Named Entity Recognition (NER) mit BiLSTM und RNN-Architektur mit dem Bahdanau Attention

Aldi Halili
Philosophische Fakultät
Abteilung der Computerlinguistik
Heinrich-Heine-Universität
40225, Düsseldorf, Deutschland
Aldi.halili@hhu.de

Abstract

Natural Language Processing (NLP) ist ein umfangreicher Bereich, der viele Herausforderungen mit sich bringt. Eine davon ist die Erkennung von Named Entity Recognition (NER). Die Aufgabe von NER ist die Identifizierung und Extraktion von Entitäten aus dem Text wie Personen, Orten, Organisationen und vielem mehr. (Jurafsky and Martin, 2023). In diesem Paper wird erläutert, wie die NER-Aufgabe mithilfe von Deep Learning-Methoden, insbesondere der Implementierung und Anwendung von Bi-directional Long Short-Term Memory (BiLSTM) und Recurrent Neural Network (RNN) mit dem Bahdanau Attention Mechanismus, gelöst wird. Bei der Evaluation [1] beider Architekturen wurde eine Accuracy von 99% für die BiLSTM-Architektur und 92% für die RNN-Architektur mit Bahdanau Attention erzielt. Siehe Abbildung 3.

1 Einführung

Unter NER in NLP versteht man die automatische Erkennung und Klassifizierung von Entitäten, die einen Ort, Organisation, Ereignis und Datumangabe benennen. Die automatische Erkennung und Klassifizierung der Entitäten führen dazu, dass wichtige Informationen extrahiert werden, die wertvoll für das Gesamtverständnis eines Textes sind. Für die Automatisierung der NER wurden im Laufe der Jahre verschiedene Methoden angewendet, wie beispielweise wörterbuchbasierte, regelbasierte, maschinelles Lernen (ML) und DL-Methoden.
Dieses Paper befasst sich mit einem Leistungsvergleich zwischen der BiLSTM-Architektur und der RNN-Architektur mit dem Bahdanau-Attention-Mechanismus bezüglich der NER-Aufgabe. Da die RNN-Architektur einige Herausforderungen aufgrund des verschwindenden Gradienten mit sich bringt, wird der Bahdanau-Attention-Mechanismus beitragen, um dieses Problem zu mildern. Die Kombination der RNN-Architektur mit Bahdanau-Attention-Mechanismus bringt andere diverse Vorteile mit sich, die die Leistungsfähigkeit bei der Verarbeitung von Sequenzdaten steigern. Dadurch wird eine höhere Accuracy bei der NER-Aufgabe erzielt.
Die LSTM sind Varianten von RNNs, die entwickelt wurden, um das Problem des verschwindenden Gradienten zu bewältigen. (Hochreiter und Schmidhuber, 1997).
Die BiLSTM Architektur (Erweiterung des LSTM-Modell) wird hier angewendet, um zu vergleichen wie beide Architekturen mit NER-Aufgabe umgehen und welche Leistungsfähigkeit sie leisten.

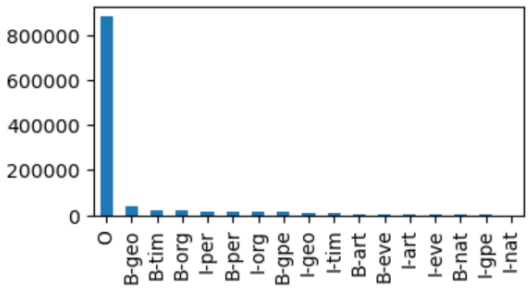
2 Dataset

Für das Training und Testen von NER-Modellen wurde das NER-Datenset (28 MG) angewendet, die auf Kaggle [2] verfügbar ist. Es besteht aus 47.959 Sätze und 1.048.575 Wörtern. In Abbildung

[1] GitHub: itsmeeeeeee/Deep-Learning-Project (github.com)

[2] Dataset: NER_dataset | Kaggle

1 ist die Tag Verteilung im Datensatz zu sehen, wobei Tag O deutlich dominiert.



