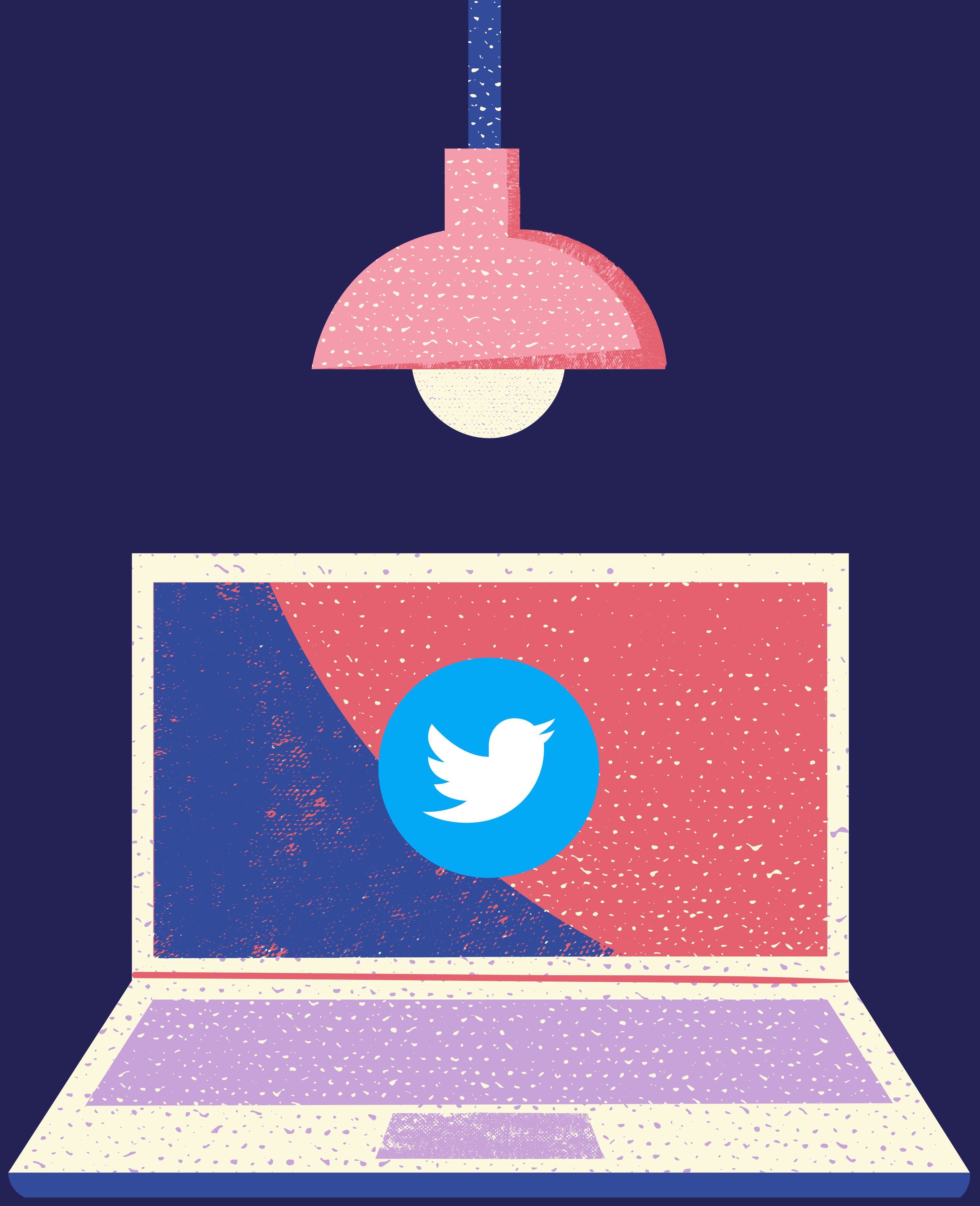


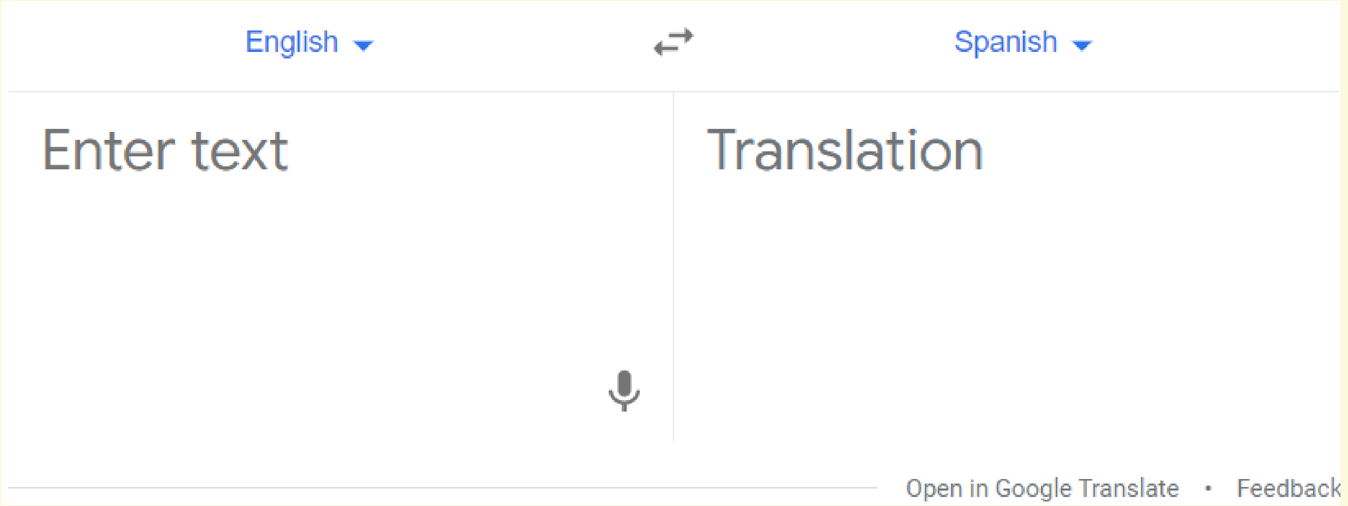
Tweet Sentiment Analysis: Natural Language Processing (NLP) using LTSM

