Deteccion de agresividad en texto

Mireya Paredes López
José Crispín Alvarado Calderón

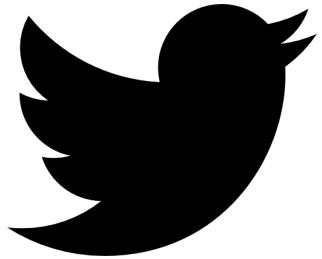
Diplomado en Deep Learning, Julio 2019
INAOE-Puebla

01

Planteamiento del problema

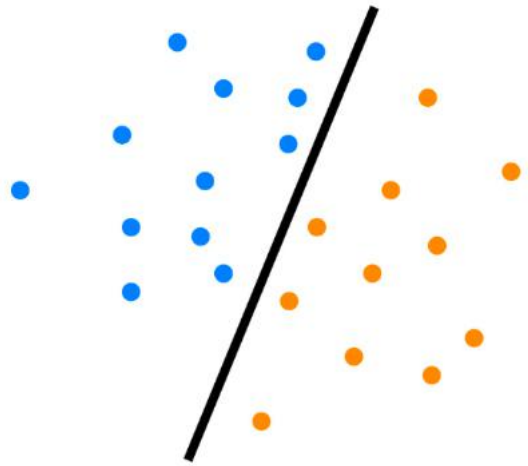
Necesidad del estudio

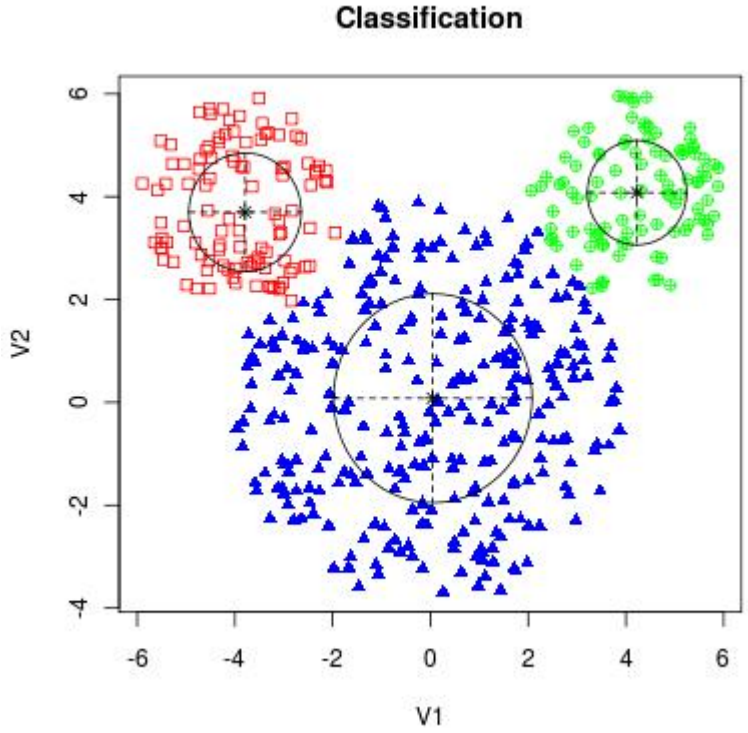
Uno de los principales riesgos a los que nos exponemos en redes sociales electrónicas, es a ser sujetos de la ciber-violencia, esto es, cualquier forma de agresión en medios digitales.



Retos en la detección de agresividad en texto

Algunos de los métodos existentes tienen un enfoque forense por lo que carecen de una funcionalidad crítica: no permiten anticipar agresiones y/o prevenirlas



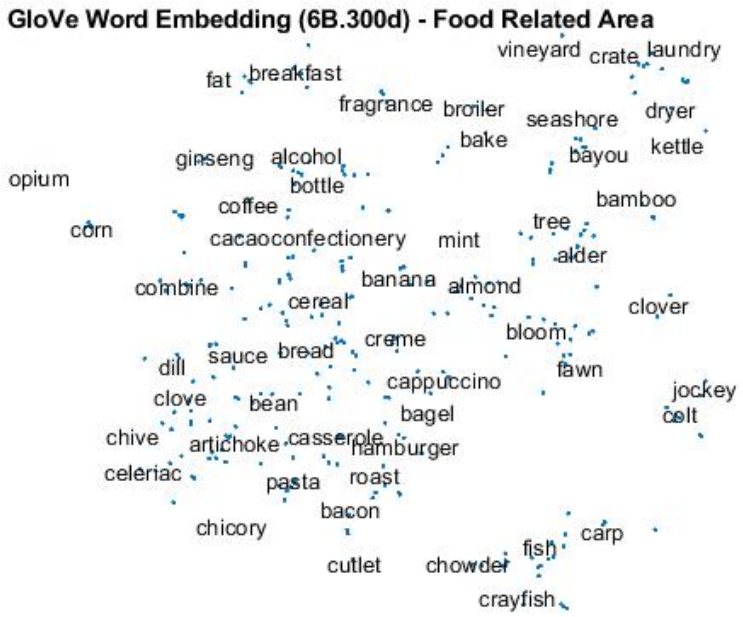


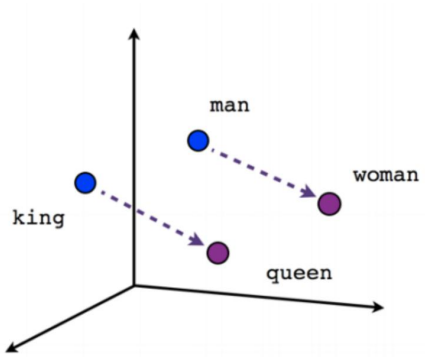
Soluciones actuales

Modelos probabilísticos

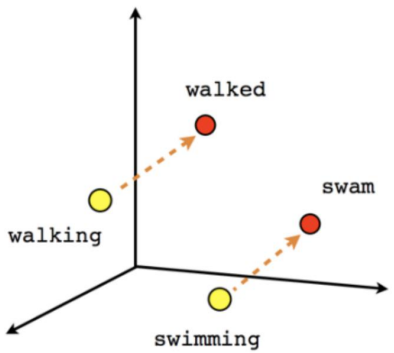
Modelos predictivos para datos secuencuales