Abbildung[3]1 : Tags-Verteilung im Dataset

Dieser Datensatz basiert auf dem Groningen Meaning Bank (GMB)[4] Korpus und wurde speziell zum Training eines Klassifikators entwickelt, um benannte Entitäten wie Namen, Orte usw. vorherzusagen. Die Tags, die in dem Datensatz verwendet werden, folgen dem IOB-Format. Das Inside-Outside-Beginning (IOB) ist ein häufig verwendetes Format zum Labeln von Entitäten in der Computerlinguistik, insbesondere im Zusammenhang mit der NER-Aufgabe. Die Bedeutung der IOB-Tags ist wie folgt definiert:
"B-..." (Beginn): Dieser Tag identifiziert den Anfang einer benannten Entität.
"I-..." (Inneres): Dieser Tag identifiziert den Fortlauf einer benannten Entität innerhalb der Entität.
"O" (Außerhalb): Dieser Tag bedeutet, dass das Wort keine Teil einer benannten Entität ist.
In diesem Datenset sind vereinzelte Sätze aus Nachrichtenartikeln enthalten.

3 RNN

RNNs sind eine Weiterentwicklung von Neuronalen Netzwerken (NN), die sich durch ihre Fähigkeit auszeichnen, Informationen über vorherige Schritte in einer Sequenz zu bewahren. Die Aufrechterhaltung dieser Informationen erfolgt durch die Nutzung von internen Zuständen im Netzwerk.
Jedes RNN besitzt einen "verborgenen Zustand", der Informationen aus den vorherigen Schritten der Sequenz speichert. Bei jedem Schritt wird dieser verborgene Zustand aktualisiert basierend auf der aktuellen Eingabe und dem vorherigen verborgenen Zustand. Dies wird oft als "Speicher" des Netzwerks betrachtet.
In der untenstehenden Abbildung (Son und Kim, 2020, S.3) ist auf der linken Seite ein RNN dargestellt, das nur eine versteckte Einheit enthält. Auf der rechten Seite sieht man die "entfaltete" Version des RNN. Hier ist ein Beispiel, das verdeutlicht, wie es sich zu beobachten lässt: Im Zeitschritt **t = 1** empfängt die versteckte Einheit **h1** sowohl eine Eingabe **x1** als auch den Wert der versteckten Einheit aus dem vorherigen Zeitschritt, also **h0**.

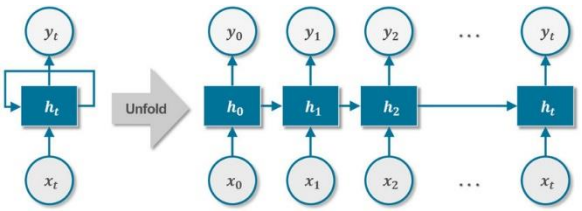


Abbildung 2: Die grundlegende Architektur eines RNN

Die Gewichte der sequenziellen Verbindungen, werden ähnlich wie die Vorwärtskopplungsverbindungen durch die Verwendung eines Optimierungsalgorithmus (wie Gradienten Abstieg) erlernt. Dieser Algorithmus wird oft in Kombination mit der Rückpropagierung verwendet, um den Gradienten der Loss Funktion zu berechnen.
Bei jedem Schritt kann das RNN auch einen Output erzeugen (Im Abbildung 2 durch **y** repräsentiert). Dieser Output wird oft basierend auf dem verborgenen Zustand berechnet. Beispielsweise könnte ein RNN in einer Spracherkennungsanwendung bei jedem Schritt ein Wort als Output ausgeben. Die Gewichte, die bei jedem Schritt angewendet werden, bleiben über die gesamte Sequenz hinweg konstant. Dieses Merkmal ermöglicht es einem RNN, Muster zu erkennen, die unabhängig von ihrer genauen Positionen über verschiedene Positionen in der Sequenz hinweg bestehen.
Obwohl RNNs theoretisch in der Lage sind, Langzeitabhängigkeiten zu erfassen, scheitern sie

[3] Deep-Learning-Project/data_preprocessing.ipynb at main · itsmeeeeeee/Deep-Learning-Project (github.com)

