

Wilhelm-Ostwald-Schule, Gymnasium der Stadt Leipzig

Dokumentation zur Besonderen Lernleistung

im Fachbereich: Informatik

Thema: KI-basierte Analyse und räumliche
Verortung von Polizeiberichten am
Beispiel Sachsen

vorgelegt von: Felix Neumann

Schuljahr: 2021/2022

Externe(r) Betreuer: Herr Paschke
Esri Deutschland GmbH

Interner Betreuer: Herr Karl

Leipzig, 27. Januar 2022

Neumann, Felix

KI-basierte Analyse und räumliche Verortung von Polizeiberichten am Beispiel Sachsen

Besondere Lernleistung im Fach Informatik am Wilhelm-Ostwald-Gymnasium Leipzig, 2022

40 Seiten

Abstract

This paper focuses on the use of artificial intelligence (AI) for natural language processing. It examines whether using AI adds value in terms of the time needed to analyze police reports. The data used in this study are police reports provided by police departments of Saxony, mainly focusing on Leipzig, Chemnitz, and Görlitz. The goal was to extract information such as type of crime, place of crime, date of crime, physical description of criminal from non-standardized texts with the highest possible accuracy. This was achieved by using a natural language processing AI performing named-entity recognition. The next step was to feed the location data into a geocoder in order to convert it into latitude and longitude. This information can now be used to display these crime reports on an interactive map where hotspots are visualized and the user can select the locations to get more information about the felony. We expect an accuracy of more than 98 % after training the deep learning model with thousands of police reports which were annotated by hand. This leads to a tremendous decrease in manual labor hours of a very repetitive task and cannot only be applied to Saxony's police departments but to police and fire departments as well as other types of reports like weather reports from stations across the globe.

Kurzreferat

Diese Arbeit behandelt den Einsatz von künstlicher Intelligenz (KI) bei der Verarbeitung natürlicher Sprache. Es soll untersucht werden, ob der Einsatz von KI einen Mehrwert in Bezug auf den Zeitaufwand für die Analyse von Polizeiberichten bringt. Die Daten, die in dieser Studie verwendet werden, sind Polizeiberichte, die von sächsischen Polizeidienststellen zur Verfügung gestellt werden. Der Schwerpunkt liegt hierbei auf den Standorten Leipzig, Chemnitz und Görlitz. Ziel ist es, Informationen wie Straftat, Tatort, Datum der Tat, Aussehen des/der Täter:in usw. aus nicht standardisierten Texten mit höchstmöglicher Genauigkeit zu extrahieren. Dies wird mit einer KI zur Verarbeitung natürlicher Sprache erreicht, die eine Named-Entity-Recognition durchführt. Im nächsten Schritt werden die Ortsdaten in einen Geocoder eingespeist, um sie in Breiten- und Längengrade umzuwandeln. Diese Informationen können nun verwendet werden, um diese Verbrechensberichte auf einer interaktiven Karte anzuzeigen. So können Hotspots visualisiert werden und der Benutzer kann Orte auswählen, um weitere Informationen über das Verbrechen zu erhalten. Wir erwarten eine Genauigkeit von mehr als 98 %, nachdem wir das Deep-Learning-Modell mit Tausenden von Polizeiberichten trainieren, die von Hand annotiert worden sind. Dies führt zu einer enormen Verringerung der manuellen Arbeitsstunden einer sehr repetitiven Aufgabe und kann nicht nur auf Sachsens Polizeidienststellen, sondern auch auf Polizei- und Feuerwehrdienststellen sowie auf andere Arten von Berichten, wie z. B. Wetterberichte, von Stationen auf der ganzen Welt angewendet werden.

Inhaltsverzeichnis

| | | |
|----------|---|-----------|
| 1 | Einleitung | 3 |
| 1.1 | Themeneinführung | 3 |
| 1.1.1 | Begründung der Wahl | 3 |
| 1.1.2 | Ziel | 3 |
| 1.1.3 | Vorweggenommenes Fazit | 4 |
| 1.1.4 | Problemstellung | 4 |
| 2 | Grundlagen | 5 |
| 2.1 | Künstliche Intelligenz und Natural Language Processing . . . | 5 |
| 2.2 | Modelle und Backbones | 5 |
| 2.3 | Named Entity Recognition, Text Classification und Sequence To Sequence | 7 |
| 2.4 | GIS | 7 |
| 2.5 | Python und Jupyter Notebooks | 8 |
| 3 | Methodik | 9 |
| 3.1 | Extrahieren der Daten von der Website der Polizei Sachsen . | 9 |
| 3.2 | Labeling der Daten mit doccano | 11 |
| 3.3 | ArcGIS API for Python | 14 |
| 3.3.1 | EntityRecognizer | 14 |
| 3.3.2 | TextClassifier | 17 |
| 3.4 | Zusammenführung der Daten und Geocoding | 21 |
| 3.5 | Integration der Modellergebnisse ins GIS | 23 |
| 3.6 | Feedback-Loop | 26 |
| 4 | Ergebnisse | 29 |
| 4.1 | Vergleich verschiedener Backbones | 29 |

| | | |
|----------|---|-----------|
| 5 | Fazit | 31 |
| 5.1 | Zusammenfassung des Vorgehens | 31 |
| 5.2 | Erlangte Erkenntnisse | 31 |
| 5.3 | Ausblick | 32 |
| 6 | Literaturverzeichnis | 33 |
| 7 | Abkürzungsverzeichnis | 36 |
| 8 | Verzeichnis der Fremd- und Fachworterklärungen | 37 |
| 9 | Verzeichnis der Abbildungen, Tafeln und Tabellen | 38 |
| | Danksagung | |
| | Selbstständigkeitserklärung | |

1 Einleitung

1.1 Themeneinführung

In dieser Arbeit wird die Anwendung verschiedener Techniken bzw. Ansätze von Natural Language Processing, vor allem Named Entity Recognition (NER), Text Classification (TC) und Sequence To Sequence (STS), auf Berichte der Polizei Sachsen thematisiert. Anschließend wird beschrieben, wie mithilfe von Esri Software ein Workflow erstellt werden kann, bei welchem aus diesen Polizeiberichten erst Features extrahiert, dann Adressen geokodiert und die Ergebnisse zuletzt grafisch dargestellt werden können.