Representaciones concisas de palabras (en un espacio vectorial)





Male-Female



Verb tense

Descripción de los datos y procedimiento

- Conjunto de datos
  - 7700 tweets (en español) etiquetados como agresivos o no agresivos

- Procedimientos utilizados
  - Deep feedforward network
  - Long Short-Term Memory (LSTM)
  - Convolutional Neural Network (CNN)

Herramientas utilizadas

- Keras (herramientas de redes neuronales)
- NLTK (kit de herramientas para el procesamiento del lenguaje natural)

Ejemplo de los datos

AggressiveDetection_train.txt

Users ▸ CRISPIN ▸ Documents ▸ DiplomadoDL2019 ▸ Proyecto ▸ AggressiveDetection_train.txt

```
104  @USUARIO @USUARIO Valiente? Defendiendo a la HDP de Mamá Rosa?? No es más que basura intelectual y encima priist
105  @USUARIO @USUARIO Vergüenza ese hdp no la conoce mientras le rinda dividendos (transas)
106  @USUARIO @USUARIO Vos tu madre hdp
107  @USUARIO @USUARIO y si este #hermano #HDP @USUARIO Tuyo ;se queda en #PANAMÁ ? El 🐷 d #JAVIDU sonreía en #GUA
108  @USUARIO @USUARIO 😂😂😂...este hijo de su asesina madre está idiota, enfermo, mariguano, alguien que lo bloque
109  @USUARIO ¡POBRE PENDEJO LAMECULOS DE LOS YANQUIS!
110  @USUARIO 10m.... Jajaja!. Esperando acá en mixcoac ya más de 1 hora. Como siempre, los chóferes saltándose calle
111  @USUARIO A pesar de ese maldito pelón tendencioso HDP de Medrano
112  @USUARIO ah robar lo único se saben hacer hdp
113  @USUARIO Ahorita les aviso para que se tomen fotos y se las manden. Para que no se preocupe mas. Viejo Lameculos
114  @USUARIO Alan por favor RT. Su madre, familia y amigos estamos muy preocupados. Necesitamos encontrarlos 😔
115  @USUARIO Alv!!! Tu, apoyando a México? Mañana va a llover a raja madre
116  @USUARIO Anda a dialogar con tu madre que posiblemente es la única que le queda un poco de paciencia diputado d
117  @USUARIO Así como esta chivando a si madre las Chivas sin habrá ni un partido😂😂😂
118  @USUARIO baia baia hdp @USUARIO está buena la robadera verdad? #NadieSeImaginaQ @USUARIO
119  @USUARIO Borracha jajajja me haces el día hdp jajajja también te amo y te extraño 🤍🤍
120  @USUARIO Borrala hdp
121  @USUARIO buenas noches solo vengo a decirle que chingues a tu madre tu,el america y el desaparecido atlante .i.
122  @USUARIO Bueno ya dense en su madre
123  @USUARIO Bueno, no es como que el que hizo a Escobar hablara siquiera español... Sí, le vale madre (a la producc
124  @USUARIO Buenos diaaas☔☔ a toda madre.....que siga lloviendo
125  @USUARIO Chiga tu loca Madre. Soy Panista cuando hubo un sondeo de opinión a la Base triste Peñanietista ????
126  @USUARIO Chinga a tu madre
127  @USUARIO Chinga tu madre estoy cenando culero 🙄
```

02

Pre-procesamiento del texto

Pre-procesamiento de los datos

- Para dar consistencia, se convirtieron las cadenas de texto a minúsculas
- Se removieron todas las menciones a @USUARIO y los links <URL>
- Se quitaron todos los carácteres especiales y signos de puntuación
- Se removieron todas las menciones de RT y números
- Se quitaron las *stop words*

```python
def preprocessing_tweet(tweet):
    ### Puts everithing in lowecase
    tweet = tweet.lower()
    ### Remove all mentions to @USUARIO
    tweet = re.sub('@\w+', '', tweet)
    ### Remove links <URL>
    tweet = re.sub('<url>', '', tweet)
    ### Remove special character and punctuation marks
    tweet = re.sub('[¡!"\'#$%&()*+,-./:;<=>¿?@[\]^_`{|}~\n]', '', tweet)
    ### Remove all mentions to retweet
    tweet = re.sub('rt','', tweet)
    ### Remove numbers
    tweet = re.sub('[0-9]+','', tweet)
    return tweet

def remove_stopwords(tweet):
    stop_words = stopwords.words('spanish')
    tweet = ' '.join([word for word in tweet.split() if word not in stop_words])
    return tweet

def clean_text(lines_t):
    tweets = []
    for line in lines_t:
        tweet = preprocessing_tweet(line)
        tweet = remove_stopwords(tweet)
        tweets.append(tweet)

    return tweets
```