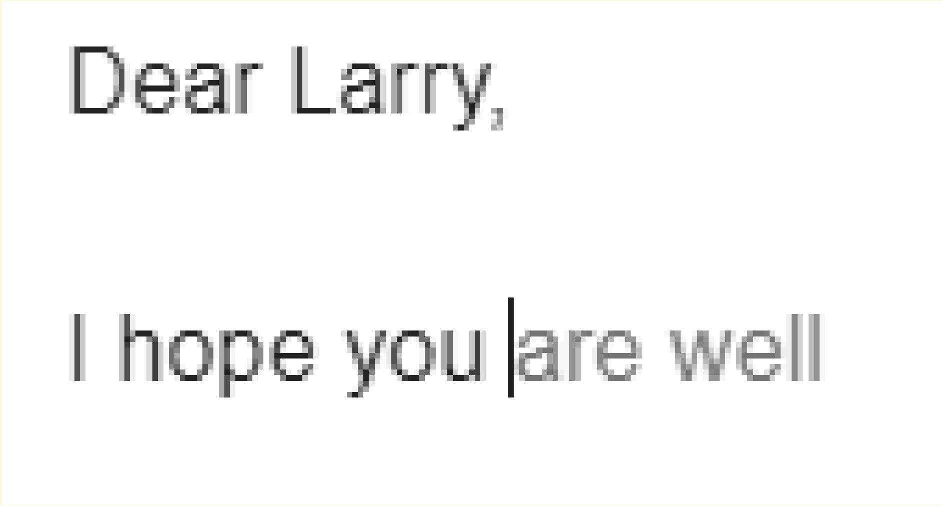
Sydney Davis, Data Science 101, Final Presentation

What is NLP + why is it used?

