Implementing Sentiment Analysis from scratch

Machine Learning for Business

Friedrich-Alexander Universität

Author Name

Marie Loerakker (22970768)

Luis Merkel (22971243)

Samed Yilmaz (22864573)

Maike Giehl (23107424)

Einführung

Dieser Bericht dokumentiert die Nachvollziehbarkeit unserer Gruppenarbeit im Bachelor-Kurs "Machine Learning for Business Advanced Concepts". Im Rahmen dieses Projekts haben wir ein neuronales Netzwerk entwickelt und das trainierte Modell in eine Applikation integriert.

Im Folgenden werden wir die wesentlichen Punkte unseres Projekts detailliert erläutern:

Use Case: Beschreibung des Problems und wie unser Modell zur Lösung beiträgt.

Datenauswahl, Datenverständnis und Datenvorbereitung: Vorgehensweise bei der Auswahl und Vorbereitung der Daten.

Aufbau des neuronalen Netzwerks: Architektur und Implementierung des Netzwerks.

Training des neuronalen Netzwerks und Erkenntnisse: Prozess des Trainings und gewonnene Erkenntnisse.

Aufbau der Applikation und Design Patterns: Integration des Modells in die App und angewandte Design-Patterns.

Fazit und Learnings: Zusammenfassung der Ergebnisse und gewonnenen Erkenntnisse.

Während des gesamten Berichts verweisen wir auf unser GitHub-Repository, das den gesamten Code sowie die verwendeten Daten enthält.

Use Case

In sozialen Medien werden täglich Millionen von Tweets veröffentlicht, die wertvolle Daten für Analysen bieten. Oft kann man nicht genau sagen, wie andere Nutzer auf den Tweet reagieren würden, wobei dies vor allem für Unternehmen wichtig sein könnte, die planen ein neues Produkt auf den Markt zu bringen.
Wir haben für diesen Fall eine App entwickelt, die Tweets im vornherein analysiert und vorhersagen kann, wie andere Personen auf die veröffentlichten Tweets reagieren werden. Diese Vorhersagen können Unternehmen helfen, besser zu planen und gezielt auf die zu erwartenden Reaktionen einzugehen.
Vor dem Posten des Tweets kann dieser in unsere App eingegeben und analysiert werden. Dabei bekommt der Nutzer eine Vorhersage, ob andere Leser des Tweets darauf neutral, glücklich, traurig oder besorgt reagieren werden. Die App wurde auf Grundlage eines selbst trainierten neuronalen Netzwerkes entwickelt. Zudem wurde die Oberfläche unserer App so entwickelt, um Nutzern eine einfache und intuitive Handhabung zu bieten.
Mit unserer App kann also sichergestellt werden, dass Tweets in Zukunft die vom Nutzer gewünschte Wirkung erzielen.

Datenauswahl, Daten Verständnis und Aufbereitung

Datenauswahl

In unserem Use Case haben wir uns entschieden, einen Klassifikationsalgorithmus zu entwickeln. Dieser Algorithmus soll in der Lage sein, textbasierte Daten in verschiedene Emotionskategorien zu klassifizieren. Dafür benötigen wir Datensätze, die spezifische Labels enthalten, um das Training und die Evaluierung des Modells zu ermöglichen.

Kriterien für die Datenauswahl

1. Label-Verfügbarkeit: Der Datensatz muss Labels enthalten, die unterschiedliche Emotionen oder Kategorien repräsentieren.

2. Textinhalte: Der DataFrame sollte eine Spalte mit Text ähnlichen Daten, wie beispielsweise Tweets, enthalten.

3. Sprachvielfalt: Für unser Modell fokussieren wir uns auf die englische Sprache.

4. Emotionskategorien: Unser Ziel ist es, den Text in fünf verschiedene Emotionskategorien zu klassifizieren. Daher muss der Datensatz ausreichend Beispiele für jede dieser Kategorien enthalten, um eine ausgewogene und genaue Klassifikation zu gewährleisten.