s x maldito hdp', 'dejen tirar basura hdp gente puerca', 'hdp sale caminando mantiene laburo cdo contratos dice q d eben tener buena conductaeste encima dice q pedo😠', 'huecos hdp den cara mal nacidos servidores públicos putas ll eguen renunciar regresen bastardos', 'jajajaja mamá apae luchona naca americanista ocurrente putita', 'jajajaja luc hona suena feo jaja', 'mamen asco poada mas argumentos joder recurren poca madre repito poca madre', 'madre mejor c omprométanse revisar porweurias epn estafamaestra', 'chinge asu puta madre tico tan mediocre pais ve programas mexi co 😂😂😂 chinga madre tico pemdwjo', 'chingue madre largue verga 😡', 'sabes hdp metes vos', 'dice lameculos chup a pitos', 'si lameculos coparmex simpatizantes opus day grupitos pedorros estilo honor verdad', 'verdaderamente dud a hijos puta madre perdón ojalá robado haga mueran vida miserables', 'chinga madre si sabes cómo país si visto pae pobre menos si toleras mamadas presidente mierda', 'juntaron v váyanse chingar paloma madre bai', '😡 wey hdp acab as mamarrrrr😂😂😂😂😂😂', 'millones pagarán bolsa izquierda después entregaron saquearon nación poca madre', 'll ama tener madre vergüenza bien dicen gobierno merece', 'huevo chingada madre', 'pinche puta vida hdp pendejo', 'aca ba primaria pública hijo puta naca madre', 'chinguen madre si bien saben hacen pendejos todas líneas vez chinguen p uta madre', 'daño irle america seguro machorra frustrada jajaja', 'discriminación madre', 'pae mia chinga madre sal udos', 'seguro edos hdp rateros verdad', 'señor superpoder hablar mierda cagar madre cambiar color piel color naran ja🤷‍♀️', 'video tomado calle principal santa madre dios abarca tramo señalado', 'misma mugrosa tan planeado valen ma dre malditos lacras', 'chingado tronco inflado sigue viviendo gol chinguen reputa madre', 'mentada madre', 'foto ma dre 😍😭 jalo jalo quieran 🤗🤗', 'verdadero ejemplo hijo puta mas va vender rey supremo aquí poca madre', 'dices madre pues q varas medirías políticos q roban', 'habla soluciones ofrece alternativas irresponsable abusivo hdp', ' hija madre 😂😂😂😂🖕🖕🤚', 'hijo puta madre deja almeyda trabajar agusto', 'jajajajajajaja hdp sido buen lunes 😪 😵', 'jajajajjajaaj hdp metas ahre pq pegan presos blda jajaja iman 😂', 'lameculos igual', 'presenta contratiempo

03

Preparación del modelo

“Tokenización” de la colección de tweets

- Se convirtió la colección de tweets en una matriz de tokens (enteros) generando una representación de los contadores

```
Found 17022 unique tokens
{'verga': 1, 'madre': 2, 'putas': 3, 'putos': 4, 'si': 5, 'loca': 6, 'pinche': 7, 'puta': 8, 'bien': 9, 'hdp': 10,
'joto': 11, 'marica': 12, 'ser': 13, 'q': 14, 'así': 15, 'vale': 16, 'puto': 17, 'pinches': 18, 'ver': 19, 'solo':
20, 'vida': 21, 'voy': 22, 'tan': 23, 'mierda': 24, 'luchona': 25, 'quiero': 26, 'siempre': 27, 'mejor': 28, 'ahora
': 29, 'va': 30, 'hoy': 31, 'pendejo': 32, 'hacer': 33, 'día': 34, 'gente': 35, 'hijo': 36, 'vez': 37, 'hace': 38,
'hijos': 39, 'maricon': 40, 'mas': 41, 'jajaja': 42, 'jajajaja': 43, 'toda': 44, 'alguien': 45, 'van': 46, 'todas':
47, 'pues': 48, 'madres': 49, 'gusta': 50, 'chingada': 51, 'pendeja': 52, 'dos': 53, 'mamá': 54, 'cada': 55, 'amor
': 56, 'méxico': 57, 'mal': 58, 'tener': 59, '😂': 60, 'vas': 61, 'mamar': 62, 'neta': 63, 'nadie': 64, 'sé': 65,
'mundial': 66, 'verdad': 67, 'bueno': 68, 'cosas': 69, 'mamando': 70, 'aquí': 71, 'puedo': 72, 'años': 73, 'putita
': 74, 'wey': 75, 'da': 76, 'decir': 77, 'ganas': 78, 'dice': 79, 'ir': 80, 'mañana': 81, 'alv': 82, 'culo': 83, 'q
uiere': 84, 'v': 85, 'nunca': 86, 'mundo': 87, 'igual': 88, 'días': 89, 'mujeres': 90, 'pedo': 91, 'pendejos': 92,
'chingas': 93, 'puede': 94, 'chingar': 95, 'd': 96, 'mujer': 97, 'mil': 98, 'ojalá': 99, 'menos': 100, 'cómo': 101,
'hacen': 102, 'valer': 103, 'amigos': 104, 'mames': 105, 'casa': 106, 'jaja': 107, 'chinga': 108, 'sabes': 109, 'di
cen': 110, 'amo': 111, 'ahí': 112, 'quieren': 113, 'luego': 114, 'digo': 115, 'después': 116, 'cuenta': 117, 'sient
o': 118, 'sólo': 119, 'creo': 120, 'ustedes': 121, 'encanta': 122, 'veces': 123, 'sabe': 124, 'gracias': 125, 'pasa
': 126, 'tiempo': 127, 'vamos': 128, 'chinguen': 129, 'fotos': 130, 'pa': 131, 'deja': 132, 'unas': 133, 'maricón':
134, 've': 135, 'mismo': 136, 'pueden': 137, 'quieres': 138, 'dinero': 139, 'quién': 140, 'jajajajaja': 141, 'veo':
142, 'caga': 143, 'perra': 144, 'buena': 145, 'foto': 146, 'trabajo': 147, 'hombres': 148, 'mientras': 149, 'rica':
150, 'noche': 151, 'rico': 152, 'peor': 153, 'personas': 154, 'digan': 155, 'buen': 156, 'dan': 157, 'hija': 158, '
```

```
(7700, 22)
[[  0   0   0 ... 522 424 452]
 [  0   0   0 ... 237   1 244]
 [  0   0   0 ...   4 162 228]
 ...
 [  0   0   0 ...   5  19  26]
 [  0   0   0 ...  16 376   3]
 [  0   0   0 ...   0   1 722]]
```