- Combination of disciplines: computer science and linguistics

- Prevalent in our lives





- Sentiment analysis

Step 1: Get data, explore, preprocess it

	sentiment	tweets
0	0	@switchfoot http://twitpic.com/2y1zl - Awww, t...
1	0	is upset that he can't update his Facebook by ...
2	0	@Kenichan I dived many times for the ball. Man...
3	0	my whole body feels itchy and like its on fire
4	0	@nationwideclass no, it's not behaving at all....
...	...	...
1599995	1	Just woke up. Having no school is the best fee...
1599996	1	TheWDB.com - Very cool to hear old Walt interv...
1599997	1	Are you ready for your MoJo Makeover? Ask me f...
1599998	1	Happy 38th Birthday to my boo of alll time!!! ...
1599999	1	happy #charitytuesday @theNSPCC @SparksCharity...

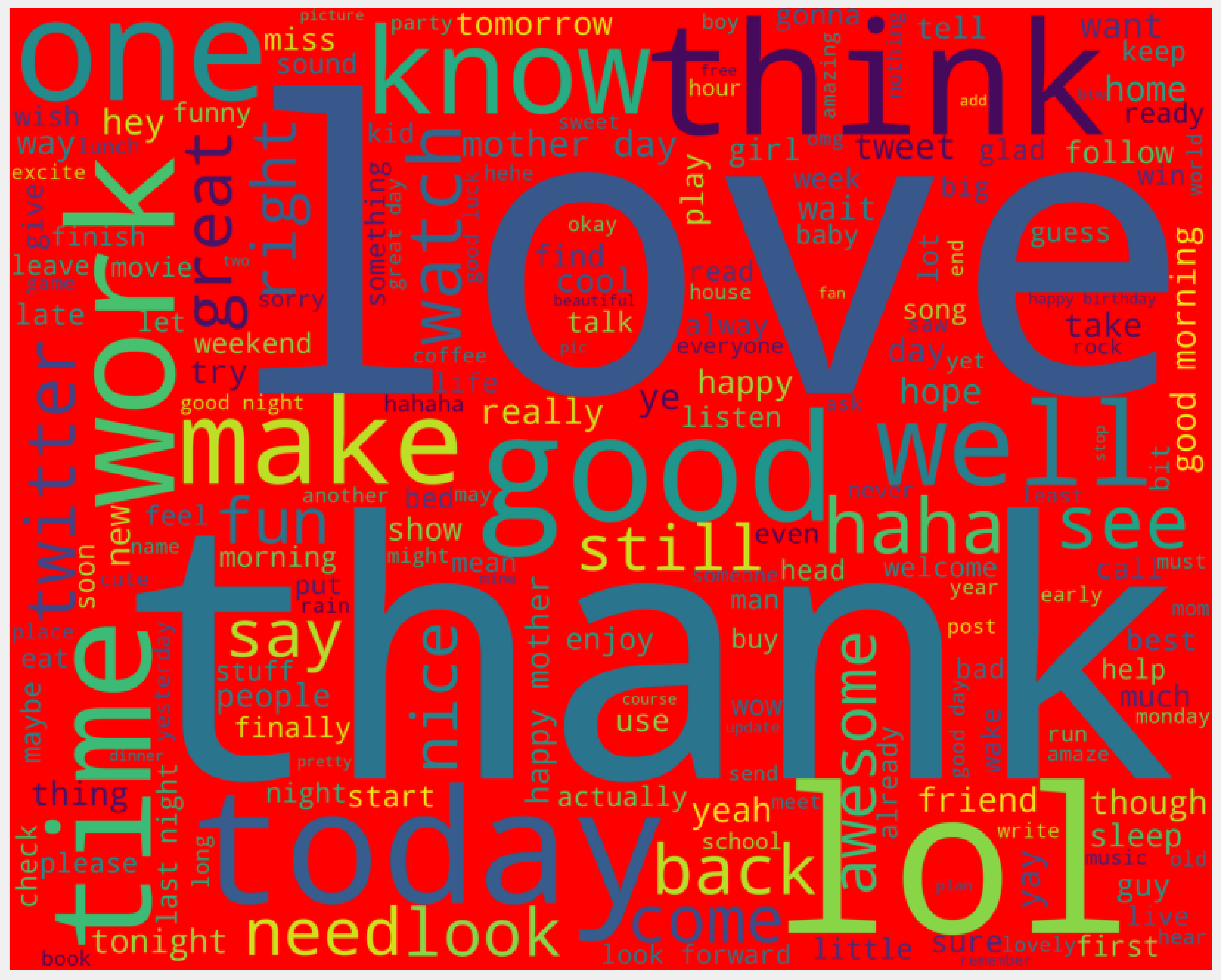
1600000 rows × 2 columns

```python
def cleanText(text):
    text = re.sub(r'@[A-Za-z0-9]+', '', text) #removing mentions
    text = re.sub(r'#', '', text) #removing hashtag symbol
    text = re.sub(r'RT[\s]+', '', text) #removing retweets
    text = re.sub(r'https?:\/\/\S+', '', text) #removing hyperlinks and whitespaces
    text = text.lower()
    return text

newData['tweets'] = newData['tweets'].apply(cleanText)
newData.head()
```