[4] Annotated Corpus for Named Entity Recognition | Kaggle

in der Praxis aufgrund des Problems des verschwindenden bzw. explodierenden Gradienten. (Pascanu et al., 2012)
Im Rahmen des Backpropagation-Prozesses, wo das Netzwerk seine Gewichtungen basierend auf dem Lernprozess anpasst, kann es vorkommen, dass die Gradienten - die Indikatoren für die Veränderungsgeschwindigkeit des Fehlers in Bezug auf die Gewichte - extrem ansteigen. Dies kann dazu führen, dass die Gewichtungen innerhalb des Netzwerks sehr groß oder sogar unendlich werden, was die Stabilität des Lernprozesses des Netzwerks beeinträchtigt.
Um dieses Problem zu bewältigen und eine bessere Performance bei NER-Task zu erzielen, wird im Rahmen dieses Projekts ein andere fortschrittliche RNN Architektur wie LSTM angewendet, die ausführlich im nächsten Abschnitt dargestellt wird.

4 LSTM

LSTM, das für Long Short-Term Memory steht, ist eine spezielle Architektur von RNN. Es wurde entwickelt, um das Problem der verschwindenden und explodierenden Gradienten anzugehen, indem es die Fähigkeit zur Modellierung von langfristigen Abhängigkeiten verbessert. Anders als bei traditionelle RNNs besitzen LSTMs eine besondere Gedächtniszelle und drei Gates (input-, output- und forget-gate). Die Gates steuern den Zugang von Informationen zur Gedächtniszelle, das Entfernen von Daten aus der Zelle und das Weitergeben der Informationen an den nächsten Zeitabschritt. Aufgrund dieser Struktur können LSTMs effizient Langzeitabhängigkeiten in Sequenzdaten lernen und speichern. Durch das Erfassen von Langzeitabhängigkeiten sind LSTMs besser in der Lage, die Beziehungen zwischen Wörtern in längeren Textabschnitten zu verstehen. Aufgrund ihrer hohen Leistungsfähigkeit werden LSTMs in verschiedenen Bereichen der NLP eingesetzt, wie Beispielweise Maschinelle Übersetzung, Textklassifikation, NER, usw. (Hochreiter und Schmidhuber, 1997)
Allerdings, um die aktuelle Ausgabe zu berechnen, muss die LSTM-Architektur lediglich Informationen aus vorherigen Zeitschritten berücksichtigen. In vielen Anwendungen, insbesondere bei der Verarbeitung natürlicher Sprache, ist es jedoch häufig von Vorteil, auch den Kontext aus der Zukunft oder nachfolgenden Zeitschritten zu berücksichtigen. Hier kommt das Bidirectional LSTM (BiLSTM) ins Spiel.

5 BiLSTM

BiLSTM steht für Bidirectional Long Short-Term Memory. Es handelt sich dabei um eine spezielle Art von RNN, das in der Lage ist, Informationen aus der Vergangenheit (den vorherigen Schritten in einer Sequenz) sowie aus der Zukunft (den folgenden Schritten) zu berücksichtigen. Ein BiLSTM besteht aus zwei LSTMs: einem, dass die Eingabesequenz in Vorwärtsrichtung durchläuft (von Anfang bis Ende), und einem anderen, das die Eingabesequenz in umgekehrter Reihenfolge durchläuft (von Ende bis Anfang). Die beiden LSTMs arbeiten parallel und ihre Ausgaben werden kombiniert. Das führt dazu, dass es sowohl den Kontext auf links als auch auf der rechten Seite des Tokens berücksichtigen werden kann. Dadurch wird ermöglicht, dass das Model längere Kontexte betrachten kann und besser mit Kontextabhängigkeiten umgehen kann. (Graves und Schmidhuber, 2005) Somit sind sie nicht so anfällig für die Position von Entitäten innerhalb des Textes. So können sie auch Entitäten erkennen, die am Anfang oder Ende eines Satzes stehen, was für unidirektionale Modelle oft problematisch ist. Daher sind BiLSTMs besonders nützlich für Aufgaben, bei denen der Kontext in beiden Richtungen für die Vorhersage eines bestimmten Outputs wichtig ist, wie z.B. bei der Entitätserkennung oder der Part-of-speech-Tagging (POS-Tagging). Obwohl BiLSTM Modelle viele der Probleme traditioneller RNNs lösen, haben Sie noch den Nachteil kontextabhängig zu arbeiten. Ein andere Nachteil liegt auch an die Sequenzielle Verarbeitung, die bei sehr langen Sequenzen dies zu einem hohen Rechenaufwand führen kann. Im Gegensatz dazu können Attention-Mechanismen wie zum Beispiel Bahdanau-Attention oft effizienter sein, da sie Berechnungen über die gesamte Eingabe parallel durchführen können.

