In [15]:
```
import pandas as pd
import tensorflow as tf
from sklearn.model_selection import train_test_split
from tensorflow.keras import datasets, layers, models, preprocessing
import numpy as np
np.set_printoptions(precision=3, suppress=True)
df = pd.read_csv('dnd_monsters.csv')
max_features = 10000
maxlen = 20
print(df.head()) # graph of data classes
df_parts = df.copy()
df_parts = np.array(df_parts)

# splits into test and train
train = df.sample(frac=0.8,random_state=1234) #divides up the data into train
test = df.drop(train.index) # test data
X = df.name
y = df.type
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, train_size=0.


print('The dataset consists of information about the different available monsters from
```

```
              name                                                url   cr  \
0        aarakocra  https://www.aidedd.org/dnd/monstres.php?vo=aar...  1/4
1          abjurer                                                NaN    9
2          aboleth  https://www.aidedd.org/dnd/monstres.php?vo=abo...   10
3  abominable-yeti                                                NaN    9
4         acererak                                                NaN   23

                   type    size  ac   hp speed          align  legendary  \
0  humanoid (aarakocra)  Medium  12   13   fly   neutral good        NaN
1   humanoid (any race)  Medium  12   84   NaN  any alignment        NaN
2           aberration   Large  17  135  swim    lawful evil  Legendary
3          monstrosity    Huge  15  137   NaN   chaotic evil        NaN
4               undead  Medium  21  285   NaN   neutral evil        NaN

                            source   str   dex   con   int   wis   cha
0                Monster Manual (BR)  10.0  14.0  10.0  11.0  12.0  11.0
1           Volo's Guide to Monsters   NaN   NaN   NaN   NaN   NaN   NaN
2               Monster Manual (SRD)  21.0   9.0  15.0  18.0  15.0  18.0
3                     Monster Manual   NaN   NaN   NaN   NaN   NaN   NaN
4  Adventures (Tomb of Annihilation)   NaN   NaN   NaN   NaN   NaN   NaN
The dataset consists of information about the different available monsters from DnD.
The model should be able to predict the kind of Dnd monster something is and related
information to it
```

In [10]:
```
import pandas as pd
import tensorflow as tf
from sklearn.model_selection import train_test_split
from tensorflow.keras import datasets, layers, models, preprocessing
from sklearn.preprocessing import LabelEncoder
import numpy as np
np.set_printoptions(precision=3, suppress=True)
df = pd.read_csv('dnd_monsters.csv')
max_features = 10000
```

```
maxlen = 20
print(df.head()) # graph of data classes
df_parts = df.copy()
df_parts = np.array(df_parts)

# splits into test and train
train = df.sample(frac=0.8,random_state=1234) #divides up the data into train
test = df.drop(train.index) # test data
X = df.name
y = df.type
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, train_size=0.
tokenizer = tf.keras.preprocessing.text.Tokenizer()
tokenizer.fit_on_texts(train)
A_train = tokenizer.texts_to_sequences(train)
A_train = tf.keras.preprocessing.sequence.pad_sequences(A_train, maxlen=20)

# sequential model
model = models.Sequential()
model.add(layers.Embedding(max_features, 32))
model.add(layers.LSTM(32))
model.add(layers.Dense(1, activation='sigmoid'))
model.summary()

#evaluate on the test data
model.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['accuracy'])
B_test = tokenizer.texts_to_sequences(test)
B_test = tf.keras.preprocessing.sequence.pad_sequences(B_test, maxlen=20)
pred = model.predict(B_test)
pred = [1.0 if p>= 0.5 else 0.0 for p in pred]
print(pred)
```

```
             name                                                url   cr  \
0        aarakocra  https://www.aidedd.org/dnd/monstres.php?vo=aar...  1/4
1          abjurer                                                NaN    9
2          aboleth  https://www.aidedd.org/dnd/monstres.php?vo=abo...   10
3  abominable-yeti                                                NaN    9
4         acererak                                                NaN   23

                   type    size  ac   hp speed          align  legendary  \
0  humanoid (aarakocra)  Medium  12   13   fly   neutral good        NaN
1   humanoid (any race)  Medium  12   84   NaN  any alignment        NaN
2           aberration   Large  17  135  swim    lawful evil  Legendary
3          monstrosity    Huge  15  137   NaN   chaotic evil        NaN
4                undead  Medium  21  285   NaN   neutral evil        NaN

                             source   str   dex   con   int   wis   cha
0                Monster Manual (BR)  10.0  14.0  10.0  11.0  12.0  11.0
1           Volo's Guide to Monsters   NaN   NaN   NaN   NaN   NaN   NaN
2               Monster Manual (SRD)  21.0   9.0  15.0  18.0  15.0  18.0
3                     Monster Manual   NaN   NaN   NaN   NaN   NaN   NaN
4  Adventures (Tomb of Annihilation)   NaN   NaN   NaN   NaN   NaN   NaN
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_8 (Embedding)     (None, None, 32)          320000

 lstm_2 (LSTM)               (None, 32)                8320

 dense_2 (Dense)             (None, 1)                 33

=================================================================
Total params: 328,353
Trainable params: 328,353
Non-trainable params: 0
_________________________________________________________________
1/1 [==============================] - 1s 712ms/step
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0]
```