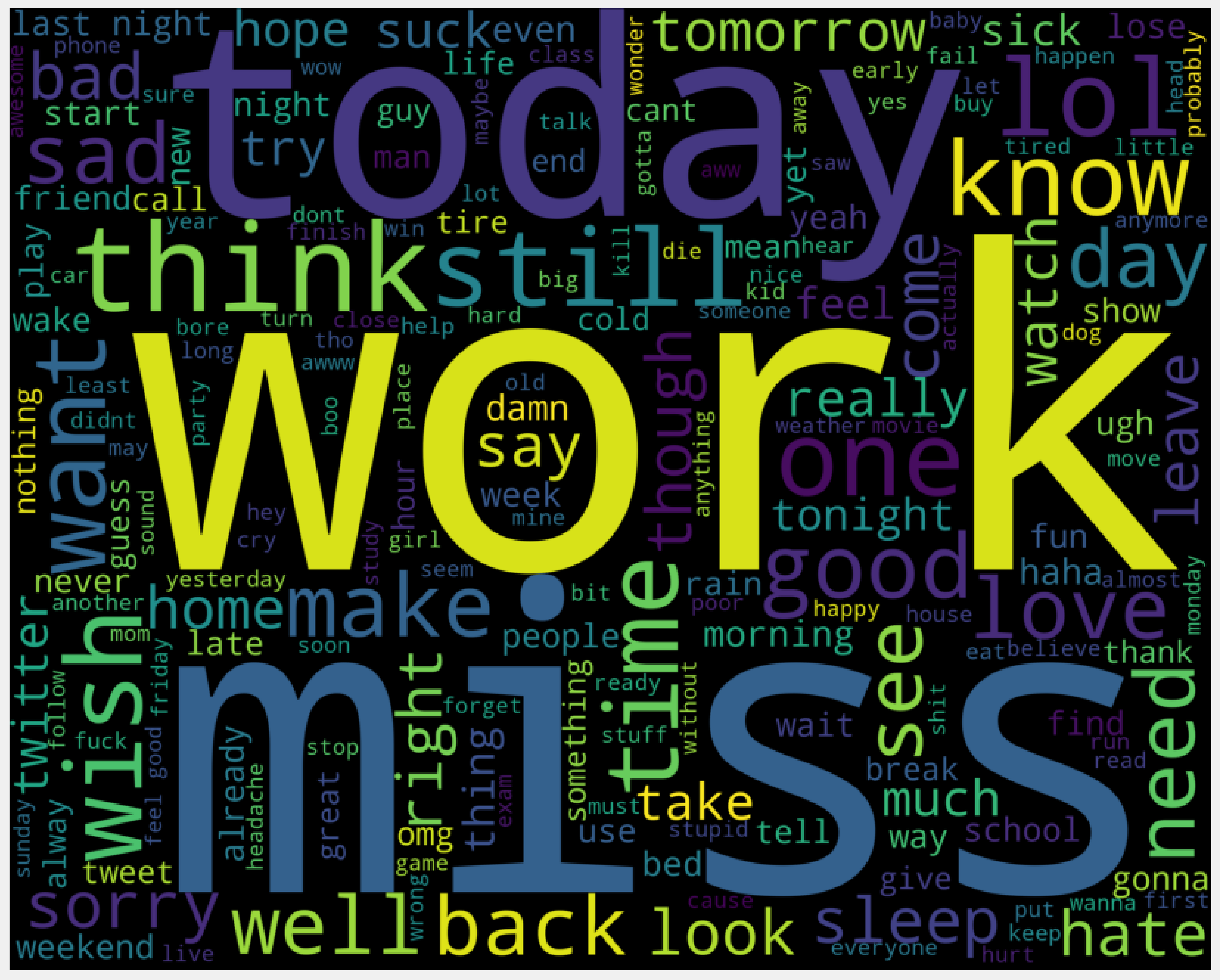
```python
def cleaned(token):
    if token == 'u':
        return 'you'
    if token == 'r':
        return 'are'
    if token == 'some1':
        return 'someone'
    if token == 'yrs':
        return 'years'
    if token == 'hrs':
        return 'hours'
```

