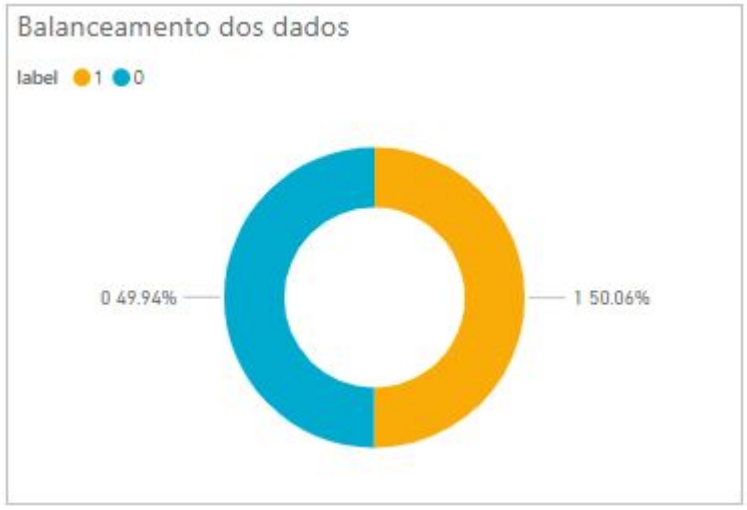
Data (total 5 columns):

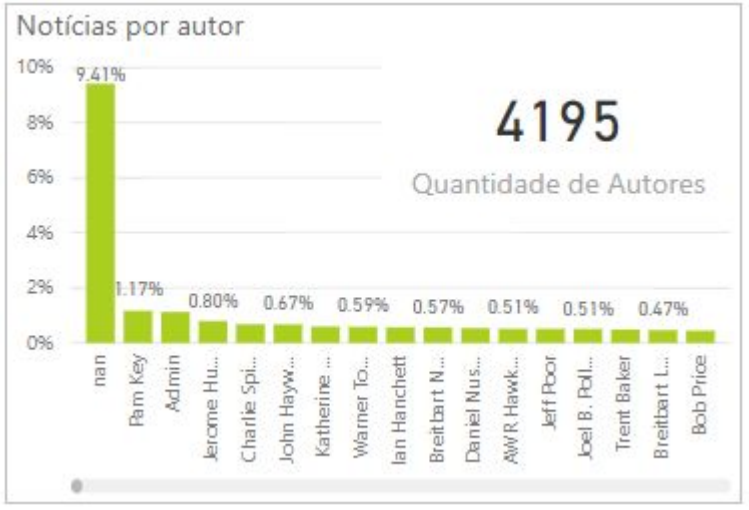
id	20800 non-null
title	20242 non-null
author	18843 non-null
text	20761 non-null
label	20800 non-null

Build a system to identify unreliable news articles

	id	title	author	text	label
0	0	House Dem Aide: We Didn't Even See Comey's Let...	Darrell Lucus	House Dem Aide: We Didn't Even See Comey's Let...	1
1	1	FLYNN: Hillary Clinton, Big Woman on Campus - ...	Daniel J. Flynn	Ever get the feeling your life circles the rou...	0
2	2	Why the Truth Might Get You Fired	Consortiumnews.com	Why the Truth Might Get You Fired October 29, ...	1
3	3	15 Civilians Killed In Single US Airstrike Hav...	Jessica Purkiss	Videos 15 Civilians Killed In Single US Airstr...	1
4	4	Iranian woman jailed for fictional unpublished...	Howard Portnoy	Print \nAn Iranian woman has been sentenced to...	1