1.1.1 Begründung der Wahl

Die ursprüngliche Idee war, Natural Language Processing an Zeitungsartikeln durchzuführen. Doch nach einiger Zeit wurde klar, dass diese Texte zu komplex und unstrukturiert sind. Um solche Texte erfolgreich durch KI analysieren zu können, müssten wahrscheinlich zehn- bis hunderttausende Trainingsdaten verfügbar sein. Daher wurde die Entscheidung getroffen, standardisiertere Texte, wie z. B. Dienstberichte der Polizei, zu verwenden.

1.1.2 Ziel

Das Ziel dieser Arbeit ist die Erstellung eines interaktiven Dashboards, auf welchem Nutzende Einsätze der Polizei Sachsen einsehen können. Des Weiteren ist es möglich, zu jedem Einsatz wichtige Informationen wie Datum, Informationen über Kriminelle, usw. abzurufen.

1.1.3 Vorweggenommenes Fazit

Die Genauigkeit des EntityRecognizers mit nahezu 100 % und des TextClassifiers mit etwa 89,68 % haben die Erwartungen übertroffen und zeigen, dass KI vor allem für die Analyse standardisierter Texte überaus hilfreich sein kann. So konnten etwa 68 % der Berichte komplett automatisch Koordinaten zugeordnet werden und auf einer interaktiven Karte angezeigt werden, während alle anderen nur kleinere Korrekturen vom Menschen benötigen. Klar ist, dass die Nutzung von KI für solche Anwendungsfälle zeitsparend ist und vor allem bei großen Datenmengen um ein Vielfaches schneller und günstiger ist als die manuelle Bearbeitung.

1.1.4 Problemstellung

Obwohl die Polizei Daten bereits in einer normalisierten Form speichert, kann dieses Beispiel Anwendung auf ältere Datensätze finden, welche noch nicht digital erfasst wurden. Auch ältere Polizeiberichte werden aufgrund von laufenden Ermittlungen immer wieder benötigt, weshalb es sinnvoll ist, diese ebenfalls auf eine moderne Weise zu speichern. Da die Vorfälle immer an einem bestimmten Ort stattgefunden haben, bringt die automatisierte Verortung und Speicherung innerhalb eines geografischen Informationssystems viele Vorteile. So ist es beispielsweise möglich in einer Datenbank nach Berichten zu suchen, welche nahe einem bestimmten Ort oder in einer bestimmten Zeitspanne stattfanden. Über weitergehende zeit-räumliche Analysen können die gespeicherten Berichte auf zeit-räumliche Muster untersucht werden, z. B. wo und wann es eine signifikante Häufung von Vorfällen gegenüber zufälligen Ereignissen gab.

2 Grundlagen

2.1 Künstliche Intelligenz und Natural Language

Processing

Bei Künstlicher Intelligenz handelt es sich um eine Form der Intelligenz von Maschinen, die anders als Natürliche Intelligenz bei Menschen oder Tieren ohne Bewusstsein oder Emotionalität auskommt^[Wikipedia-Autoren, 2021a].

In diesem Projekt handelt es sich um eine schwache KI, die Menschen beim Labeling von genormten Texten unterstützen soll^[Wikipedia-Autoren, 2021e].

Genauer wird hier Deep Learning, eine Art des maschinellen Lernens, welche auf künstlichen neuronalen Netzen basiert^[Wikipedia-Autoren, 2021c], verwendet. Künstliche neuronale Netze sind Systeme, die im Aufbau grundlegend auf biologischen neuronalen Netzen, wie beispielsweise dem menschlichen Gehirn, basieren^[Wikipedia-Autoren, 2021b].

2.2 Modelle und Backbones

Um die Effizienz der Künstlichen Intelligenz erheblich zu steigern, kann sie auf vorgefertigten Backbones aufbauen, die bereits ein Verständnis von Sprache besitzen. Da ein Großteil der Layer neuronaler Netze sich nicht wesentlich voneinander unterscheiden, ist die Verwendung von Backbones sehr sinnvoll. Layer sind dabei eine Gruppierung von Neuronen, welche Transformationen auf ihren jeweiligen Input anwenden. Alle dieser Layer, bis auf die Layer nahe dem Output-Layer haben die Aufgabe, grundlegende Strukturen zu erkennen. Da diese für nahezu jede mögliche Anwendung erforderlich sind, ist es nur selten ratsam, das Modell von

Grund auf zu trainieren^[Vasani, 2019]. Diese Backbones werden generell in kontextfreie und kontextuelle vortrainierte neuronalen Netze unterschieden, wobei die kontextuellen Netze weiterhin in unidirektionale und bidirektionale neuronale Netze unterteilt werden können^[Devlin, Chang, Scientists et al. 2018a].
^[Devlin, Chang, Lee et al. 2018]. Kontextfreie neuronale Netze erzeugen für jedes Wort in ihrem Vokabular eine Definition, weshalb zur sinnvollen Analyse von Sätzen kontextuelle Netze häufig besser geeignet sind. Diese können bei mehrdeutigen Wörtern anhand des Kontexts entscheiden, welche der Definitionen in diesem Fall benötigt wird. Unidirektionale neuronale Netze können jedoch nur vorwärts nach Kontexthinweisen suchen, während bidirektionale auch rückwärts auf den Inhalt schließen können. Dies sollte vor allem bei komplexeren Sätzen zu einer Steigerung der Genauigkeit der KI um bis zu 24 % führen, während nur ein Hundertstel der Trainingsdaten benötigt wird^{[Howard und Ruder, 2018][Peters et al. 2018]}.