6 Bahdanau-Attention-Mechanismus

Der Bahdanau-Attention-Mechanismus ermöglicht es, sich auf bedeutende Abschnitte des Inputs zu konzentrieren und die Aufmerksamkeit während der Verarbeitung dynamisch anzupassen, anstatt die gesamte Sequenz gleichzeitig zu verarbeiten. (Bahdanau et al., 2014)

Der Bahdanau Attention Mechanismus wird typischerweise in einem Encoder-Decoder Modell verwendet. Wobei **Encoder** (normalerweise ein RNN sein kann) verarbeitet die Eingabesequenz und komprimiert die Informationen in einen Kontextvektor fester Länge. **Decoder** (normalerweise ein RNN sein kann) andererseits verwendet den Kontextvektor, um die Ausgabesequenz zu erzeugen. Das Komprimieren aller Informationen einer Sequenz, insbesondere einer langen Sequenz, kann jedoch zu Information Verlust führen. Je länger die Sequenz ist, desto mehr Information verloren geht. Zudem kann die Integration des Bahdanau-Attention-Mechanismus in die RNN-Architektur dazu beitragen, das Problem des verschwindenden oder explodierenden Gradienten bei der Verarbeitung langer Sequenzen zu bewältigen. Dabei wird der Alignmentscores zwischen der Eingabe zu jedem Zeitschritt und der Ausgabe zum aktuellen Zeitschritt berechnet. Im Bahdanau-Modell wird dies erreicht, indem der versteckte Zustand des Decoders zum vorherigen Zeitschritt und der versteckte Zustand des Encoders zu jedem Zeitschritt in ein kleines Feedforward-Network eingespeist werden. Die Ausgabe dieses Netzwerk ist der Alignmentscrores. Diese dienen als Eingabe der Softmax-Funktionen, um so die Aufmerksamkeitsgewichte zu generieren. Dadurch wird sichergestellt, dass die Aufmerksamkeitsgewichte positiv sind und sich zu eins summieren. Jeder versteckte Zustand des Encoders wird dann mit seinem entsprechenden Aufmerksamkeitsgewicht multipliziert (dies ist der "Aufmerksamkeits"-Teil - stark gewichtete Encoder-Zustände haben mehr Einfluss auf die Ausgabe). Diese werden dann alle summiert, um den Kontextvektor für diesen Zeitschritt zu erzeugen. (Bahdanau et al., 2014). Die Bahdanau Attention ermöglicht es dem Modell, den gesamten Kontext eines Eingabesatzes zu berücksichtigen, wenn eine bestimmte Position analysiert wird. Es ermöglicht dem Modell, mehr Gewicht auf relevante Teile der Eingabe zu legen, die zur Erkennung einer benannten Entität beitragen können. Die Konzentration der Ausgabegenerierung (in jedem Schritt) auf verschiedene Teile der Eingabesequenz ist besonders nützlich für sequenzielle Aufgabe wie maschinelle Übersetzung und Information Extraktion, wobei wichtige Informationen aus dem Input-Text gezielt extrahiert werden können und somit bessere Leistung bei Erkennung von Named Entities erzielt werden.

7 Ergebnisse

Beide Modelle wurden mit denselben Hyperparametern trainiert. Die Embedding-Dimension betrug 100, die Hidden-Dimension 120 und es wurden 10 Trainingsepochen durchgeführt. Auch beim "Early Stopping" wurden dieselben Hyperparameter verwendet, nämlich eine Toleranz von 0,001 und ein min_delta von 0,001. Sowohl für das Loss Function als auch für den Optimizer wurde für beide Modelle derselbe Ansatz gewählt, nämlich der CrossEntropyLoss und der Adam-Optimizer.