Build a system to identify unreliable news articles



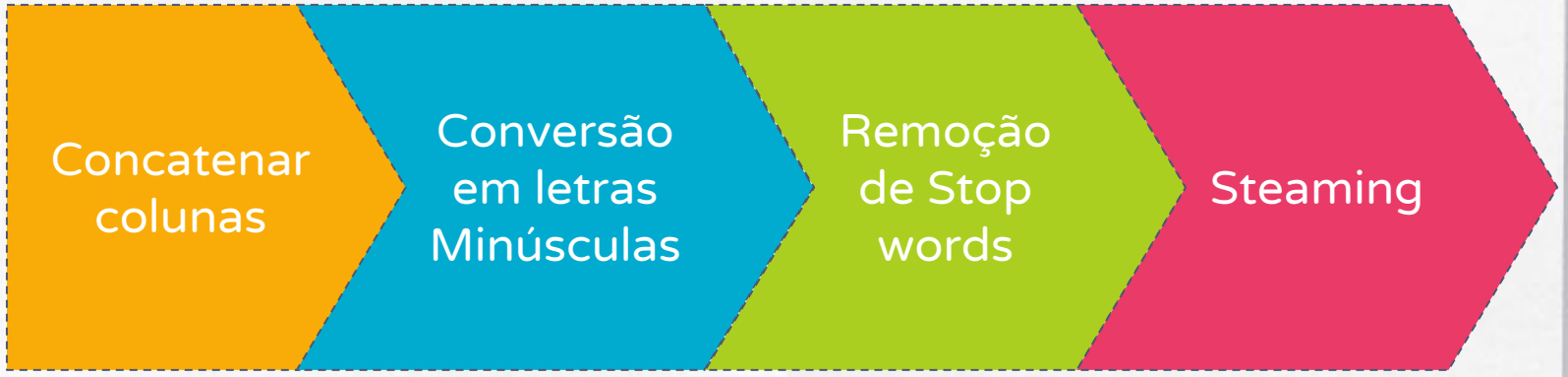


4.
Metodologia

NLP, Naive Bayes, Adaboost, LSTM e CNN

Processamento de Linguagem Natural (NLP)

Os dados de texto requerem uma preparação especial antes que você possa começar a usá-lo para modelagem preditiva.



Processamento de Linguagem Natural (NLP)

- Modelo Bag-of-Words
- Contagens de palavras com o CountVectorizer
- Frequência de palavras com TfidfVectorizer

Processamento de Linguagem Natural (NLP)

```
#top 8000 features
count_vectorizer = CountVectorizer(ngram_range=(1, 3), max_features=8000)
count_vectorizer.fit(data['all'].values)
doc_array = count_vectorizer.transform(data['all'].values).toarray()
frequency_matrix = pd.DataFrame(doc_array, columns=count_vectorizer.get_feature_names())

frequency_matrix
```

	aaron	aaron klein	abandon	abbott	abduct	abdullah	abedin	abid	abil	abl	...
0	0	0	0	0	0	0	0	0	0	0	...
1	0	0	0	0	0	0	0	0	0	0	...
2	0	0	0	0	0	0	0	0	0	1	...
3	0	0	0	0	0	0	0	0	0	2	...
4	0	0	0	0	0	0	0	0	0	0	...
5	0	0	0	0	0	0	0	0	0	0	...
6	0	0	0	0	0	0	0	0	0	0	...
7	0	0	1	0	0	0	0	0	0	0	...
8	0	0	0	0	0	0	0	0	0	3	...

... outra forma de lidar com tarefas preditivas em Machine Learning, principalmente quando as informações disponíveis são incompletas ou imprecisas, é por meio do uso de algoritmos baseados no **teorema de Bayes**.

de, CARVALHO, André Carlos Ponce de Leon F., FACELI, Katti, LORENA, Carolina, GAMA, João. Inteligência Artificial - Uma Abordagem de Aprendizado de Máquina. LTC, 08/2011

Naive Bayes

```
#tfidf

transformer = TfidfTransformer(smooth_idf=False)
count_vectorizer = CountVectorizer(ngram_range=(1, 3))
counts = count_vectorizer.fit_transform(data['all'].values)
features = transformer.fit_transform(counts)
```

