NLP Projektbericht – Sentiment Analysis (IMDb Reviews)

1. Einleitung

Dieses Projekt wurde im Rahmen des Natural Language Processing (NLP) Kurses durchgeführt. Ziel war es, ein Textklassifikationsproblem zu lösen, bei dem Filmrezensionen anhand ihres Inhalts als positiv oder negativ eingestuft werden. Dabei sollten ein klassischer Machine-Learning-Algorithmus (Naive Bayes) und ein Deep-Learning-Modell (LSTM) miteinander verglichen werden.

2. Datensatzbeschreibung

Für das Projekt wurde der IMDb Movie Review Datensatz verwendet. Dieser besteht aus insgesamt 50.000 englischsprachigen Filmrezensionen, die bereits in zwei gleich große Klassen unterteilt wurden:
- 25.000 positive Reviews
- 25.000 negative Reviews

Der Datensatz ist somit vollständig balanciert. Die Texte sind unterschiedlich lang, typischerweise zwischen 50 und 400 Wörtern. Die Daten stammen aus echten Nutzerbewertungen und enthalten informelle Sprache, HTML-Tags und unterschiedliche Schreibstile.

3. Datenaufbereitung (Preprocessing)

Die Textdaten wurden vor dem Training bereinigt. Die folgenden Schritte wurden durchgeführt:
- Umwandlung in Kleinbuchstaben
- Entfernen von HTML-Tags
- Entfernen von URLs
- Entfernen von Satzzeichen
- Entfernen von Zahlen
- Entfernen mehrfacher Leerzeichen

Für klassische Modelle wurde TF-IDF als Vektorisierungsmethode verwendet, während für das LSTM Sequenz-Tokenisierung, Integer-Encoding und Padding genutzt wurden.

4. Modelle und Methoden

Es wurden zwei Klassifikationsmodelle implementiert:

4.1 Naive Bayes (klassisches Machine Learning)

Für das klassische Modell wurde der Multinomial Naive Bayes Klassifikator verwendet. Die Texte wurden zuvor mit einem TF-IDF Vectorizer in numerische Feature-Repräsentationen überführt. Es wurde ein n-gram Bereich von (1,2) sowie maximal 20.000 Features verwendet.

4.2 LSTM (Deep Learning)

Für das Deep Learning Modell wurde ein LSTM-Netzwerk aufgebaut. Es bestand aus:
- Embedding Layer (20.000 Vokabeln, Embedding-Dimension 64)
- LSTM Layer mit 64 Einheiten
- Dense Layer mit ReLU-Aktivierung
- Dropout zur Regularisierung
- Sigmoid-Ausgabe für binäre Klassifikation

Die Sequenzen wurden auf eine Länge von 200 Tokens gepadded.

5. Training der Modelle

Das Naive-Bayes-Modell wurde direkt auf TF-IDF Merkmalen trainiert. Das LSTM-Modell wurde für 5 Epochen mit einer Batch-Größe von 128 trainiert. 20% der Trainingsdaten wurden als Validierungsmenge genutzt.

6. Ergebnisse

Die Ergebnisse der beiden Modelle waren wie folgt:

Naive Bayes:
- Accuracy: 86.83%
- Sehr gute Erkennung beider Klassen

LSTM:
- Accuracy: 82.68%
- Höheres Recall für negative Reviews
- Etwas schlechtere Klassifikation positiver Reviews

7. Vergleich und Interpretation

Das klassische Naive-Bayes-Modell erzielte eine höhere Genauigkeit als das LSTM-Modell. Dies ist nicht ungewöhnlich, da TF-IDF + Naive Bayes auf kurzen Texten und bei begrenztem Training oft schnelle und starke Ergebnisse liefert.

Das LSTM-Modell hätte durch folgende Maßnahmen verbessert werden können:
- Verwendung von vortrainierten Embeddings (z. B. GloVe)
- Erhöhung der Embedding-Dimension
- Nutzung eines Bidirectional LSTM
- Längeres Training (mehr Epochen)

8. Herausforderungen

Im Projektverlauf traten folgende Herausforderungen auf:
- Unterschiede in Spaltennamen des Datensatzes (‚review' → ‚text') mussten angepasst werden.

- Manche Bibliotheken fehlten in der Python-Umgebung und mussten nachinstalliert werden.
- Keras zeigte das Modell zunächst als „unbuilt“, da einige Parameter erst beim Training gesetzt werden.

Alle Probleme konnten durch Anpassungen im Code und erneutes Ausführen der Verarbeitung behoben werden.

9. Fazit

Das Projekt demonstriert die grundsätzlichen Unterschiede zwischen klassischen und neuronalen Modellen im Bereich der Textklassifikation. Obwohl Deep Learning in vielen Szenarien überlegen ist, zeigen klassische Modelle wie Naive Bayes auf strukturierten Textdaten weiterhin starke Leistungen.

Der Vergleich beider Ansätze verdeutlicht, dass Modellwahl, Datenmenge und Preprocessing entscheidend für die Performance sind.