04

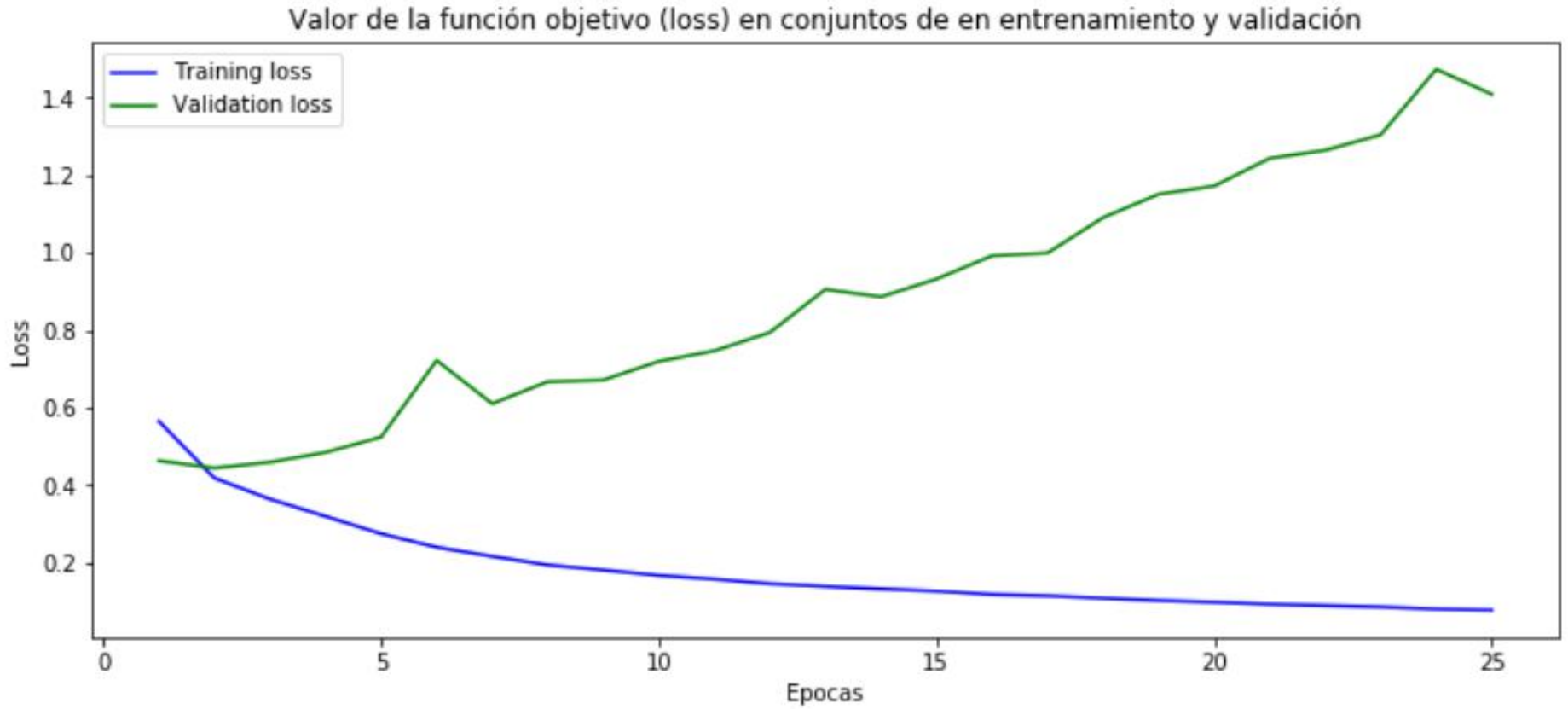
Implementación de los modelos

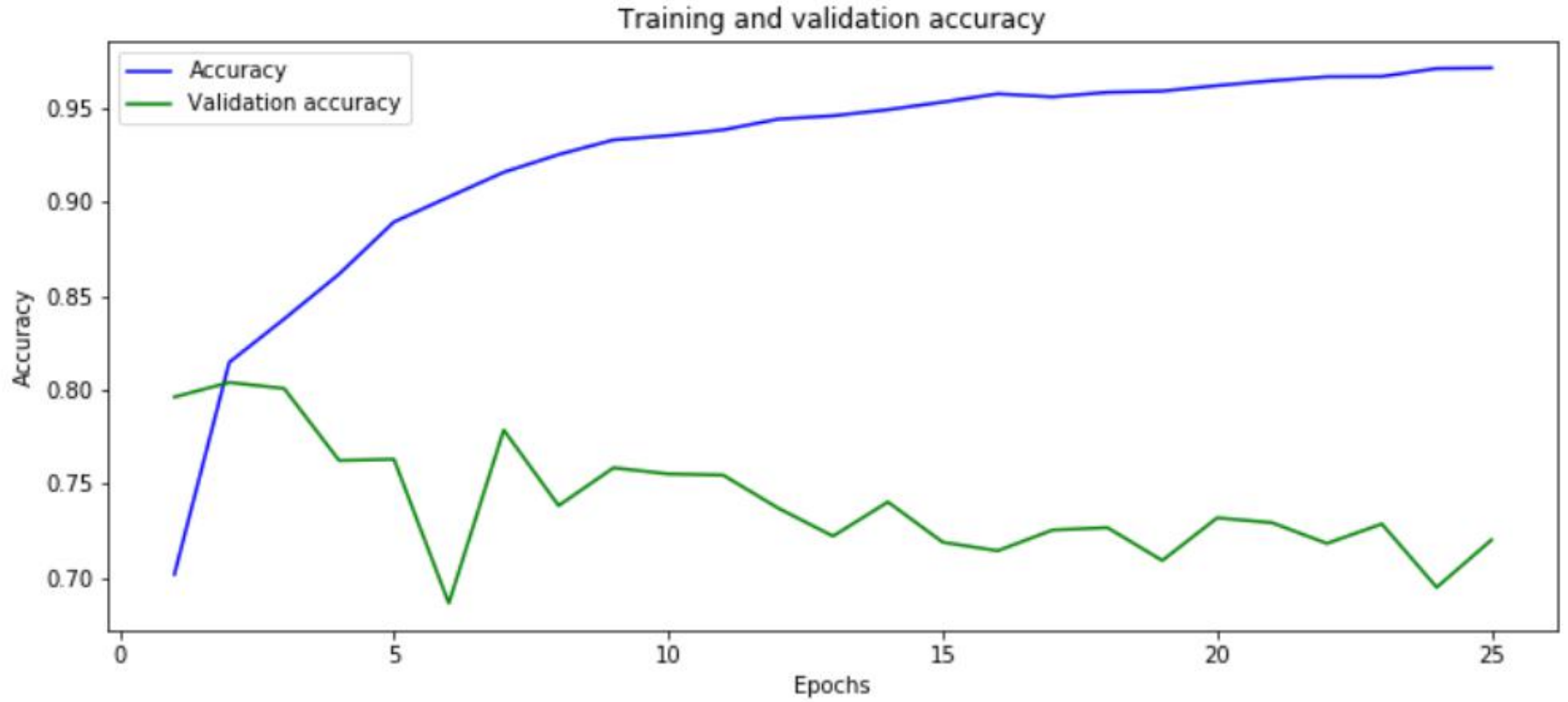
Deep Feedforward Network

```
model_1 = Sequential()
model_1.add(Embedding(max_words, embedding_dim, input_length = max_len))
model_1.add(Flatten())
model_1.add(Dense(32, activation = 'relu'))
model_1.add(Dense(1, activation = 'sigmoid'))
model_1.summary()

model_1.compile(optimizer = 'rmsprop',
                loss = 'binary_crossentropy',
                metrics  =['accuracy'])

history_1 = model_1.fit(X_train, Y_train,
                    epochs = 25,
                    batch_size = 32,
                    validation_split = 0.2)
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 22, 22)            22000
_________________________________________________________________
flatten_1 (Flatten)          (None, 484)               0
_________________________________________________________________
dense_1 (Dense)              (None, 32)                15520
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 33
=================================================================
Total params: 37,553
Trainable params: 37,553
Non-trainable params: 0
```