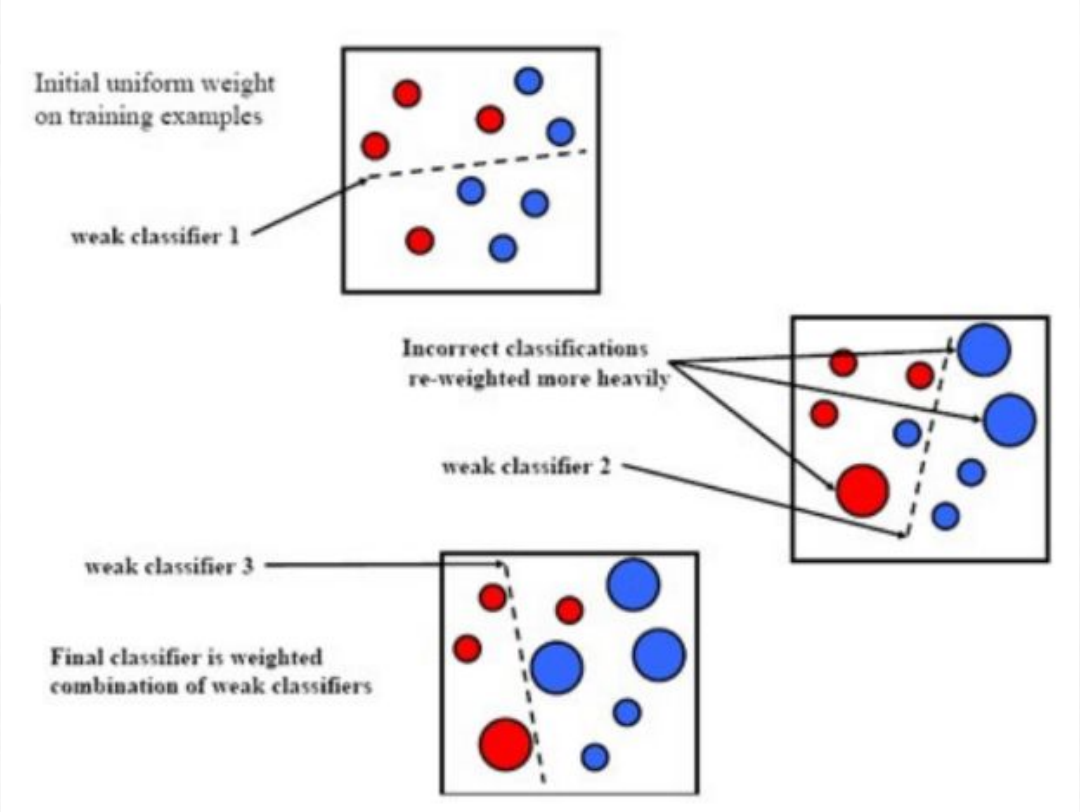
```
#Naive Bayes

NB = MultinomialNB()
cvscores = []
for train, test in kfold.split(X, Y):
  NB.fit(X[train], Y[train])
  scores = NB.score(X[test], Y[test])
  cvscores.append(scores * 100)

print("ACC %.2f%% (std +/- %.2f%%)" % (np.mean(cvscores), np.std(cvscores)))
```

```
ACC 84.92% (std +/- 0.77%)
```

Um classificador **AdaBoost** é um meta-estimador que começa ajustando um classificador no conjunto de dados original e então ajusta cópias adicionais do classificador no mesmo conjunto de dados, mas onde os pesos das instâncias classificadas incorretamente são ajustados de forma que os classificadores subseqüentes se concentrem mais casos difíceis..

http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html

Adaboost


Initial uniform weight
on training examples
weak classifier 1
Incorrect classifications
re-weighted more heavily
weak classifier 2
weak classifier 3
Final classifier is weighted
combination of weak classifiers

Adaboost

```
#tfidf

transformer = TfidfTransformer(smooth_idf=False)
count_vectorizer = CountVectorizer(ngram_range=(1, 3))
counts = count_vectorizer.fit_transform(data['all'].values)
features = transformer.fit_transform(counts)
```

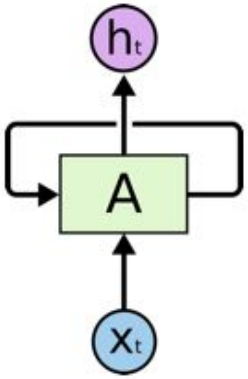
```
#AdaBoost

Adab= AdaBoostClassifier(DecisionTreeClassifier(max_depth=1),n_estimators=5)
cvscores = []
for train, test in kfold.split(X, Y):
  Adab.fit(X[train], Y[train])
  scores = Adab.score(X[test], Y[test])
  cvscores.append(scores * 100)

print("ACC %.2f%% (std +/- %.2f%%)" % (np.mean(cvscores), np.std(cvscores)))
```

```
ACC 93.74% (std +/- 0.40%)
```