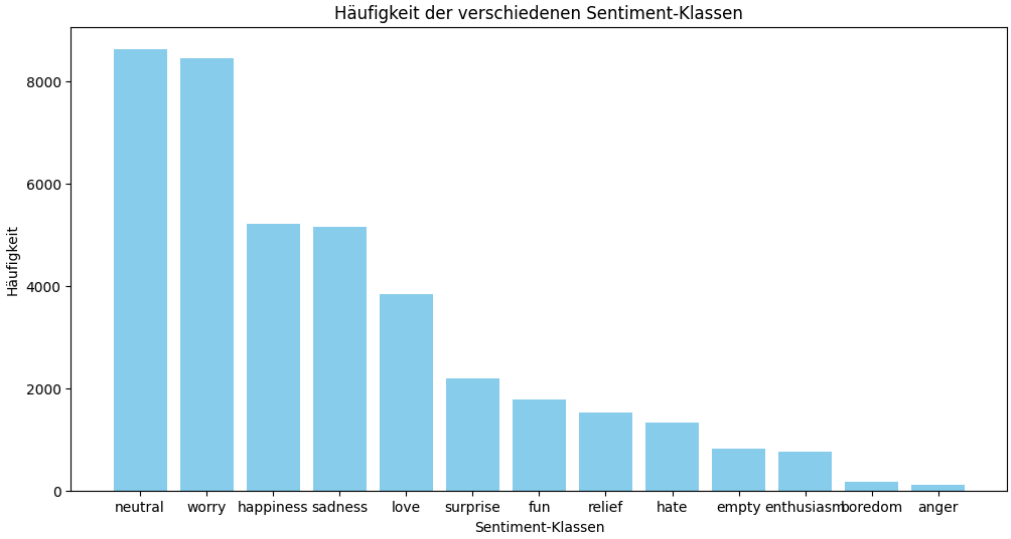
5. Datenqualität: Der Datensatz sollte möglichst wenige fehlende Werte und eine hohe Textqualität aufweisen.

6. Datenumfang: Ein ausreichender Umfang des Datensatzes ist erforderlich, um das Modell robust und verlässlich zu trainieren.

Basierend auf diesen Kriterien haben wir den im Repository verfügbaren Datensatz ausgewählt. Dieser Datensatz erfüllt unsere Anforderungen an die Label-Verfügbarkeit, Textinhalte und Emotionskategorien.

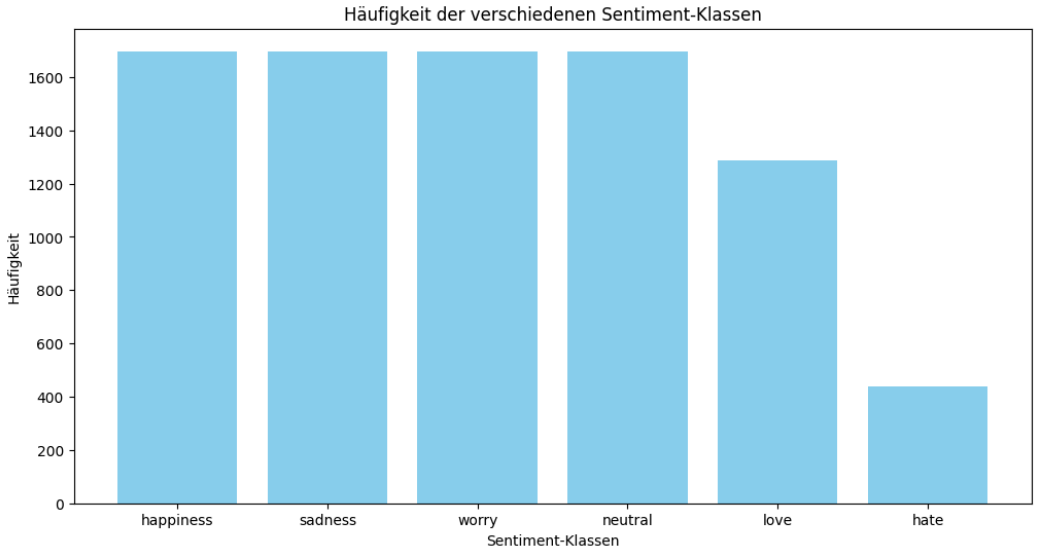
Daten Verständnis

Unser Datensatz umfasst 120.000 Zeilen und drei Spalten: ['tweet_id', 'sentiment', 'content'] mit den Datentypen int64 für tweet_id sowie object für sentiment und content. Die Content-Spalte enthält die Tweets, während Sentiment die Labels darstellt. Die Labels umfassen: empty, sadness, enthusiasm, neutral, worry, surprise, love, fun, hate, happiness, boredom, relief, anger. Zur Veranschaulichung der Verteilung der Sentiment-Klassen haben wir die Häufigkeiten der Kategorien berechnet und in einem Balkendiagramm dargestellt:



Wie bereits dargelegt, fokussieren wir uns auf die Klassifikation von fünf primären Sentiment-Kategorien. Um eine umfassende Datenbasis zu gewährleisten, haben wir die fünf am häufigsten vertretenen Labels ausgewählt. Zusätzlich haben wir die Kategorie "hate" in den Datensatz aufgenommen, da diese eine wesentliche Emotion in Tweets repräsentiert.