Desempeño del modelo



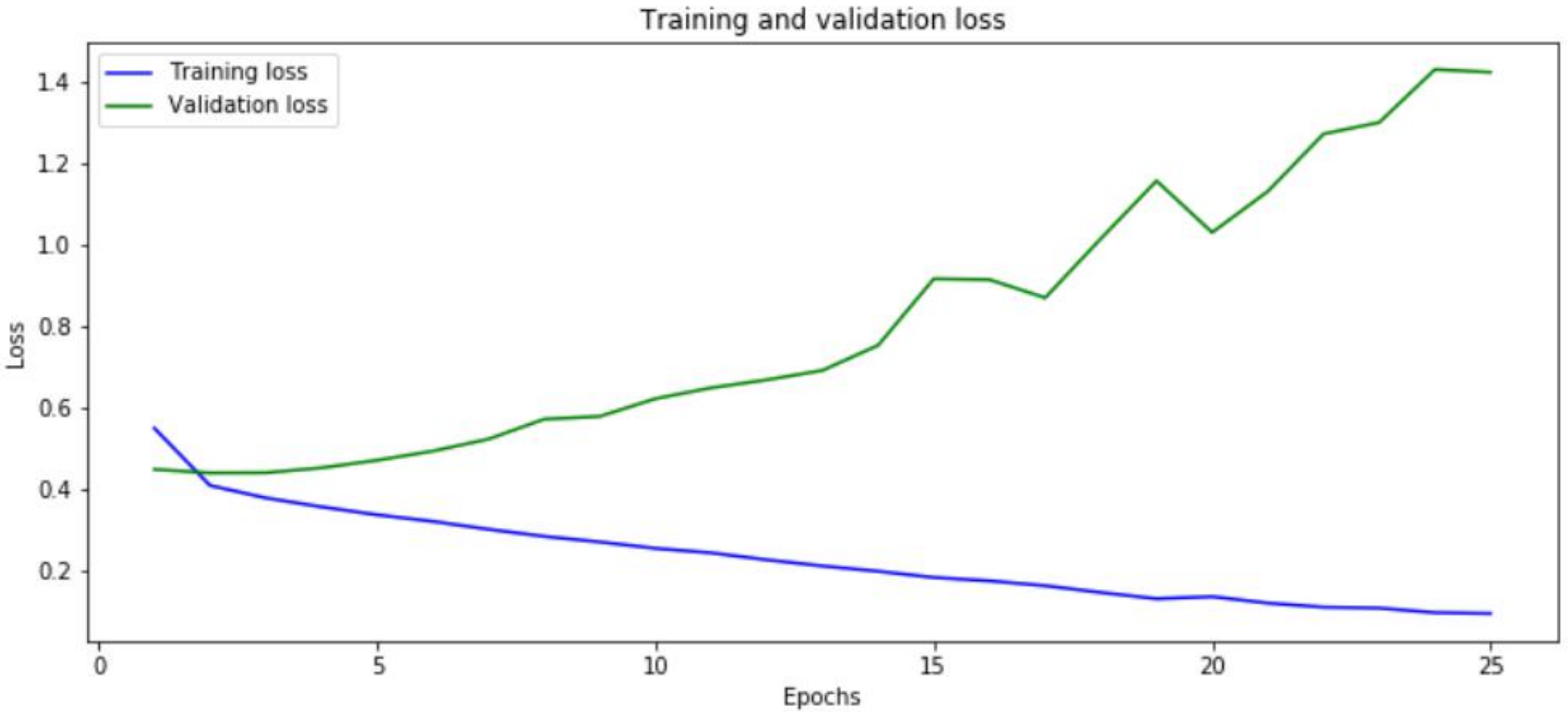

Training and validation accuracy
Accuracy
Validation accuracy
0.95
0.90
0.85
0.80
0.75
0.70
Accuracy
0
5
10
15
20
25
Epochs

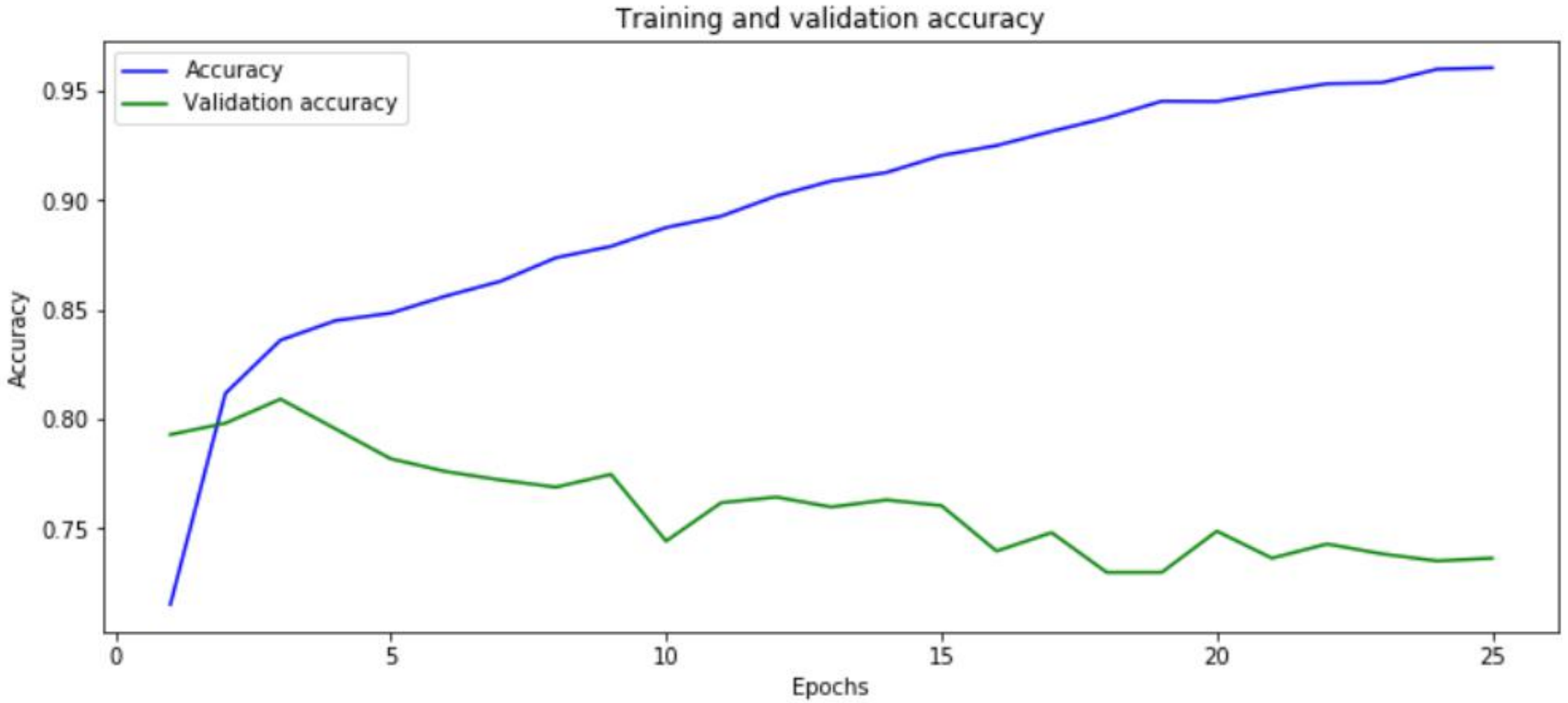
Modelo usando LSTM

```
model_2 = Sequential()
model_2.add(Embedding(max_words, embedding_dim, input_length = max_len))
model_2.add(Dropout(0.2))
model_2.add(LSTM(embedding_dim))
model_2.add(Dropout(0.2))
model_2.add(Dense(1, activation = 'sigmoid'))
model_2.summary()

model_2.compile(optimizer = 'adam',
                loss = 'binary_crossentropy',
                metrics  =['accuracy'])
history_2 = model_2.fit(X_train, Y_train,
                    epochs = 25,
                    batch_size = 32,
                    validation_split = 0.2)
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_2 (Embedding)      (None, 22, 100)           100000
_________________________________________________________________
dropout_1 (Dropout)          (None, 22, 100)           0
_________________________________________________________________
lstm_1 (LSTM)                (None, 100)               80400
_________________________________________________________________
dropout_2 (Dropout)          (None, 100)               0
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 101
=================================================================
Total params: 180,501
Trainable params: 180,501
Non-trainable params: 0
```