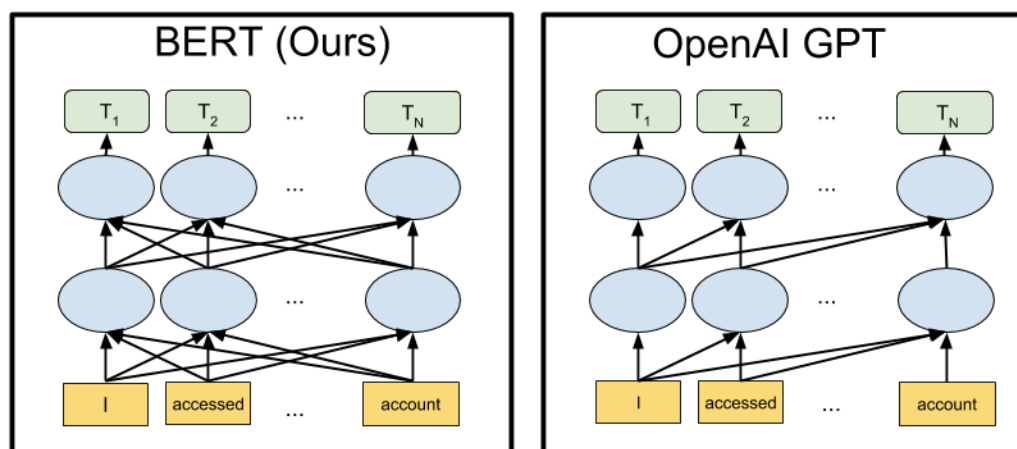


Abbildung 1: Vergleich des Aufbaus von BERT, einem bidirektionalen, und GPT, einem unidirektionalen Netz^[Devlin, Chang, Scientists et al. 2018b].

2.3 Named Entity Recognition, Text Classification und Sequence To Sequence

Named Entity Recognition ist eine Technik des Natural Language Processing (NLP), bei welcher Entitäten aus einem Text in vordefinierte Kategorien eingeordnet werden^[Roldós, 2020]. Aufgrund dieser Eigenschaften lässt sich ein solches Modell anwenden, um beispielsweise Adressen, Namen, Uhrzeiten usw. aus Texten herauszufiltern. Bei Text Classification handelt es sich ebenfalls um eine NLP-Technik, jedoch wird hierbei jedem Text direkt eine Kategorie zugeordnet. Diese Modelle werden z. B. häufig verwendet, um die Sprache von Texten zu erkennen oder zu entscheiden, ob es sich bei einer E-Mail um Spam handelt. Sequence To Sequence Modelle, welche auch in die Kategorie des NLP fallen, „übersetzen“ Texte in andere. Beispielsweise kann man mit diesen Modellen Texte von einer Sprache in eine andere übersetzen oder auch nicht standardisierte Adressen in standardisierte umwandeln.

2.4 GIS

Ein geografisches Informationssystem (GIS) ist ein System, das alle Arten von Daten erstellt, verwaltet, analysiert und kartiert. Ein GIS verbindet Daten mit einer Karte und integriert Standortdaten mit allen Arten von beschreibenden Informationen. Auf diese Weise entsteht eine Grundlage für die Kartierung und Analyse, die in der Wissenschaft und in fast allen Branchen eingesetzt wird. GIS hilft den Benutzenden, Muster, Beziehungen und geografische Zusammenhänge zu verstehen. Zu den Vorteilen gehören eine verbesserte Kommunikation und Effizienz sowie eine bessere Verwaltung

und Entscheidungsfindung^[Esri, 2021b].

2.5 Python und Jupyter Notebooks

Die Entscheidung für die Programmiersprache Python war aufgrund der bestehenden ArcGIS API (siehe 3.3 ArcGIS API for Python) logisch. Allgemein handelt es sich bei Python um die meistverwendete Programmiersprache im Gebiet der Data Science, wodurch viele Bibliotheken bereits verfügbar sind. Durch die Verwendung von Jupyter Notebooks können einzelne Zeilen und Abschnitte des Codes separat ausgeführt werden. Ein weiterer Vorteil dieser Notebooks ist, dass der Output automatisch visualisiert wird, wenn es sich um einen bekannten Datentyp handelt.

3 Methodik

3.1 Extrahieren der Daten von der Website der Polizei

Sachsen

Die Daten auf der Website der Polizei Sachsen liegen in einem Format vor, indem sie nicht gescraped werden können, d. h. sie lassen sich nicht von der Website herunterladen. Daher ist es erforderlich die Daten manuell zu speichern. Hierfür wurde ein Python-Script erstellt, welches den Nutzer beim Start nach einer ID fragt. Anschließend wurde ein Keyboard Controller erstellt, welcher es ermöglicht, Tasten zu drücken. Des Weiteren wurde ein Keyboard Listener gestartet, welcher beim Drücken der F8-Taste eine Funktion ausführt. Nun wurden automatisch die Tasten Ctrl und C gedrückt und somit die aktuelle Textauswahl kopiert. Daraufhin pausiert das Programm für eine Zehntelsekunde, um sicherzustellen, dass der Text kopiert wurde. Anschließend kann über `win32clipboard`^[Hammond et al. 2021] auf den kopierten Text zugegriffen werden. Aus diesem wurden nun leere Zeilen entfernt und der Text anschließend in eine Textdatei mit der ID als Name gespeichert. Daraufhin wird die ID um 1 erhöht und der Listener wartet auf eine weitere Tastenbetätigung.


```

1  import win32clipboard # to access clipboard
2  import time # to sleep script
3  from pynput.keyboard import Listener, Key, Controller # to
   ↪ listen for keystrokes and press keys
4
5  keyboard_controller = Controller()
6
7  start_id = int(input("Start-ID: "))
8
9  def on_press(key):
10     global start_id
11     if key == Key.f8:
12         with keyboard_controller.pressed(Key.ctrl):
13             keyboard_controller.press('c')
14             time.sleep(0.1)
15             win32clipboard.OpenClipboard()
16             with open("../Reports/" + "{:04d}".format(start_id) +
   ↪ ".txt", "w", encoding="utf8") as file:
17                 file.write("\n".join([ll.rstrip() for ll in
   ↪ win32clipboard.GetClipboardData().splitlines()
   ↪ if ll.strip()]))
18             win32clipboard.CloseClipboard()
19             start_id += 1
20
21  with Listener(on_press=on_press) as listener:
22     listener.join()

```

Abbildung 3: Script zum Speichern der Daten der Polizei-Website.