Unsere Analyse zeigt, dass die Verteilung der Daten im Datensatz ungleichmäßig ist. Um jedoch eine möglichst gleichmäßige Verteilung für das Training des Modells sicherzustellen, haben wir die Anzahl der Daten pro Sentiment auf das Niveau der Kategorie "sadness" begrenzt. Diese Maßnahme ermöglicht es uns, folgende Verteilung zu erreichen:



Obwohl die Kategorien "love" und "hate" in dieser Verteilung immer noch unterdurchschnittlich repräsentiert sind, haben wir uns bewusst für diesen Trade-off entschieden, um ein ausreichendes Datenvolumen für das Modelltraining zu gewährleisten. Diese Strategie hilft, die Balance zwischen Datenvielfalt und -menge zu optimieren und die Robustheit des Modells zu verbessern.

Aufbau des Neuronalen Netzwerkes

Das hier beschriebene neuronale Netzwerk hat eine spezifische Architektur mit mehreren Schichten: einer *Input-Layer*, zwei *Hidden-Layers* und einer *Output-Layer*. Jede Schicht hat folgende Konfiguration:

Input-Layer: Besteht aus 384 Neuronen, die die Eingabefeatures aus den Sätzen aufnehmen.

Erste Hidden-Layers: Enthält 50 Neuronen zur Zwischenverarbeitung der Eingabefeatures.

Zweite Hidden-Layers: Umfasst 20 Neuronen zur weiteren Verfeinerung

Output-Layer: Besteht aus 6 Neuronen, die jeweils eine der Zielklassen für Sentiments repräsentieren: neutral, worry, happiness, sadness, love und hate.

Gewichte und Biases werden initialisiert, wobei die Gewichte zufällig zwischen -0.5 und 0.5 gewählt und die Biases auf null gesetzt werden.

Aktivierungsfunktion

Die gewählte Aktivierungsfunktion ist die Sigmoid-Funktion, definiert als $\sigma(x) = 1/(1 + e^{-x})$. Diese Funktion wird auf die gewichtete Summe der Eingaben plus Biases angewendet, um sowohl eine nichtlineare Ausgabe zu erzeugen (wichtig für komplexe Muster) und die Ausgabe auf den Bereich [0;1] zu normieren.

Feature-Extraction

Die Feature-Extraktion erfolgt mit Hilfe eines vor trainierten Sentence Transformer-Modells, speziell BERT. Dieses Modell wandelt jeden Tweet in einen 768-dimensionalen Vektor um, der semantische

Informationen enthält und als Eingabefeature für das neuronale Netzwerk dient. Zu unserem bedauern ist die Rechenkraft welche dieses Modell benötigt nicht mit der Kapazität von Stramlit vereinbar daher wenden wir für die App nutzung ein mean Pooling algorithmus an welcher den Durchschnitt der Token-Embeddings (Wort-Einbettungen) berechnet (ähnlich zu manchen BERT Modellen), um eine Repräsentation für den gesamten Satz zu erzeugen. Dieses Vorgehen ermöglicht uns einen zuverlässigen Zugang zu der App und eine immer noch semantische Darstellung der Eingabe Sätze.

Trainieren des NN und Erkenntnisse

Forward Propagation

Die Forward Propagation umfasst das Durchlaufen der Eingabedaten durch das Netzwerk Schicht für Schicht. Es ist ein iterativer Prozess, der in jeder Schicht gleich abläuft und folgendermaßen aussieht:
Eine Schicht erhält einen Eingabevektor, welcher mithilfe von Matrizenmultiplikation mit den gespeicherten Gewichten multipliziert, zu den Biases addiert und schließlich durch die Sigmoidfunktion geleitet wird.
Die nachfolgende Schicht erhält diese Ausgabe wieder als Eingabevektor. Der Durchlauf endet nach der Output-Schicht. Jedes Neuron liefert hier eine Zahl zwischen 0 und 1. Das entspricht der Wahrscheinlichkeit, dass die entsprechende Emotion dem eingegebenen Tweet zugeordnet wird. Die größte Zahl ist dann die vorhergesagte Emotion. Diese Wahrscheinlichkeiten werden dann mit den tatsächlichen Labels verglichen, um den eigentlichen Fehler zu berechnen, der für das Training relevant ist.