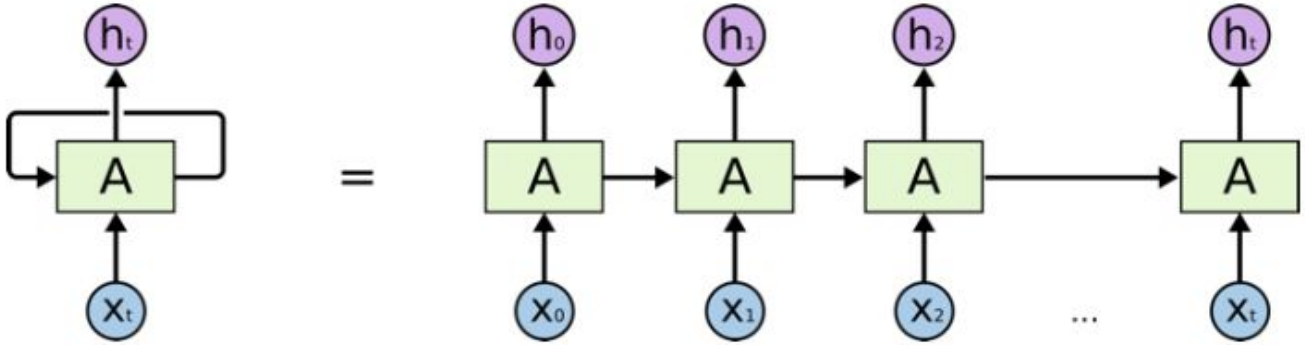
LSTM

A LSTM é um tipo de rede neural recorrente. Foi inicialmente proposta por Hochreiter e Schmidhuber 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 .