In [30]:

```python
import pandas as pd
import tensorflow as tf
from sklearn.model_selection import train_test_split
from tensorflow.keras import datasets, layers, models, preprocessing
from sklearn.preprocessing import LabelEncoder
import numpy as np
np.set_printoptions(precision=3, suppress=True)
df = pd.read_csv('dnd_monsters.csv')
max_features = 10000
maxlen = 20
print(df.head()) # graph of data classes
df_parts = df.copy()
df_parts = np.array(df_parts)

# splits into test and train
train = df.sample(frac=0.8,random_state=1234) #divides up the data into train
test = df.drop(train.index) # test data
X = df.name
y = df.type
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, train_size=0.
tokenizer = tf.keras.preprocessing.text.Tokenizer()
tokenizer.fit_on_texts(train)
A_train = tokenizer.texts_to_sequences(train)
A_train = tf.keras.preprocessing.sequence.pad_sequences(A_train, maxlen=20)

#rnn
model = models.Sequential()
model.add(layers.Embedding(max_features, 32))
model.add(layers.SimpleRNN(32))
model.add(layers.Dense(1, activation='sigmoid'))
model.summary()

#evaluate on the test data
model.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['accuracy'])
B_test = tokenizer.texts_to_sequences(test)
B_test = tf.keras.preprocessing.sequence.pad_sequences(B_test, maxlen=20)
pred = model.predict(B_test)
pred = [1.0 if p>= 0.5 else 0.0 for p in pred]
print(pred)
```

```
              name                                                url   cr  \
0        aarakocra  https://www.aidedd.org/dnd/monstres.php?vo=aar...  1/4
1          abjurer                                                NaN    9
2          aboleth  https://www.aidedd.org/dnd/monstres.php?vo=abo...   10
3  abominable-yeti                                                NaN    9
4         acererak                                                NaN   23

                   type    size  ac   hp speed          align  legendary  \
0  humanoid (aarakocra)  Medium  12   13   fly   neutral good        NaN
1   humanoid (any race)  Medium  12   84   NaN  any alignment        NaN
2           aberration   Large  17  135  swim    lawful evil  Legendary
3          monstrosity    Huge  15  137   NaN   chaotic evil        NaN
4                undead  Medium  21  285   NaN   neutral evil        NaN

                             source   str   dex   con   int   wis   cha
0                Monster Manual (BR)  10.0  14.0  10.0  11.0  12.0  11.0
1           Volo's Guide to Monsters   NaN   NaN   NaN   NaN   NaN   NaN
2               Monster Manual (SRD)  21.0   9.0  15.0  18.0  15.0  18.0
3                     Monster Manual   NaN   NaN   NaN   NaN   NaN   NaN
4  Adventures (Tomb of Annihilation)   NaN   NaN   NaN   NaN   NaN   NaN
Model: "sequential_22"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_22 (Embedding)    (None, None, 32)          320000

 simple_rnn_1 (SimpleRNN)    (None, 32)                2080

 dense_23 (Dense)            (None, 1)                 33

=================================================================
Total params: 322,113
Trainable params: 322,113
Non-trainable params: 0
_________________________________________________________________
1/1 [==============================] - 0s 292ms/step
[1.0, 1.0, 0.0, 1.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0]
```

In [12]:

```
#embedding approaches
import numpy as np
import os
import pathlib
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.layers.experimental.preprocessing import TextVectorization
from tensorflow.keras import layers
from sklearn.metrics import classification_report

samples = []
labels = []
index = 0
EMBEDDING_DIM = 128
MAX_SEQUENCE_LENGTH = 200

with open('dnd_monsters.csv', 'r') as f:
    content = f.read()
```

```
    lines = content.split("\n")
    lines = lines[10:]
    content = "\n".join(lines)
    samples.append(content)
    labels.append(index)
    index += 1

split_1_portion = 0.2  # train
split_2_portion = 0.8  # test
split_1 = int(split_1_portion * len(samples))
split_2 = int(split_2_portion * len(samples))
val_samples = samples[:split_1]
val_labels = labels[:split_1]
train_samples = samples[split_1:split_2]
train_labels = labels[split_1:split_2]
test_samples = samples[split_2:]
test_labels = labels[split_2:]
vectorizer = TextVectorization(max_tokens=20000, output_sequence_length=200)
text_ds = tf.data.Dataset.from_tensor_slices(train_samples).batch(128)
vectorizer.adapt(text_ds)
voc = vectorizer.get_vocabulary()
word_index = dict(zip(voc, range(len(voc))))
embedding_layer = layers.Embedding(len(word_index) + 1, EMBEDDING_DIM, input_length=MA
int_sequences_input = keras.Input(shape=(None,), dtype="int64")
embedded_sequences = embedding_layer(int_sequences_input)
test_x = vectorizer(np.array([[s] for s in test_samples])).numpy()
preds = model.predict(test_x)
pred_labels = [np.argmax(p) for p in preds]
print(classification_report(test_labels, pred_labels))
```

```
1/1 [==============================] - 0s 179ms/step
[0.0, 1.0, 0.0, 1.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0]
```

We see that various approaches have different speeds in terms of getting everything done. In terms of the sequential model, he preparation that is needed to make a sequential model adds on time to the performace speed of this approach. It starts off quickly but ends up slowing down after a while. As for approaches such as RNN, we can see that it's performance is effected by the additonional work that is needed to prepare it. The performance is somewhat better than it was in the previous approach but it still needs some work in terms of accuracy. As for the various embedding approaches, we can see that in terms of of performance, we can see that it's effected by the work that needed to prepare it for evaluation. In addition, it has a lower accuracy than the other approaches but is faster than the other approaches.