In [24]:
```
import numpy
import sys
import nltk
nltk.download('stopwords')
from nltk.tokenize import RegexpTokenizer
from nltk.corpus import stopwords
from keras.models import Sequential
from keras.layers import Dense, Dropout, LSTM
from keras.utils import np_utils
from keras.callbacks import ModelCheckpoint
```
```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [13]:
```
file = open("frankestein.2.txt").read()
```

In [14]:
```
def tokenize_words(input):
    input = input.lower()
    tokenizer = RegexpTokenizer(r'\w+')
    tokens = tokenizer.tokenize(input)
    filtered = filter(lambda token: token not in stopwords.words('english'), tokens)
    return " ".join(filtered)

processed_inputs = tokenize_words(file)
```

In [15]:
```
chars = sorted(list(set(processed_inputs)))
char_to_num = dict((c, i) for i, c in enumerate(chars))
```

In [16]:
```
input_len = len(processed_inputs)
vocab_len = len(chars)
print ("Total number of characters:", input_len)
print ("Total vocab:", vocab_len)
```
```
Total number of characters: 269995
Total vocab: 43
```

In [17]:
```
seq_length = 100
x_data = []
y_data = []
```

In [18]:
```
for i in range(0, input_len - seq_length, 1):
    in_seq = processed_inputs[i:i + seq_length]
    out_seq = processed_inputs[i + seq_length]
    x_data.append([char_to_num[char] for char in in_seq])
    y_data.append(char_to_num[out_seq])

n_patterns = len(x_data)
print ("Total Patterns:", n_patterns)
```
```
Total Patterns: 269895
```

In [25]:
```
X = numpy.reshape(x_data, (n_patterns, seq_length, 1))
X = X/float(vocab_len)
```

In [26]:
```
y = np_utils.to_categorical(y_data)
```

In [28]:
```
model = Sequential()
model.add(LSTM(256, input_shape=(X.shape[1], X.shape[2]), return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(256, return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(128))
model.add(Dropout(0.2))
model.add(Dense(y.shape[1], activation='softmax'))
```

In [29]:
```
model.compile(loss='categorical_crossentropy', optimizer='adam')
```
```
W0522 14:22:30.735936 140386517747520 deprecation_wrapper.py:119] From /mnt/disks/user/anacon
da3/lib/python3.7/site-packages/keras/optimizers.py:790: The name tf.train.Optimizer is depre
cated. Please use tf.compat.v1.train.Optimizer instead.

W0522 14:22:30.762437 140386517747520 deprecation_wrapper.py:119] From /mnt/disks/user/anacon
da3/lib/python3.7/site-packages/keras/backend/tensorflow_backend.py:3295: The name tf.log is
deprecated. Please use tf.math.log instead.
```

In [30]:
```
filepath = "model_weights_saved.hdf5"
checkpoint = ModelCheckpoint(filepath, monitor='loss', verbose=1, save_best_only=True, mode=
'min')
desired_callbacks = [checkpoint]
```

In [31]:
```
model.fit(X, y, epochs=4, batch_size=256, callbacks=desired_callbacks)
```
```
W0522 14:27:16.484817 140386517747520 deprecation.py:323] From /mnt/disks/user/anaconda3/lib/
python3.7/site-packages/tensorflow/python/ops/math_grad.py:1250: add_dispatch_support.<locals
>.wrapper (from tensorflow.python.ops.array_ops) is deprecated and will be removed in a futur
e version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
```
```
Epoch 1/4
269895/269895 [==============================] - 1093s 4ms/step - loss: 2.9398

Epoch 00001: loss improved from inf to 2.93980, saving model to model_weights_saved.hdf5
Epoch 2/4
269895/269895 [==============================] - 1060s 4ms/step - loss: 2.8282

Epoch 00002: loss improved from 2.93980 to 2.82825, saving model to model_weights_saved.hdf5
Epoch 3/4
269895/269895 [==============================] - 1054s 4ms/step - loss: 2.6424

Epoch 00003: loss improved from 2.82825 to 2.64240, saving model to model_weights_saved.hdf5
Epoch 4/4
269895/269895 [==============================] - 1012s 4ms/step - loss: 2.5462

Epoch 00004: loss improved from 2.64240 to 2.54623, saving model to model_weights_saved.hdf5
```

Out[31]:
```
<keras.callbacks.History at 0x7fadc7305198>
```

In [32]:
```
filename = "model_weights_saved.hdf5"
model.load_weights(filename)
model.compile(loss='categorical_crossentropy', optimizer='adam')
```

In [33]:
```
num_to_char = dict((i, c) for i, c in enumerate(chars))
```

In [34]:
```
start = numpy.random.randint(0, len(x_data) - 1)
pattern = x_data[start]
print("Random Seed:")
print("\"", ''.join([num_to_char[value] for value in pattern]), "\"")
```
```
Random Seed:
" stroyed unacquainted obvious laws electricity occasion man great research natural philosoph
y us exci "
```

In [35]:
```
for i in range(1000):
    x = numpy.reshape(pattern, (1, len(pattern), 1))
    x = x / float(vocab_len)
    prediction = model.predict(x, verbose=0)
    index = numpy.argmax(prediction)
    result = num_to_char[index]
    seq_in = [num_to_char[value] for value in pattern]

    sys.stdout.write(result)

    pattern.append(index)
    pattern = pattern[1:len(pattern)]
```
```
e seare seare seare seare seare seare seare seare seare seare seare seare seare seare seare s
eare seare seare seare seare seare seare seare seare seare seare seare seare seare seare sear
e seare seare seare seare seare seare seare seare seare seare seare seare seare seare seare s
eare seare seare seare seare seare seare seare seare seare seare seare seare seare seare sear
e seare seare seare seare seare seare seare seare seare seare seare seare seare seare seare s
eare seare seare seare seare seare seare seare seare seare seare seare seare seare seare sear
e seare seare seare seare seare seare seare seare seare seare seare seare seare seare seare s
eare seare seare seare seare seare seare seare seare seare seare seare seare seare seare sear
e seare seare seare seare seare seare seare seare seare seare seare seare seare seare seare s
```