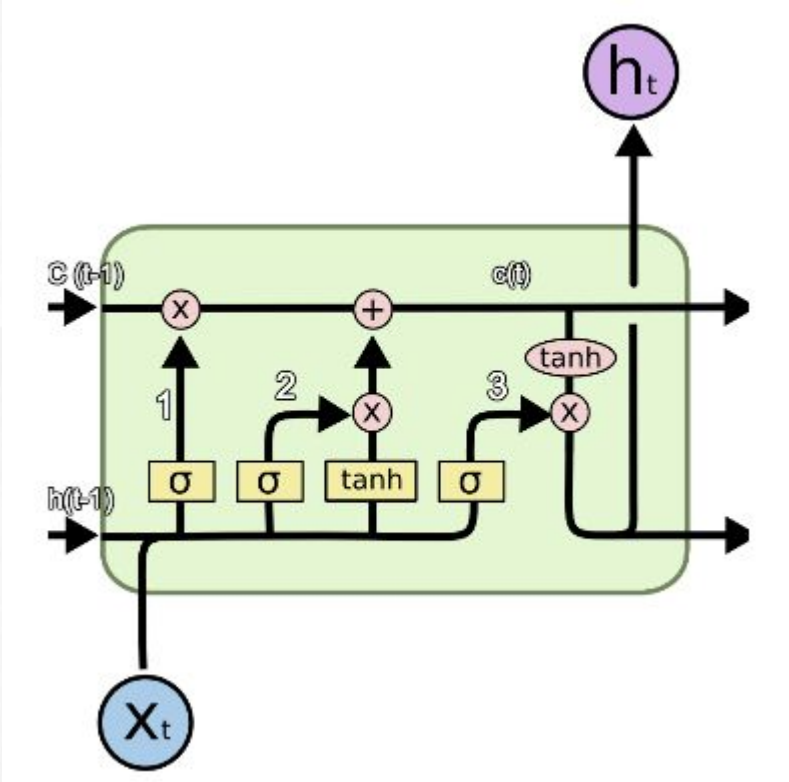
LSTM



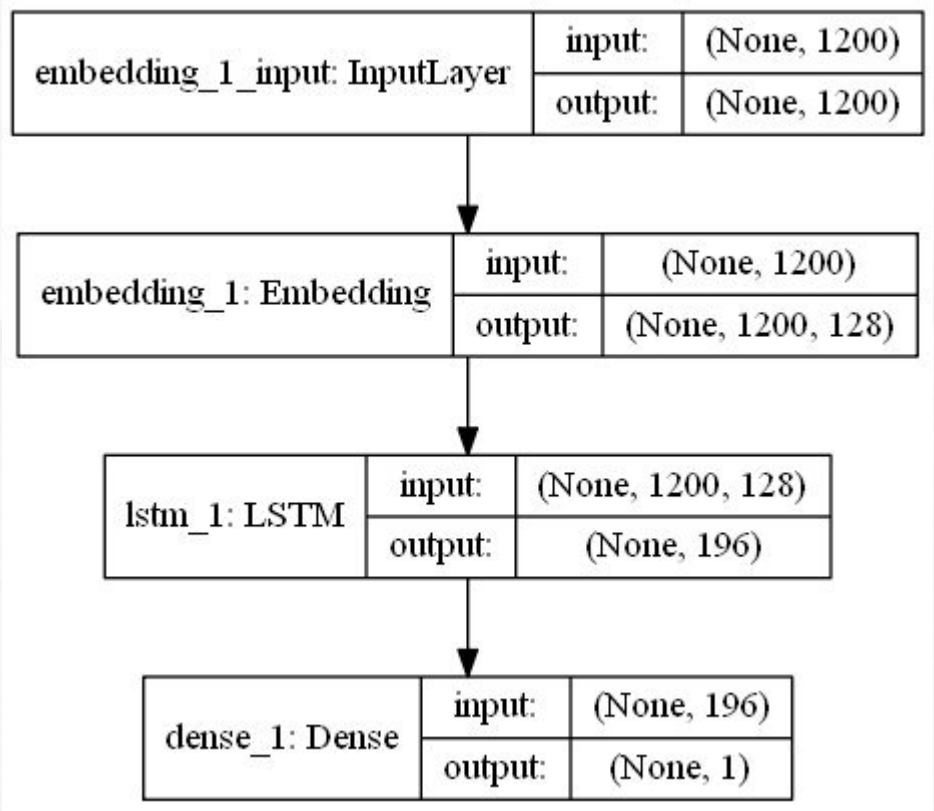
A typical RNN (Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/)



LSTM



LSTM

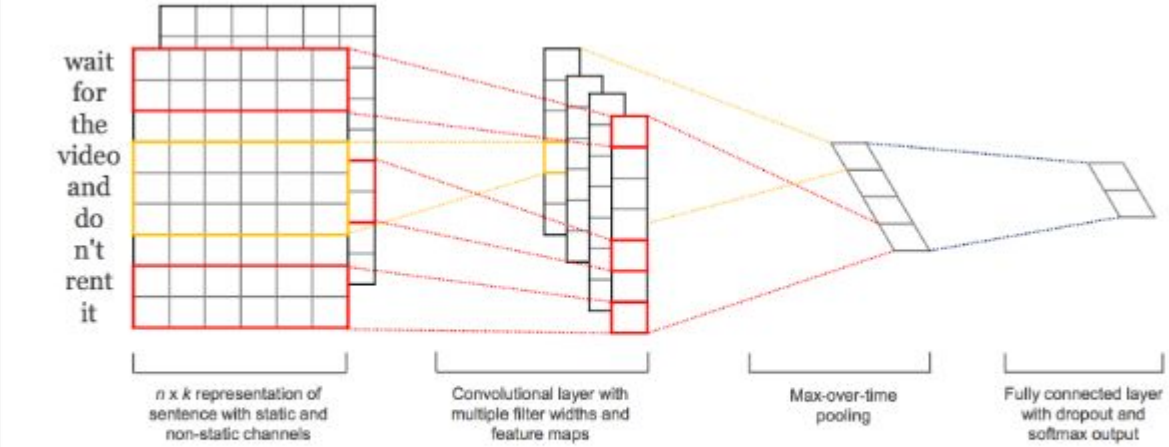

embedding_1_input: InputLayer
input: (None, 1200)
output: (None, 1200)
embedding_1: Embedding
input: (None, 1200)
output: (None, 1200, 128)
lstm_1: LSTM
input: (None, 1200, 128)
output: (None, 196)
dense_1: Dense
input: (None, 196)
output: (None, 1)

CNN+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.