Modell	Accuracy	Train-Loss	Test/Dev-Loss
BiLSTM	0.99	0.030	0.039
RNN-Attention	0.92	0.185	0.186
BiLSTM-CRF	0.99	0.031	0.031
Bert-Finetuned	0.91	0.311	0.311

Abbildung 3: Vergleich der Modell-Performanz auf dieselben Daten

Die Ergebnisse des gesamten Projektes in Bezug auf beide Architekturen lassen sich durch Abbildung 3 erläutern.
Bei der BiLSTM-Architektur wurde eine Accuracy von 0.99 auf Testdata erreicht, was hervorragende Ergebnisse bedeutet. Andersseits sollte man auch erwähnen, dass es sich um einen stark unausgeglichenen Datensatz handelt, die aufgrund der sehr hohen Anzahl von O-Tags, die in den Trainings- und Testdaten vorhanden sind. Bei der RNN-Architektur mit Bahdenau-Attention-Mechanismus wurden zufriedenstellende Ergebnisse mit einer Test-Accuracy von 0.92 erzielt. Bezüglich der hervorragenden erzielten Ergebnisse spielt der Datensatz eine wichtige Rolle und somit positiv auswirkt.

Wie aus Abb. 2 ersichtlich ist, wurden auch zwei andere Modelle, nämlich BiLSTM-CRF [5] und Bert-Finetuned [6] für Named Entity Recognition (NER), mit den gleichen Daten trainiert. Das BiLSTM-CRF-Modell wurde über 5 Epochen trainiert und erzielte dieselbe Accuracy bei den Testdaten wie das BiLSTM-Modell. Das Bert-Finetuned-Modell erreichte eine Test- Accuracy von 0,91, was der RNN-BahdenauAttention-Architektur sehr ähnlich ist.

Obwohl Attention-Modelle als State-of-the-Art-Methoden für NLP gelten, scheinen sie im Vergleich zu BiLSTM-Modellen in Bezug auf die NER-Aufgabe nicht besser zu sein. Im Gegenteil, BiLSTM-Modelle haben eine höhere Accuracy gezeigt. Dies wurde auch in anderen Studien belegt, wie beispielsweise beim Transformer BERT-Modell.

Literaturverzeichnis

Daniel Jurafsky & James H. Martin 7 "Speech and Language Processing." Auflage 3, Stanford University, 2023

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." arXiv preprint arXiv:1409.0473 (2014).

Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9.8 (1997): 1735-1780.

Graves, Alex, and Jürgen Schmidhuber. "Framewise phoneme classification with bidirectional LSTM and other neural network architectures." Neural networks 18.5-6 (2005): 602-610.

Huang, Zhiheng, Wei Xu, and Kai Yu. "Bidirectional LSTM-CRF models for sequence tagging." arXiv preprint arXiv:1508.01991 (2015).

Lample, Guillaume, et al. "Neural architectures for named entity recognition." arXiv preprint arXiv:1603.01360 (2016).

Li, Jing, et al. "A survey on deep learning for named entity recognition." IEEE Transactions on Knowledge and Data Engineering 34.1 (2020): 50-70.

Yadav, Vikas, and Steven Bethard. "A survey on recent advances in named entity recognition from deep learning models." arXiv preprint arXiv:1910.11470 (2019).

Ma, Xuezhe, and Eduard Hovy. "End-to-end sequence labeling via bi-directional lstm-cnns-crf." arXiv preprint arXiv:1603.01354 (2016).

Pascanu, Razvan, Tomas Mikolov, and Yoshua Bengio. "On the difficulty of training recurrent neural networks." International conference on machine learning. Pmlr, 2013.

Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).

Son, Hyojoo, and Changwan Kim. "A deep learning approach to forecasting monthly demand for residential–sector electricity." *Sustainability* 12.8 (2020): 3103.

Christian Wurm "Neuronale Netze und Tiefe Architekturen" Heinrich-Heine-Universität Düsseldorf, 2022

[5] BiLSTM-CRF-Modell: Building a Named Entity Recognition model using a BiLSTM-CRF network (dominodatalab.com)

[6] Bert-Finetuned: NER - Named Entity Recognition Tutorial | Kaggle

[7] Daniel Jurafsky & James H. Martin: Speech and Language Processing (stanford.edu)