3.2 Labeling der Daten mit doccano

Das Open-Source-Werkzeug doccano vereinfacht das Labeln von Texten. Zum Importieren müssen alle Polizeiberichte in einer Textdatei zusammengefasst werden, wobei auf jeder Zeile ein neuer Bericht steht. Dafür werden alle Berichte einzeln geöffnet und Zeilenumbrüche durch Leerzeichen ersetzt, woraufhin diese Texte durch Zeilenumbrüche getrennt in eine Zeichenfolge gespeichert und anschließend in eine neue Datei geschrieben werden.

```

1  import re, os
2
3  s = ""
4
5  for item in os.listdir('BeLL/Reports'):
6      file = open("BeLL/Reports/" + item, 'r', encoding='utf8')
7      ogtext = file.read()
8      file.close()
9      text = [line for line in ogtext.split('\n') if line.strip()
10             ↪ != '']
11      for string in text:
12          s += string + ' '
13      s += '\n'
14
15  file = open("BeLL/allreports.txt", 'w', encoding='utf8')
16  file.write(s)
17  file.close()

```

Abbildung 4: Erstellung einer mit doccano importierbaren Datei.

Anschließend werden in doccano die Labels erstellt. Des Weiteren können Tastenkombinationen festgelegt werden, um das Annotieren zu beschleunigen.

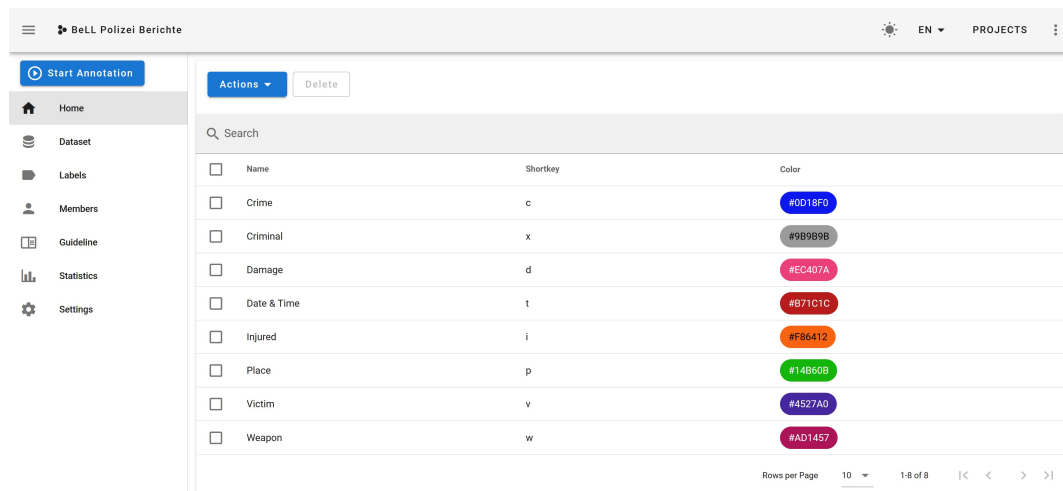
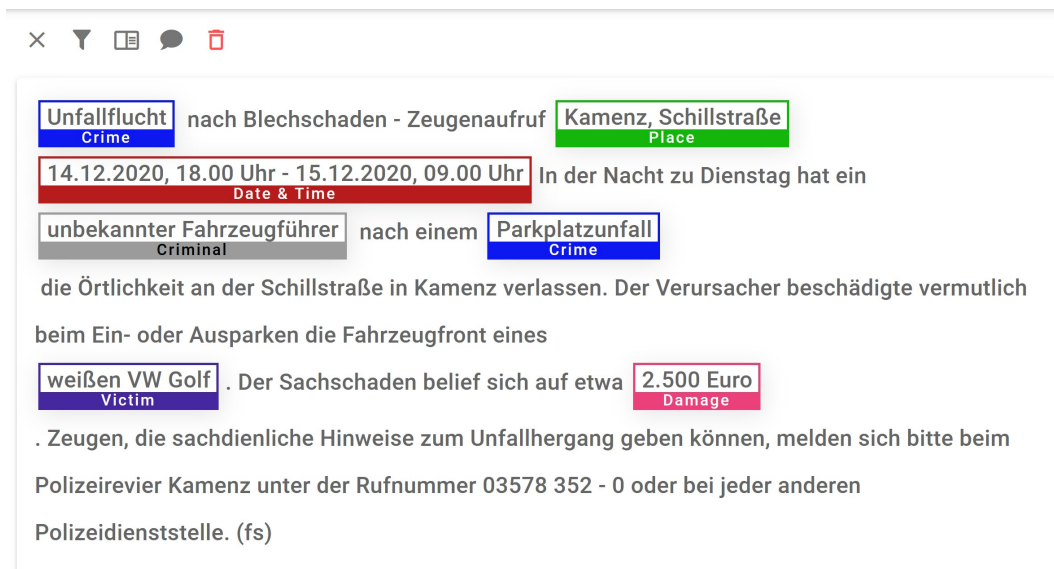


Abbildung 5: Erstellte Labels in doccano.

