ASSIGNMENT - 13

Process of designing a **Natural Language Processing (NLP)** model for sarcasm detection using an **Artificial Neural Network (ANN).**

1. Data Collection:
   Collect a dataset of headlines along with their labels indicating whether they are sarcastic or not. The dataset can be obtained from sources like the one you mentioned or other publicly available datasets.

Data Preprocessing:
Preprocess the headlines to make them suitable for model training. This includes lowercasing, removing special characters, and tokenization.

```
import re
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

cleaned_headlines = [re.sub(r'[^\w\s]', '', headline.lower()) for headline in headlines]

tokenizer = Tokenizer(oov_token="<OOV>")
tokenizer.fit_on_texts(cleaned_headlines)
word_index = tokenizer.word_index
sequences = tokenizer.texts_to_sequences(cleaned_headlines)
padded_sequences = pad_sequences(sequences, padding='post')
```

2: Model Architecture
Design an ANN model for sarcasm detection.

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense, Dropout
vocab_size = len(word_index) + 1
embedding_dim = 100
model = Sequential([
    Embedding(vocab_size, embedding_dim, input_length=padded_sequences.shape[1]),
    LSTM(64, return_sequences=True),
    LSTM(64),
    Dense(64, activation='relu'),
    Dropout(0.5),
    Dense(1, activation='sigmoid')
])
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

3: Model Training

Split your dataset into training and testing sets and train your model.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(padded_sequences, labels, test_size=0.2,
random_state=42)
model.fit(X_train, y_train, epochs=10, batch_size=32, validation_data=(X_test, y_test))
```

4: Model Evaluation

Evaluate your model's performance on the test set.

```
loss, accuracy = model.evaluate(X_test, y_test)
print("Test Loss:", loss)
print("Test Accuracy:", accuracy)
```

5: Model Deployment and Improvements

We can consider improving your model by experimenting with different architectures, embeddings (like Word2Vec, GloVe), regularization techniques, and hyper parameter tuning.

The field of NLP is constantly evolving, and there are advanced techniques and pre-trained models available that can significantly boost the performance of sarcasm detection models