Backpropagation und Training

Das Netzwerk wird mittels Backpropagation trainiert:

Fehlerberechnung: Der Unterschied zwischen den vorhergesagten und den tatsächlichen Labels wird durch den Mean-Square-Error berechnet.

Anpassung von Gewichten und Biases: Das Ziel der Anpassung ist es, diesen Fehlerwert zu minimieren. Das geschieht mit Gradient-Descent. Der Gradient ist allgemein ein Vektor, der die Richtung und die Rate der steilsten Zunahme einer Funktion in einem mehrdimensionalen Raum angibt. Berechnet wird er hier mit Hilfe der partiellen Ableitung des Fehlers nach jedem Gewicht und Bias. Da der Fehlerwert minimiert werden soll, wird der Gradient in negativer Richtung verwendet (also Descent).

Die Gewichte und Biases werden nun anhand des berechneten Gradienten angepasst, welcher zusätzlich mit einer Learning-Rate-Variablen multipliziert wird, um die Änderungen in einem kontrollierten Maße durchzuführen. Dieser Prozess wird iterativ durch das ganze Modell durchgeführt, nur diesmal vom Output- zum Input-Layer, da die Anpassungen auf dem berechneten Fehler basieren.

Das Training erfolgt über mehrere Epochen, wobei jede Epoche einen vollständigen Durchlauf durch den Trainingsdatensatz darstellt. Die Daten werden in Mini-Batches verarbeitet, um die Balance zwischen Rechenaufwand und effektivem Lernen zu gewährleisten. +

Ergebnis

Das trainierte neuronale Netzwerk zur Sentimentanalyse erreicht eine Testgenauigkeit von lediglich 30%. Dies bedeutet, dass nur 30% der Vorhersagen des Modells korrekt mit den tatsächlichen Sentiment Klassen übereinstimmen, mit der Klassifikation von 6 Kategorien ist unser Modell damit besser als raten. Diese niedrige Genauigkeit weist auf erhebliche Einschränkungen des Modells hin, die weiter untersucht werden müssen.

In unserem Fall ist diese niedrige Genauigkeit auf die begrenzte Datenmenge zurückzuführen. Im Rahmen dieses Kurses steht uns keine hohe Rechenkraft zur Verfügung.Der Fokus unseres Projektes liegt

auf dem Learning Outcome der Entwicklung und nicht auf einem Modell mit hoher Genauigkeit.

Daher ist es bei einer begrenzten Anzahl von Datenpunkten nicht in der Lage, die zugrundeliegenden Muster und Zusammenhänge in den Daten ausreichend zu lernen. Dies führt zu einer schlechten Generalisierungsfähigkeit auf den Testdatensatz.

Ebenso besteht bei wenigen Trainingsdaten ein hohes Risiko, dass sich das Modell an die Trainingsdaten anpasst. Das bedeutet, dass das Modell sehr gut auf die Trainingsdaten abgestimmt ist, aber bei neuen, ungesehenen Daten versagt, da es nicht in der Lage ist, die allgemeinen Muster zu erkennen (overfitting).

Aufbau der App und Design Patterns

Zur Klassifizierung von textbasierten Daten in die Emotionskategorien “neutral”, “worry”, “happiness”, “sadness”, “love” und “hate” haben wir eine Streamlit app erstellt. Insgesamt wurde die App so schlicht wie möglich gehalten, um die App benutzerfreundlich zu erhöhen.

Im Folgenden werden der Aufbau der App und die angewandten Design Patterns beschrieben.

Aufbau der App

Zur Navigation durch die drei Hauptseiten der App wurde eine Sidebar integriert. Dieser beinhaltet zur Auswahl dieser Seiten radio buttons:

- “Classify from Dataset”
- “Enter custom text”
- “Dataset”

Classify from Dataset

Mit “Classify from Dataset” kann der Benutzer mittels eines Dropdown-Menüs einen fertigen Tweet aus dem Datensatz auswählen. Dabei wird mit einem Klick auf den Button die Sentimentanalyse des jeweiligen Textes gestartet und darunter das Ergebnis der Klassifizierung angezeigt.

Enter custom Text

Dem Benutzer wird es ebenfalls ermöglicht, einen eigenen Text zu schreiben. Im Bereich “Enter custom Text” kann der selbstgeschriebene Text analysiert werden, um das Sentiment vorherzusagen.