Als Nächstes werden etwa 650 der Berichte annotiert. Hierbei werden Abschnitten des Textes manuell Labels zugewiesen. Während diese Arbeit

sehr repetitiv ist, kam es immer wieder zu Ausnahmesituationen, beispielsweise durch das Auftreten von Berichten, bei welchen es sich um Morde handelte oder wo eine Tatwaffe angeführt wurde.



The screenshot shows a news article snippet with several key phrases highlighted by colored boxes and labeled with semantic categories. At the top left, there are icons for a close button (X), a filter (funnel), a document, a speech bubble, and a trash can. The text of the article is as follows:

Unfallflucht (Crime) nach Blechschaden - Zeugenaufruf **Kamenz, Schillstraße** (Place)
14.12.2020, 18.00 Uhr - 15.12.2020, 09.00 Uhr (Date & Time) In der Nacht zu Dienstag hat ein
unbekannter Fahrzeugführer (Criminal) nach einem **Parkplatzunfall** (Crime)
die Örtlichkeit an der Schillstraße in Kamenz verlassen. Der Verursacher beschädigte vermutlich
beim Ein- oder Ausparken die Fahrzeugfront eines
weißen VW Golf (Victim). Der Sachschaden belief sich auf etwa **2.500 Euro** (Damage).
. Zeugen, die sachdienliche Hinweise zum Unfallhergang geben können, melden sich bitte beim
Polizeirevier Kamenz unter der Rufnummer 03578 352 - 0 oder bei jeder anderen
Polizeidienststelle. (fs)

Abbildung 6: Beispiel eines annotierten Berichts.

Diese annotierten Daten können im Anschluss als JSONL-Datei (JSON Lines)^[Ward, 2021] gespeichert werden.

```

1 {"id": 166, "text": "Unfallflucht nach Blechschaden -
  → Zeugenaufwurf Kamenz, Schillstraße 14.12.2020, 18.00 Uhr
  → - 15.12.2020, 09.00 Uhr In der Nacht zu Dienstag hat ein
  → unbekannter Fahrzeugführer nach einem Parkplatzunfall
  → die Örtlichkeit an der Schillstraße in Kamenz verlassen.
  → Der Verursacher beschädigte vermutlich beim Ein- oder
  → Ausparken die Fahrzeugfront eines weißen VW Golf. Der
  → Sachschaden belief sich auf etwa 2.500 Euro. Zeugen, die
  → sachdienliche Hinweise zum Unfallhergang geben können,
  → melden sich bitte beim Polizeirevier Kamenz unter der
  → Rufnummer 03578 352 - 0 oder bei jeder anderen
  → Polizeidienststelle. (fs)", "meta": {},
  → "annotation_approver": "admin", "comments": [],
  → "labels": [[0, 12, "Crime"], [398, 408, "Damage"], [146,
  → 172, "Criminal"], [345, 359, "Victim"], [46, 66,
  → "Place"], [67, 113, "Date & Time"], [184, 199,
  → "Crime"]]}

```

Abbildung 7: Beispielzeile aus dem exportierten Datensatz.

Hierbei erhält jede der Zeilen zusätzlich zum Text eine ID und die Labels, welche im Array, einer Datenstruktur in welcher Daten mithilfe von Indizes organisiert werden können, "labels" jeweils als Array mit dem Startzeichen und dem Endzeichen gespeichert werden.

3.3 ArcGIS API for Python

3.3.1 EntityRecognizer

Beim EntityRecognizer handelt es sich um eine Python-Klasse aus der ArcGIS API, welche ein Entity Recognition Model erstellt, womit Text-Entitäten aus unstrukturierten Texten extrahiert werden können. Der erste Schritt ist nun das Importieren der Daten, welche aus doccano als JSONL-Datei exportiert wurden. Dabei wird nicht nur der Pfad und Datentyp der JSONL-

Datei spezifiziert, sondern auch angegeben, welches Label Adressinformationen enthält, in diesem Fall `'Place'`.

```
1 data = prepare_data(path="annotations.json",  
    ↪ class_mapping={'address_tag': 'Place'},  
    ↪ dataset_type='ner_json')
```

Abbildung 8: Einlesen der Daten aus der JSONL-Datei.

Als Nächstes wird ein Objekt der Klasse `EntityRecognizer` erstellt, wobei die Sprache sowie ein möglicher Backbone angegeben werden kann. Wie in 4.1 Vergleich verschiedener Backbones untersucht, handelt es sich bei dem Standardwert `backbone="spacy"` um das sowohl schnellste als auch genaueste Modell.

```
1 ner = EntityRecognizer(data, lang="de",  
2     # backbone="bert-base-german-cased"  
3     )
```

Abbildung 9: Erstellung eines `EntityRecognizer`.

Daraufhin sucht man nach der optimalen Learning-Rate, um das Training des Modells zu beschleunigen. Bei der Learning-Rate handelt es sich um einen Tuning-Parameter, welcher die Schrittgröße jeder Iteration (Epoche) bestimmt, während eine Bewegung zum Minimum einer Verlustfunktion hin stattfindet. Die Learning-Rate spiegelt somit auch die Lerngeschwindigkeit des Modells wider. Eine kleine Learning-Rate führt zu einer großen Trainingsdauer, da die Schritte entlang der Verlustfunktion sehr klein sind. Eine große Learning-Rate hingegen kann dazu führen, dass mit einer Iteration das Minimum der Verlustfunktion übersprungen wird^[Esri, 2021a].

```
1 lr = ner.lr_find()
```

Abbildung 10: Finden der optimalen Learning-Rate.