NLTK

Positive words



Negative words



Step 2: decide on ML/algorithm, prepare further

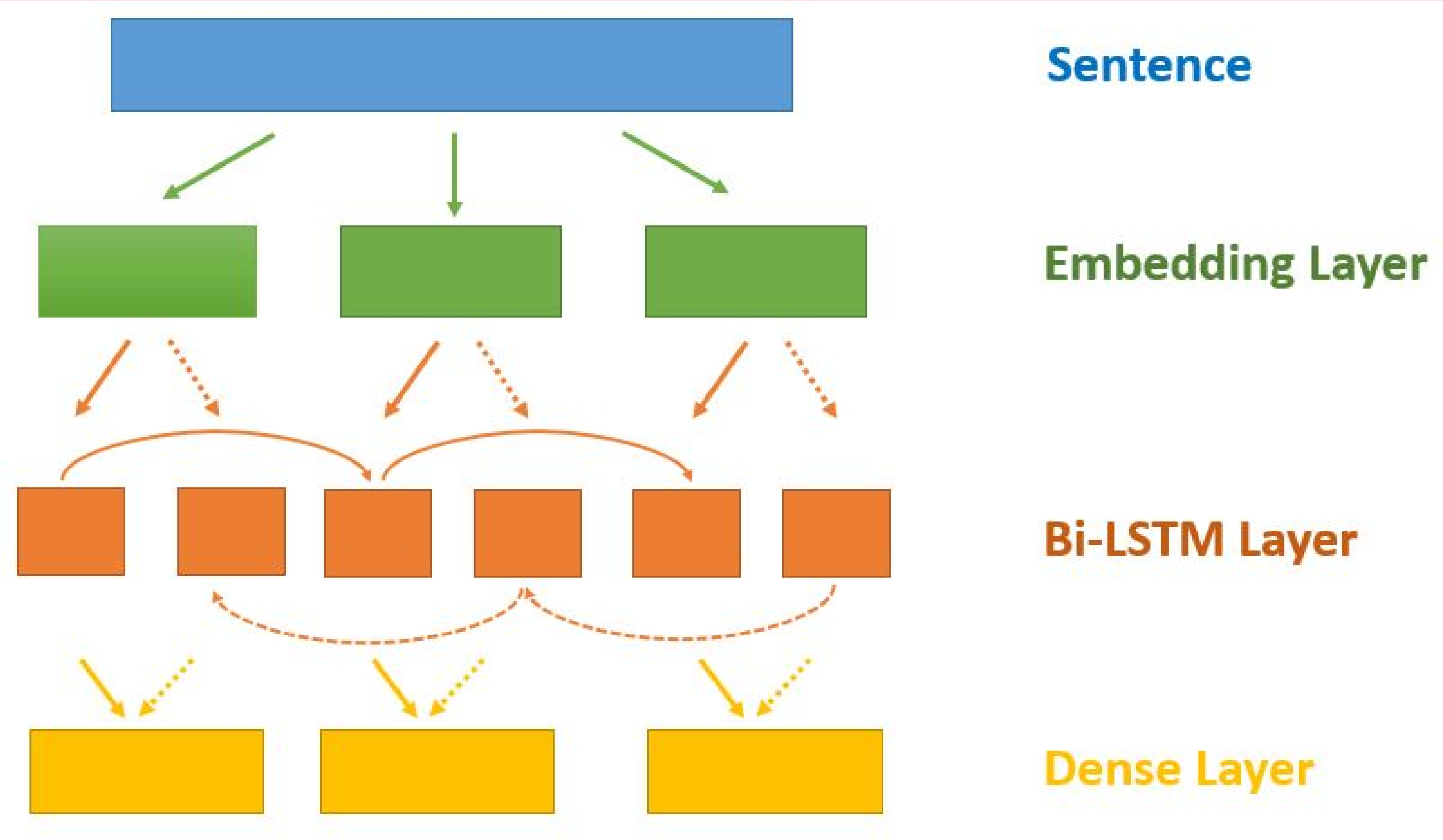
what is the best neural network for sentiment analysis

```python
word_to_index, index_to_word, word_to_vec_map = read_glove_vecs('glove.6B.50d.txt')
```

```python
word_to_index['hello']
```

```
176468
```

```
        0.       0.       0.       0.       0.       0.       0.       0.       0.
        0.       0.       0.       0.       0.       0.       0.]
 [357161. 368306.  46173. 372306. 160418. 239785. 179025. 329974.  58999.
  349437.       0.       0.       0.       0.       0.       0.       0.       0.
        0.       0.       0.       0.       0.       0.       0.]
 [330826. 302352.  97698. 184322. 251645. 132701. 302292. 151204. 286963.
  154049. 231458. 338210.       0.       0.       0.       0.       0.       0.
```