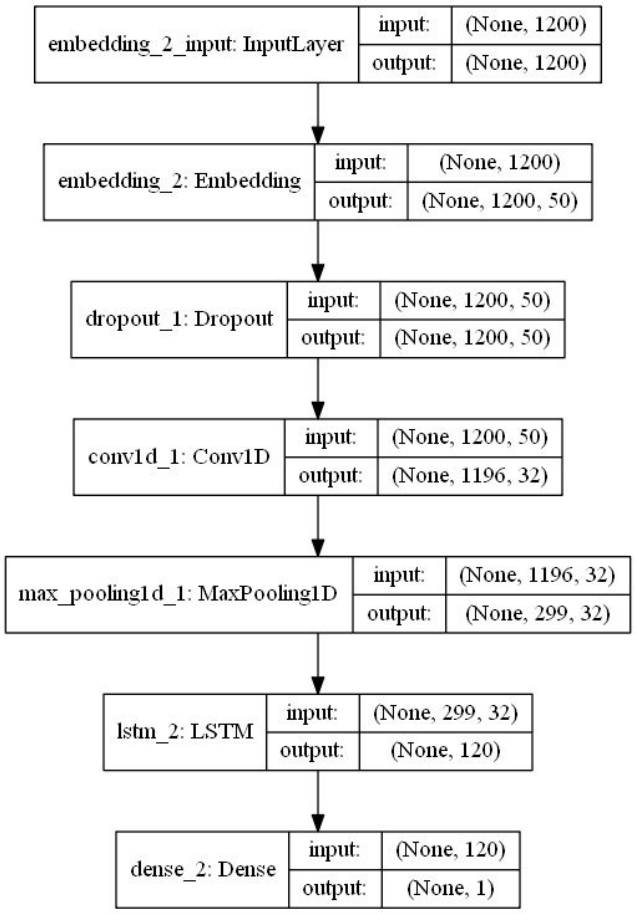
Redes neurais convolucionais se destacam em aprender a estrutura espacial em dados de entrada.

CNN+LSTM



Depiction of the multiple-channel convolutional neural network for text.
Taken from "Convolutional Neural Networks for Sentence Classification."

CNN+LSTM


embedding_2_input: InputLayer
input: (None, 1200)
output: (None, 1200)
embedding_2: Embedding
input: (None, 1200)
output: (None, 1200, 50)
dropout_1: Dropout
input: (None, 1200, 50)
output: (None, 1200, 50)
conv1d_1: Conv1D
input: (None, 1200, 50)
output: (None, 1196, 32)
max_pooling1d_1: MaxPooling1D
input: (None, 1196, 32)
output: (None, 299, 32)
lstm_2: LSTM
input: (None, 299, 32)
output: (None, 120)
dense_2: Dense
input: (None, 120)
output: (None, 1)

5.

Experimentos e Resultados

Naive Bayes 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

Features

count	20.800
mean	395
min	0
35%	200
50%	297
90%	755
97%	1.200
max	10.890

LSTM e CNN+LSTM - Word-padding

	LSTM ACC(%)	CNN+LSTM ACC(%)
1200 termos	**91.43**	94.08
800 termos	89.16	90.76
400 termos	89.13	93.98
297 termos	90.39	93.26
200 termos	89.04	**96.62**

LSTM e CNN+LSTM - Zero-padding

	LSTM ACC(%)	CNN+LSTM ACC(%)
1200 termos	50.28	78.15
800 termos	66.51	76.26
400 termos	89.30	90.21
297 termos	89.50	**96.08**
200 termos	**89.88**	95.25

6.
Conclusão

Conclusão

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+LSTM, ficam mais robustos para NLP.