Im folgenden Schritt wird das Modell unter Angabe der Learning-Rate und Epochenanzahl trainiert. Zusätzlich kann auch `early_stopping=True` angegeben werden, um das Training zu beenden, wenn über die letzten Epochen keine Verbesserung der Genauigkeit erfolgt ist.

```
1 ner.fit(epochs=80, lr=lr, early_stopping=True)
```

Abbildung 11: Trainieren des Netzes.

Der nächste Schritt ist das Speichern des soeben erstellten Modells, sodass es zu einem späteren Zeitpunkt erneut geladen werden kann, um es auf Texte anzuwenden.

```
1 ner.save('crime_model_improved3')
```

Abbildung 12: Speichern des Modells.

Mit der Methode `extract_entities("Reports/", drop=True)` kann das Modell auf Datensätze angewendet werden. Diese müssen dafür als Textdateien im angegebenen Pfad liegen. Das optionale Argument `drop=True` gibt an, ob Berichte, welche keine Adressinformationen enthalten, automatisch aus den Ergebnissen entfernt werden sollen. Die Funktion gibt daraufhin einen `pandas-DataFrame`^[Mendel et al. 2021] zurück, welcher die extrahierten Daten enthält. Bei einem `pandas-DataFrame` handelt es sich um eine zweidimensionale und größenveränderliche tabellarische Datenstruktur^[pandas Entwicklerteam, 2021].

```
1 results = ner.extract_entities("Reports/", drop=True)
```

Abbildung 13: Extrahieren der Informationen aus Dateien im Reports-Ordner.

Als Nächstes werden die Ergebnisse gespeichert.

```
1 results.to_csv('resultsspacey.csv')
```

Abbildung 14: Speicherung der Ergebnisse.

3.3.2 TextClassifier

Der TextClassifier weist, anders als der EntityRecognizer, ganzen Texten eine Kategorie zu.

Dafür wurden in doccano die folgenden Kategorien erstellt: Diebstahl / Einbruch, Unfall, Verstoß gg. Corona-Verordnung, Drogenmissbrauch, Sachbeschädigung, Tempolimit, Brand(stiftung), Andere. „Andere“ beinhaltet dabei alle Kategorien, die nicht häufig genug auftraten. Beispielsweise enthielten alle dieser Datensätze nur einen Bericht, der in die Kategorie Mord fallen würde. Da dies nicht genügend Daten sind, um die KI daran zu trainieren, wurde diese Zusammenfassung vorgenommen.

Auch hier werden zu Beginn Trainingsdaten von doccano importiert. Im Falle des TextClassifiers handelt es sich hier um eine TXT-Datei im TSV-Format. Dabei enthält die erste Spalte die Kategorie für den zugehörigen Text, welcher sich in der zweiten Spalte befindet. Des Weiteren wird mit den Argumenten `text_columns="text"` und `label_columns="crime"` angegeben, welchen Namen die Spalte hat, in welcher sich Texte befinden und welchen Namen die Spalte hat, in der sich die Labels befinden.

```
1 data = prepare_textdata(path=os.getcwd(),
    ↪ task="classification", train_file="Book1.txt",
    ↪ text_columns="text", label_columns="crime")
```

Abbildung 15: Einlesen der Trainingsdaten aus der TXT-Datei.

Der nächste Schritt ist die Erstellung eines TextClassifiers unter Angabe eines Backbones. Für den TextClassifier ist das Modell spaCy, der schnellste vorgefertigte Backbone, nicht verfügbar.

```
1 model = TextClassifier(data,  
    ↳ backbone="bert-base-german-cased")
```

Abbildung 16: Erstellung eines TextClassifiers.

Nun muss, wie beim EntityRecognizer, die optimale Learning-Rate gefunden werden.

```
1 lr = model.lr_find()
```

Abbildung 17: Finden der optimalen Learning-Rate.

Darauf folgt, ähnlich wie beim EntityRecognizer, die `fit`-Funktion, mit welcher das Modell in diesem Fall für vier Epochen trainiert wird.

```
1 model.fit(epochs=4, lr=lr)
```

Abbildung 18: Training des Modells.

Anschließend wurden die Layer des Modells „aufgetaut“, sodass es weiter feinjustiert werden kann. Das „Auftauen“ der Layer beschreibt dabei das Zulassen von Modifikationen an den künstlichen Neuronen der Layer.

```
1 model.unfreeze()
```

Abbildung 19: „Auftauen“ des Modells.

Nun folgt das Training mit sechs weiteren Epochen.

```
1 model.fit(epochs=6, lr=lr)
```

Abbildung 20: Training des Modells (Feinjustierung).

Mit `model.accuracy()` kann die Genauigkeit des Modells ausgegeben werden.

Anschließend wurde das Modell für die spätere Verwendung gespeichert.

```
1 model.save("classifier_model")
```

Abbildung 21: Speichern des Modells.

Folgend die Anwendung des TextClassifier-Modells auf einen Beispielbericht:

```
1 model.predict("""Mit 243 km/h unterwegs
2 Zeit:      30.04.2021, 10:30 Uhr bis 16:00 Uhr
3 Ort:       A 4, Fahrtrichtung Erfurt
4 Nur Fliegen ist schöner, dachte sich vermutlich der Fahrer eines
  ↳ Audi S6, der am Freitag auf der Autobahn 4 in Fahrtrichtung
  ↳ Erfurt unterwegs war.
5 Beamte der Verkehrspolizeiinspektion führten Kontrollen in der
  ↳ Nähe der Anschlussstelle Meerane durch und ermittelten dabei
  ↳ die Geschwindigkeit von 2.684 Fahrzeugen. Spitzenreiter war
  ↳ der Audi-Fahrer. Bei ihm zeigte das Messgerät 243 km/h -
  ↳ erlaubt waren 100 km/h. Die Ermittlungen zum Fahrzeugführer
  ↳ dauern an. Fest steht bereits, dass er mit zwei Punkten in
  ↳ Flensburg sowie einem Bußgeld von 600 Euro und einem
  ↳ dreimonatigen Fahrverbot rechnen muss. (kh)""")
```

Abbildung 22: Anwendung des Modells auf Beispieldatensatz.