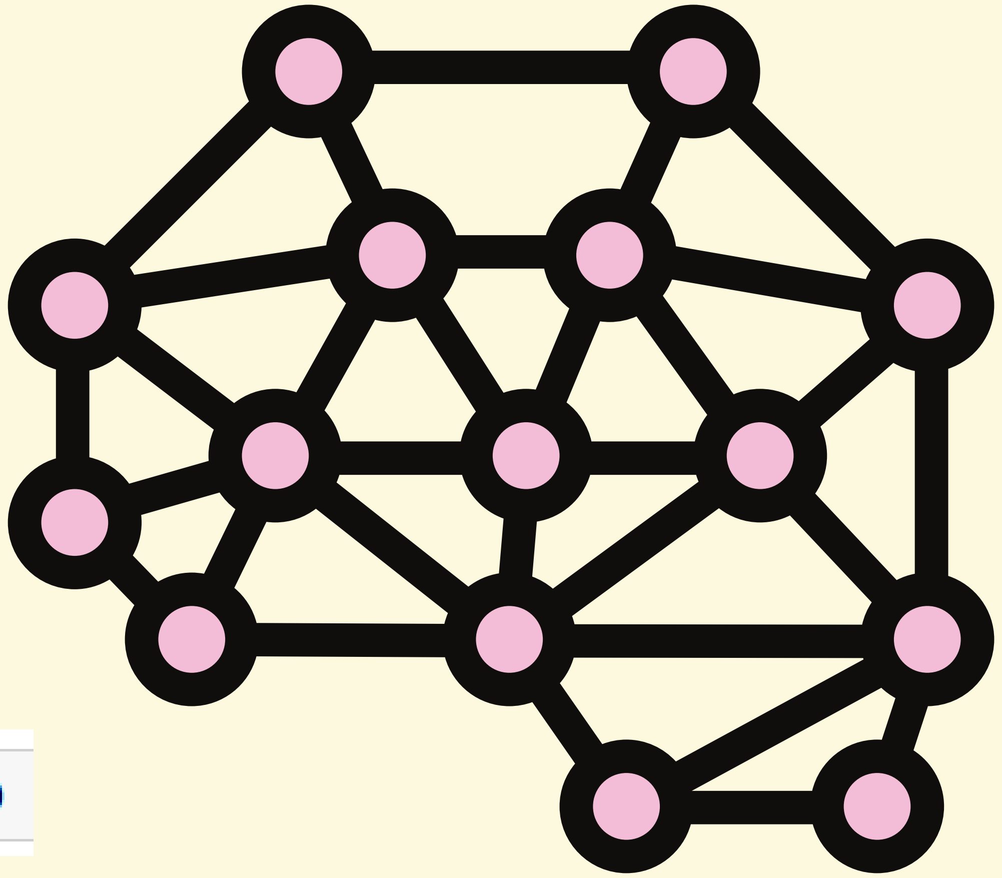



```python
# Here's my sequential model with the embedding layer, two bidirectional LSTMs, and a density layer at the end (0-1)

model = Sequential()

model.add(pretrained_embedding_layer(word_to_vec_map, word_to_index, max_len))
model.add(Bidirectional(LSTM(units=128, return_sequences=True)))
model.add(Bidirectional(LSTM(units=128, return_sequences=False)))
model.add(Dense(units=1, activation='sigmoid'))

model.summary()
```

Step 3: hyperparameters + running it

Adam optimizer

```
# Use the default adam optimizer

model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```



Trained in batches

```
model.fit(X_train, Y_train, validation_data=(X_test, Y_test), epochs = 20, batch_size = 128, shuffle=True)
```

Epoch 1
training accuracy: 0.7133
val_accuracy: 0.7426

overfitting? →

Epoch 20
training accuracy: 0.9564
val_accuracy: 0.7301

Step 4: tweak it, run it back

```python
from collections import Counter
Counter(unks).most_common(50)
```

```
[("i'm", 32149),
 ("can't", 11370),
 ("i'l", 6284),
 ("that's", 5478),
 ("i've", 5085),
 ("he's", 1976),
 ("mother's", 1879),
 ("i'd", 1855),
 ('hahaha', 1723),
 ("we're", 1578),