Desempeño del modelo




Training and validation accuracy
Accuracy
Validation accuracy
0.95
0.90
0.85
0.80
0.75
Accuracy
0
5
10
15
20
25
Epochs

Modelo usando CNN

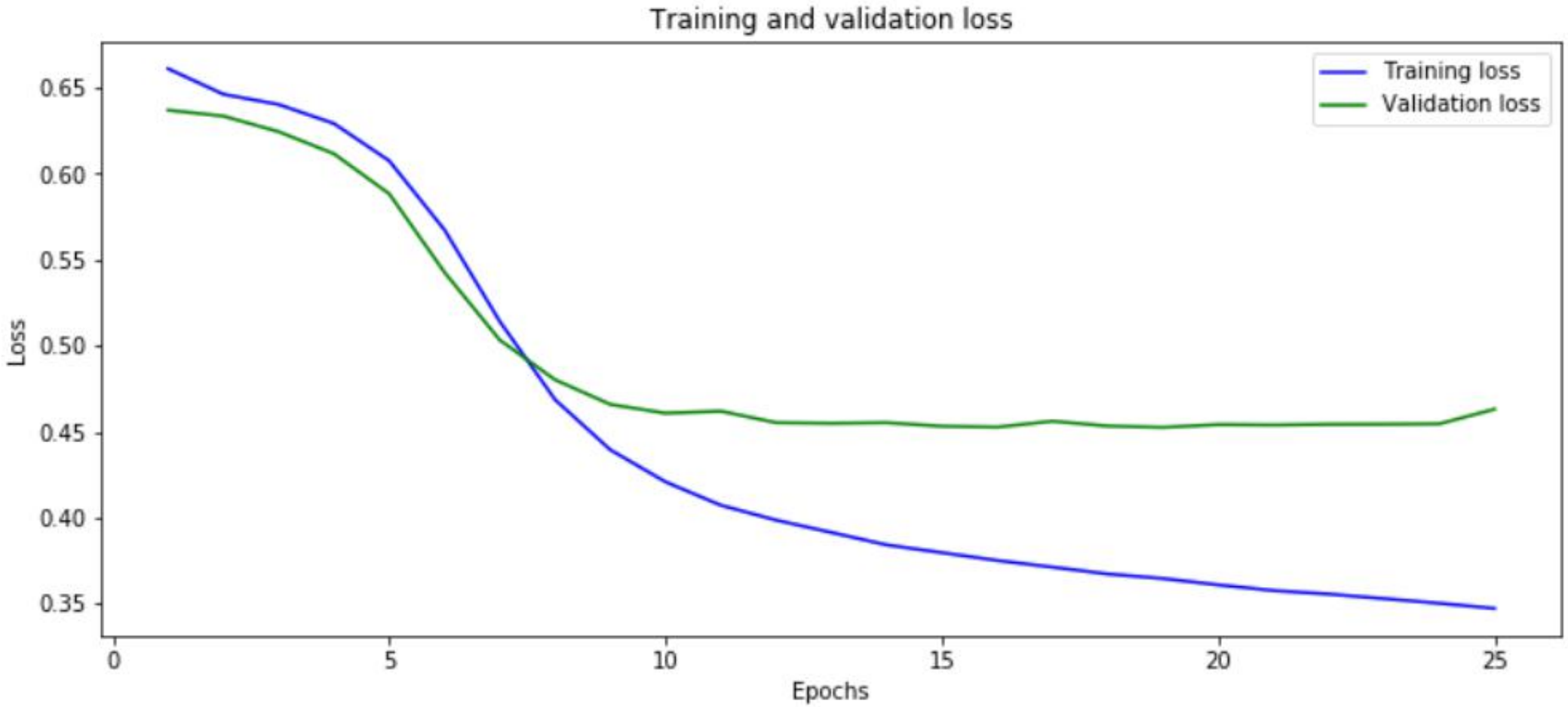
```
from keras.optimizers import RMSprop
from keras import layers
model_3 = Sequential()
model_3.add(Embedding(max_words, embedding_dim, input_length = max_len))
model_3.add(layers.Conv1D(64, 7, activation='relu', data_format = 'channels_first'))
model_2.add(Dropout(0.2))
model_3.add(layers.MaxPooling1D(5))
model_3.add(layers.Conv1D(64, 7, activation='relu', data_format = 'channels_first'))
model_2.add(Dropout(0.2))
model_3.add(layers.GlobalMaxPooling1D())
model_3.add(layers.Dense(units=64, activation='relu'))
model_3.add(layers.Dense(1, activation ='sigmoid'))
model_3.summary()

model_3.compile(optimizer = RMSprop(lr=1e-4),
                loss = 'binary_crossentropy',
                metrics  = ['accuracy'])
history_3 = model_3.fit(X_train, Y_train,
                    epochs = 25,
                    batch_size = 32,
                    validation_split = 0.2)
```