Dataset

Mit der Option “Dataset” kann man die visualisierten Informationen über den Datensatz mittels eines Balken- oder Tortendiagramm sehen:

- Verteilung der verschiedenen Sentiment-Klassen
- Häufigkeit der verschiedenen Sentiment-Klassen
- Häufigste Wörter pro Sentiment-Klasse

Design Patterns

Design Patterns sind wiederverwendbare Komponenten, die von Designern eingesetzt werden, um wiederkehrende Herausforderungen bei der Gestaltung von Benutzeroberflächen zu bewältigen. Sie beschleunigen den Entwicklungsprozess und erhöhen die Benutzerfreundlichkeit der App. Daher wurden bei der Implementierung der App einige Design Patterns integriert:

Page Grids

Page Grids werden verwendet, um Inhalte zu organisieren und grundlegende Designprinzipien anzuwenden, um für eine Konsistenz für Hauptbereiche wie Anzeigen, Navigation, allgemeine Inhalte zu sorgen. In unserer App wurden folgende Page Grids berücksichtigt:

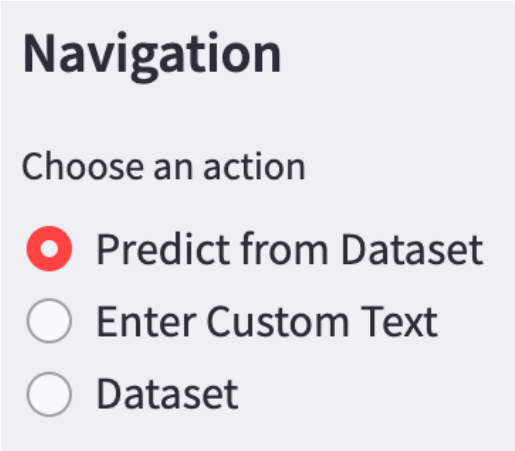
- Navigation befindet sich oben, vorzugsweise links
- Inhalt wird in der Mitte der Seite platziert

Clear Primary Action

Die wichtigsten Klickflächen werden farblich hervorgehoben, um dem Benutzer klarzustellen, welche Aktion die oberste Priorität hat.

Navigation

Die integrierte Navigation Sidebar hat eine positive psychologische Wirkung auf den Besucher, da sie mit Navigation assoziiert und somit durch die App geleitet werden.



Conclusion und Learnings

In unserem Projekt ist es uns gelungen, eine App zu entwickeln, die mithilfe eines neuronalen Netzwerks eine Sentimentanalyse bei Tweets durchführt. Die App ist dafür zuständig, die Tweets eindeutig zu identifizieren und dem Nutzer zu helfen, die hervorgerufenen Emotionen des Lesers besser zu verstehen. Der eigentliche Nutzen der App ist aber schlussendlich eher in den Hintergrund zu stellen, da aufgrund begrenzter Ressourcen das Modell nur bis zu einem gewissen Grad trainiert werden konnte. Mögliche Hindernisse, auf die wir bei der Entwicklung gestoßen sind, fingen mit der Datenqualität an. Zum bestmöglichen Training wäre ein Datensatz gut gewesen, welcher ein breiteres Spektrum an Emotionen abbildet. Zudem wäre ein größerer Datensatz hilfreich gewesen. Ein solcher stand uns auch aufgrund des zu großen Speicherplatzes nicht zur Verfügung. Ein weiterer Punkt, der das Training der App verschlechterte, war der begrenzte Arbeitsspeicher. Dieser hat uns nicht ermöglicht, auf ein größeres BERT Modell und so auf eine bessere Interpretation der Tweets zuzugreifen. Da das neuronale Netzwerk auch eher einfach gehalten ist, wurde auf neuere, aber deutlich komplexere Möglichkeiten der Implementierung verzichtet.

Dieses Projekt hat uns gezeigt, wie kompliziert und herausfordernd die Entwicklung einer solchen App sein kann. Wir haben dennoch wertvolle Einblicke in das Thema Sentiment Analysis, Datenauswahl und-verarbeitung sowie Entwicklung eines neuronalen Netzwerkes erhalten. Da das Ziel im Vorhinein schon war, eher den Aufbau von neuronalen Netzwerken besser zu verstehen, statt ein Modell mit höchster Genauigkeit zu entwickeln, haben wir dieses Ziel erfüllt.