 ("there's", 1425),
 ("what's", 1356),
 ("they're", 1179),
```

Epoch 1:
training accuracy:
0.7379
val_accuracy:
0.7714

→

```python
from collections import Counter
Counter(unks).most_common(50)
```

```
[(':/', 672),
 ('(:', 452),
 ('shouldnt', 429),
 ('. .', 426),
 ('asot', 363),
 (":'(", 327),
 ('folowfriday', 314),
 ('tweps', 312),
 (';-)', 307),
 ('->', 288),
 ('iï', 278),
```

Epoch 20:
training accuracy:
0.9693
val_accuracy:
0.7823

In summary: looking at the unknown words in our twitter dataset and cleaning the data accordingly took us from a 73% validation accuracy to 78%.

"Your model is only as good as your data." - unknown

Step 5: evaluate incorrect data for patterns and adjust strategy

Expected sentiment: 1 (positive)
Input: "thinking about vacationing in hawaii ... alone"

Expected sentiment: 0 (negative)
Input: "i can understand ... sorry for making you talk about it cherish the positive memories"

Expected sentiment: 1 (positive)
Input: "no need math"

To conclude:

- Endless number of ways to improve data and algorithim
- As NLP algorithms improve, efficiency does too
- Data cleaning/mining/exploration is one of the biggest keys to NLP