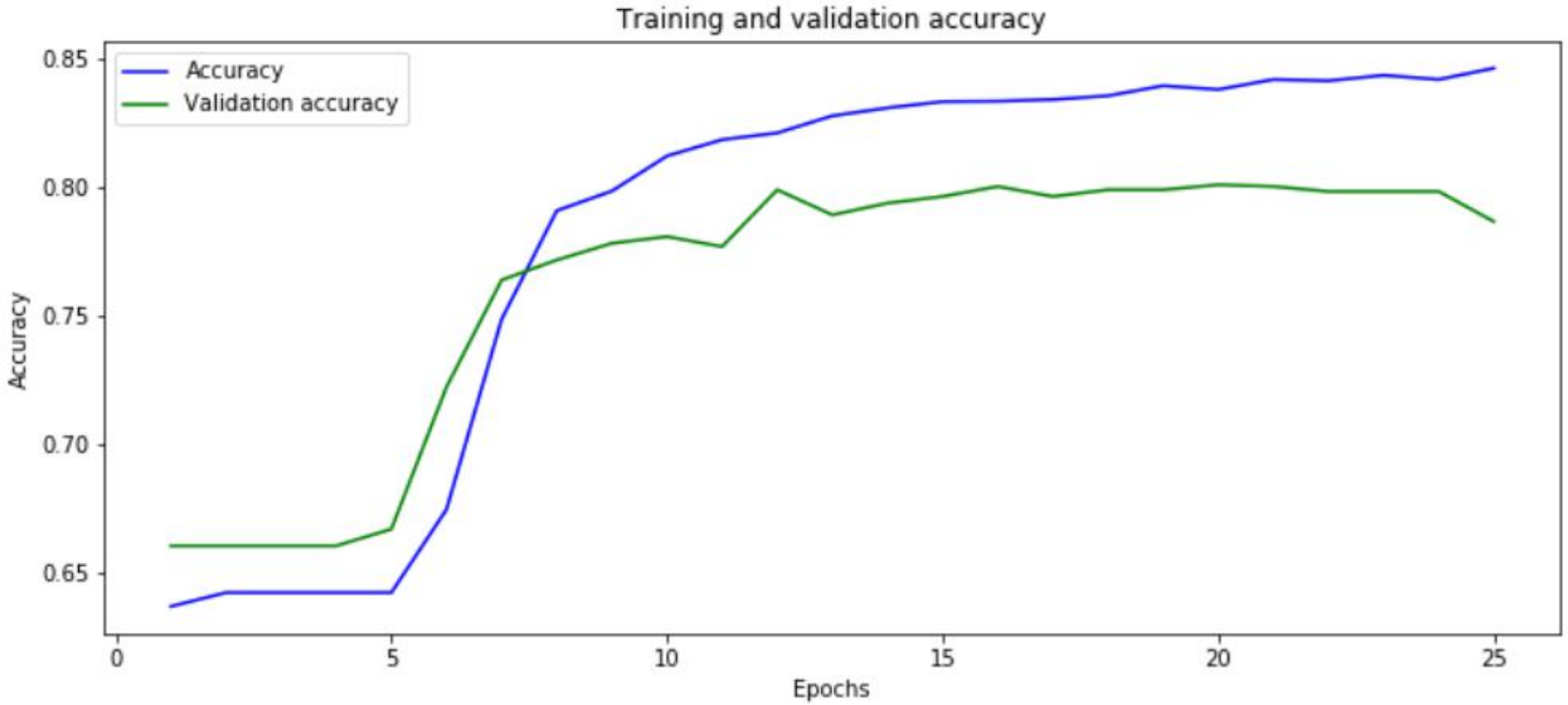
Modelo usando CNN

Layer (type)	Output Shape	Param #
embedding_3 (Embedding)	(None, 22, 100)	100000
conv1d_1 (Conv1D)	(None, 64, 94)	9920
max_pooling1d_1 (MaxPooling1	(None, 12, 94)	0
conv1d_2 (Conv1D)	(None, 64, 88)	5440
global_max_pooling1d_1 (Glob	(None, 88)	0
dense_4 (Dense)	(None, 64)	5696
dense_5 (Dense)	(None, 1)	65

Total params: 121,121
Trainable params: 121,121
Non-trainable params: 0

Desempeño del modelo




Training and validation accuracy
Accuracy
Validation accuracy
0.85
0.80
0.75
0.70
0.65
Accuracy
0
5
10
15
20
25
Epochs

05

Conclusiones

- El primer modelo (Feedforward Netwrok) tuvo una precisión máxima del 80.39% en validación pero el error en la función objetivo empezaba a aumentar
- LSTM tuvo una precisión maxima del 80.91% (ligeramente mayor al anterior)
- CNN tuvo una precisión máxima de 81.06% pero se puede observar que el error en la validación va decayendo conforme al aumento de épocas y la precisión va aumentando consistentemente con el proceso de entrenamiento.

Trabajo a futuro

- Explorar otras técnicas de representaciones de palabras (e.g. *profile based representations*)
- Implementación de modelos utilizando *word embeddings* pre-entrenados
- Explorar métodos de aumentación de datos (e.g. *reversing sequences* para redes recurrentes)
- Implementación de modelos que combinen LSTM con CNN

Referencias

[1] Francois Chollet. 2017. Deep Learning with Python (1st ed.). Manning Publications Co., Greenwich, CT, USA.

[2] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. Deep Learning. The MIT Press.

[3] H. J. Escalante, E. Villatoro, S. E. Garza, A. P. López, M. Montes-y-Gómez, L. Villaseñor-Pineda.Early Detection of Deception and Aggressiveness using Profile-Based Representations. Expert Systems with Applications, 2017