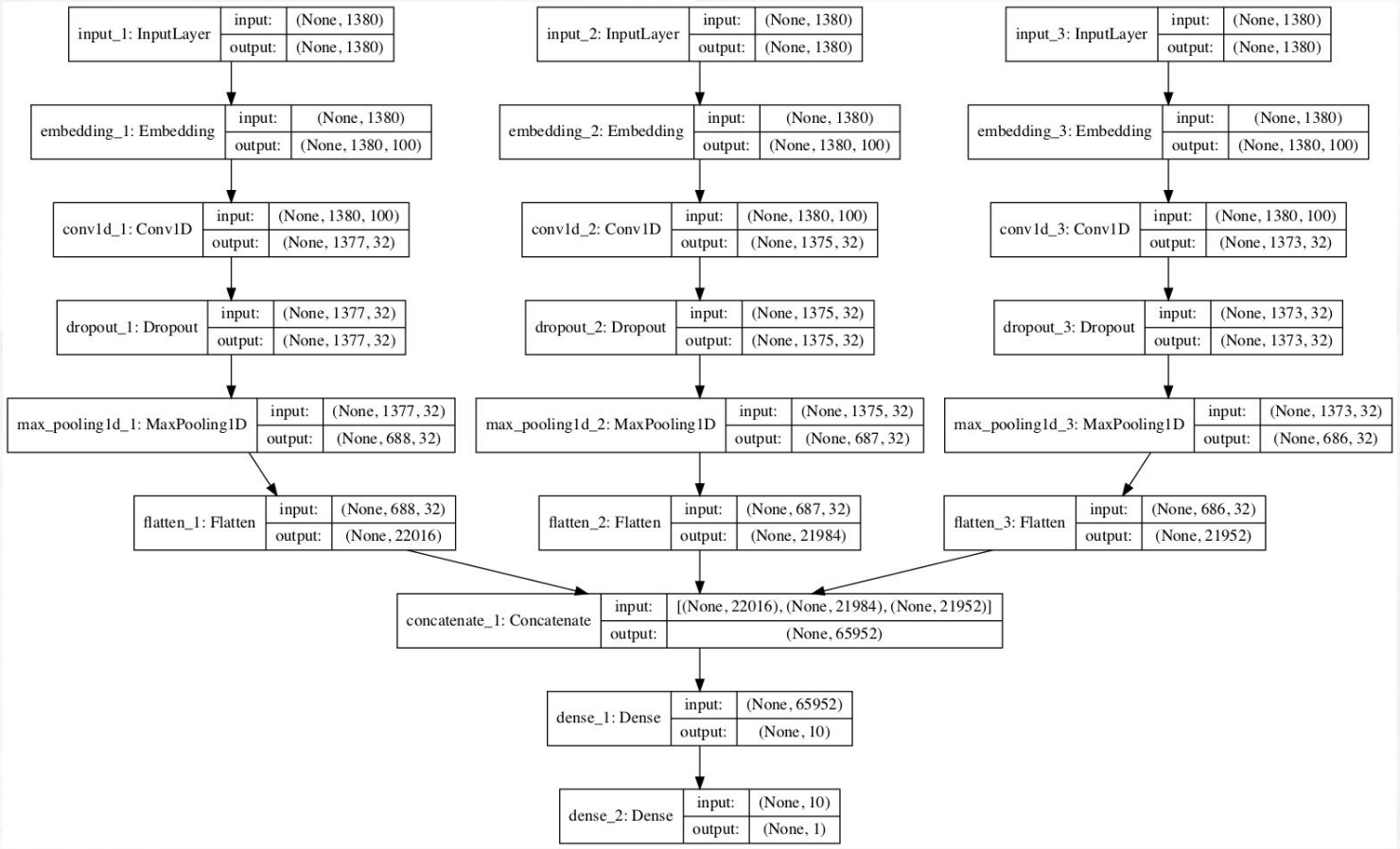
6.

Trabalhos Futuros

Trabalhos Futuros

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.

Trabalhos Futuros



Obrigado!

lucasacparreiras@gmail.com