Als Nächstes werden die Inhalte aller Dateien in eine Liste geladen.

```

1     filelist = os.listdir(os.getcwd()+"/Reports")
2     textdata = []
3     for file in filelist:
4         with open("Reports/" + file, "r", encoding="utf8") as
           ↪ file:
5             textdata.append(file.read())

```

Abbildung 23: Laden der Inhalte aus den TXT-Dateien.

Anschließend wird eine Liste für die Ergebnisse erstellt und eine Methode definiert, welche Tupel, Listen an Elementen unter Beachtung der Reihenfolge, in Dictionaries, Listen von Schlüssel-Wert-Paaren, umwandelt, welche statt des Textes den Dateinamen enthalten.

```

1     results = []
2     def tuple_to_dictionary(tuple, filename):
3         return {"filename": filename, "label": tuple[1],
           ↪ "confidence": tuple[2]}

```

Abbildung 24: Erstellung einer Liste für die Ergebnisse sowie einer Methode, welche die Umwandlung eines Tupels in ein Dictionary durchführt, das anstatt des Textes den Dateinamen enthält.

Nun wird das Modell auf die Daten angewandt. Dabei gibt die Funktion `model.predict(text)` ein Tupel zurück, welches den Text, das zugeordnete Label sowie die Konfidenz, also die Sicherheit des Ergebnisses, enthält. Des Weiteren wird die Liste der Ergebnisse in einen pandas-Dataframe^[Mendel et al. 2021] umgewandelt.


```

1     for filename in filelist:
2         with open("Reports/" + filename, "r", encoding="utf8")
           ↪ as file:
3             res = model.predict(file.read())
4             dictionary = tuple_to_dictionary(res, filename)
5             results.append(dictionary)
6 df = pd.DataFrame(results,
           ↪ columns=["filename", "label", "confidence"])

```

Abbildung 25: Anwendung des Modells auf alle Texte.

Der letzte Schritt ist die Speicherung der Ergebnisse, z. B. in Form einer CSV-Datei.

```

1     df.to_csv("classificationresults.csv")

```

Abbildung 26: Speicherung der Ergebnisse.

3.4 Zusammenführung der Daten und Geocoding

Nun folgt die Zusammenführung der Daten des EntityRecognizers und Text-Classifiers.

Dafür werden als Erstes die beiden CSV-Dateien aus den vorherigen Schritten geladen und die `id`-Spalten gelöscht, da nur die `filename` Spalte für die Zuordnung relevant ist.

```

1     nerdata = pd.read_csv("resultsspacy.csv")
2     classdata = pd.read_csv("classificationresults.csv")
3     nerdata.drop(columns=["id"], axis=1, inplace=True)
4     classdata.drop(columns=["id"], axis=1, inplace=True)

```

Abbildung 27: Einlesen der pandas-Dataframes aus CSV-Dateien.

Als Nächstes werden die beiden Dataframes mithilfe der gemeinsamen

Spalte `filename` zusammengeführt.

```
1 joineddf = pd.merge(nerdata, classdata, on="filename")
```

Abbildung 28: Zusammenführung der beiden Dataframes mithilfe der Spalte `filename`.

Der darauffolgende Schritt ist die Geokodierung der Daten, d. h. den Adressfeldern werden Koordinaten zugeordnet. Zurückgegeben wird ein `SpatialDataframe`, welcher eine zusätzliche Spalte namens `"SHAPE"` enthält. Diese beinhaltet Dictionaries mit x- und y-Koordinaten. Der hier angegebene Geocoder ist limitiert auf ein Rechteck, welches Sachsen einschließt, sodass alle Koordinaten in oder zumindest in der Nähe von Sachsen liegen.

```
1 sdf = GeoAccessor.from_df(joineddf, address_column="Place",
    → geocoder=Geocoder.fromitem(Item(gis,
    → "511d49713dbd4327a3dd7c31ecbc1226")))
```

Abbildung 29: Geokodierung der Adressen mithilfe eines Geocoders für die Region Sachsen.

Folgend ein Beispieldatensatz des `SpatialDataframes`:

| ID | TEXT | Filename | Crime | |
|------------|--|-----------------------|--------------------------------------|---------------------------------|
| 99 | Moped-Fahrer bei Unfall schwer verletzt Kreba-Neudorf, Bautzener Straße 17.12.2020, 09.50 Uhr Ein Moped-Fahrer hat sich am Donnerstagmorgen bei einem Unfall an der Bautzener Straße in Kreba-Neudorf schwer verletzt. Der 65-Jährige befuhrt die S 121 nach Mücka und kam aus bislang ungeklärter Ursache nach links von der Fahrbahn ab. Der Mann kam zu Fall. Ein Rettungswagen brachte ihn in ein Krankenhaus. Am Moped entstand ein Sachschaden von ungefähr 3.000 Euro. (fs) | 0128.txt | Unfall,von der Fahrbahn ab | |
| Criminal | Damage | Date & Time | Injured | Place |
| 65-Jährige | 3.000 Euro | 17.12.2020, 09.50 Uhr | schwer verletzt | Kreba-Neudorf, Bautzener Straße |
| Victim | Weapon | label | confidence | SHAPE |
| | Accident | 0.974445 | { "x": 14.6852545, "y": 51.3441152 } | |

Abbildung 30: Beispieldatensatz des `SpatialDataframes`.

