```
import nltk
from nltk.tokenize import word_tokenize
from difflib import SequenceMatcher
from nltk.corpus import stopwords
from sklearn.metrics import precision_score, recall_score, f1_score
import numpy as np
import editdistance


nltk.download('punkt')
nltk.download('stopwords')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
True
```

```
def preprocess_text(text):
    text = text.lower()  # Convert to lowercase
    tokens = word_tokenize(text)  # Tokenize
    stop_words = set(stopwords.words('english'))  # Remove stopwords
    filtered_tokens = [word for word in tokens if word.isalnum() and word not in stop_words]
    return filtered_tokens


def calculate_similarity(text1, text2):
    matcher = SequenceMatcher(None, text1, text2)
    return matcher.ratio()


# Function to calculate word-level accuracy using Levenshtein distance
def calculate_word_accuracy(reference, candidate):
    # Tokenize and preprocess the reference and candidate texts
    reference_tokens = preprocess_text(reference)
    candidate_tokens = preprocess_text(candidate)
    distance = editdistance.eval(reference_tokens, candidate_tokens)
    max_len = max(len(reference_tokens), len(candidate_tokens))

    # Accuracy score based on edit distance
    accuracy = 1 - (distance / max_len) if max_len > 0 else 0
    return accuracy


dataset = [
    {
        "script": "Hello, this is John from XYZ Insurance. I am calling to explain
        "transcribed_text": "Hello, this is John from XYZ Insurance. I am calling t
    },
    {
        "script": "Our health insurance plan offers comprehensive coverage includir
```

```
            "transcribed_text": "Our health insurance plan covers hospital stays, emerg
        },
        {
            "script": "Thank you for choosing ABC Telecom. Our new unlimited data plan
            "transcribed_text": "Thank you for selecting ABC Telecom. The unlimited dat
        }
    ]


    for i, data in enumerate(dataset):
        script = data["script"]
        transcribed_text = data["transcribed_text"]

        # Preprocess texts
        script_tokens = preprocess_text(script)
        transcribed_tokens = preprocess_text(transcribed_text)

        # Calculate similarity and accuracy
        similarity = calculate_similarity(script, transcribed_text)
        accuracy = calculate_word_accuracy(script, transcribed_text)

        # Generate quality report
        report = {
            'Total Script Words': len(script_tokens),
            'Transcribed Words': len(transcribed_tokens),
            'Similarity Score': round(similarity, 3),
            'Accuracy Score (Levenshtein-based)': round(accuracy, 3),
            'Deviation (%)': round(100 - (accuracy * 100), 2),
        }

        # Print the quality report
        print(f"\nQuality Report for Script {i + 1}:")
        for key, value in report.items():
            print(f"{key}: {value}")
```

```
Quality Report for Script 1:
Total Script Words: 22
Transcribed Words: 19
Similarity Score: 0.927
Accuracy Score (Levenshtein-based): 0.818
Deviation (%): 18.18

Quality Report for Script 2:
Total Script Words: 13
Transcribed Words: 10
Similarity Score: 0.781
Accuracy Score (Levenshtein-based): 0.615
Deviation (%): 38.46

Quality Report for Script 3:
Total Script Words: 11
Transcribed Words: 9
```

```
Similarity Score: 0.834
Accuracy Score (Levenshtein-based): 0.727
Deviation (%): 27.27
```

Start coding or generate with AI.