Der letzte Schritt ist nun die Bereinigung der Daten und der Upload

nach ArcGIS Online. Hierfür werden alle Sonderzeichen usw. aus den Spaltennamen entfernt und alle Datensätze, welche keine Koordinaten enthalten, gelöscht. Anschließend kann der SpatialDataframe mit der Methode `spatialdataframe.to_featurelayer("name")` veröffentlicht werden.

```
1      # Bereinigung der Spaltennamen (Ersetzung von Leerzeichen  
      ↳ durch Unterstriche) usw.  
2      sdf.spatial.sanitize_column_names(inplace=True)  
3      # Löschung aller Datensätze, deren SHAPE Spalte null ist, da  
4      # diese nicht auf einer Karte dargestellt werden können  
5      newdf = sdf[sdf.SHAPE.notnull()]  
6      # Veröffentlichung nach ArcGIS Online  
7      newdf.spatial.to_featurelayer("Polizei")
```

Abbildung 31: Bereinigung der Daten und Veröffentlichung nach ArcGIS Online.

3.5 Integration der Modellergebnisse ins GIS

Nach dem Upload des SpatialDataframes als Feature-Layer nach ArcGIS Online kann die Auswertung der Daten vorgenommen werden. Dafür wurde in ArcGIS Online eine Karte erstellt, der Feature-Layer importiert und das Attribut `label` ausgewählt, um die Berichte nach Kategorie der Straftat farblich zu gruppieren.

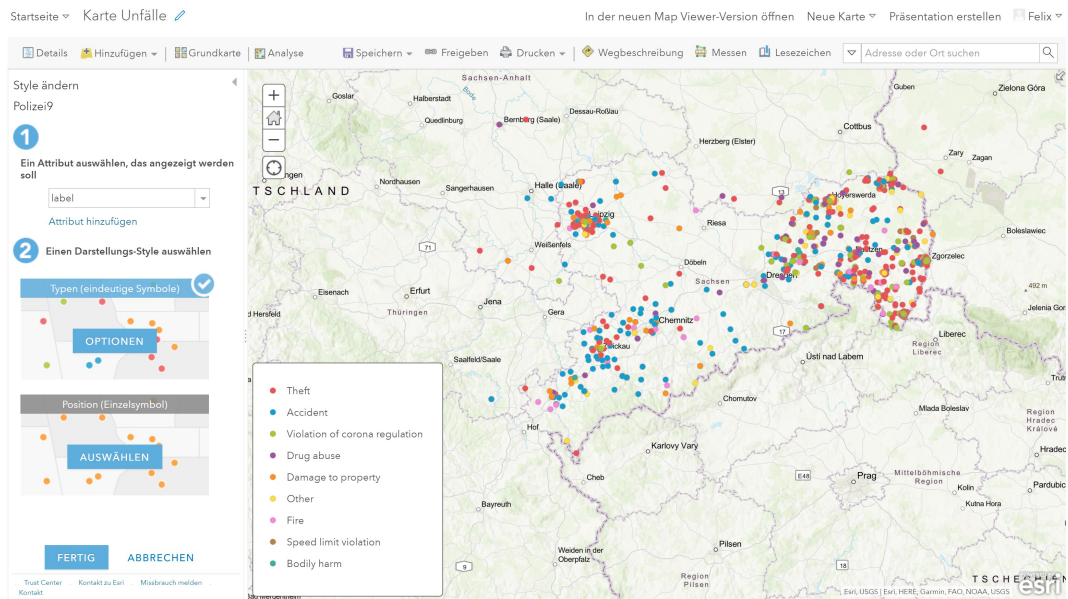


Abbildung 32: Auswahl eines Attributs für die Darstellung der Punkte des Feature-Layers.

Zusätzlich können auch komplexere Abfragen gemacht werden: So wurde hier ein Script in der Esri Programmiersprache Arcade erstellt, welches zurückgibt, ob es sich bei einem Bericht um einen Unfall handelt oder nicht.

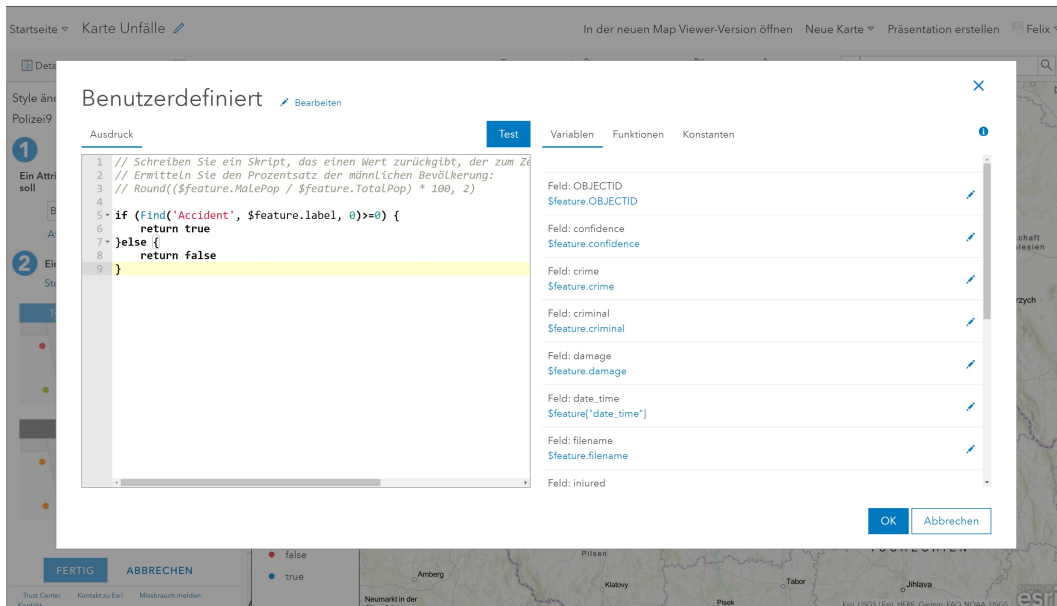


Abbildung 33: Script in der Esri Sprache Arcade.

Nun müssen die Datensätze, für welche das Script **false** zurückgibt, aus-

geblendet werden.

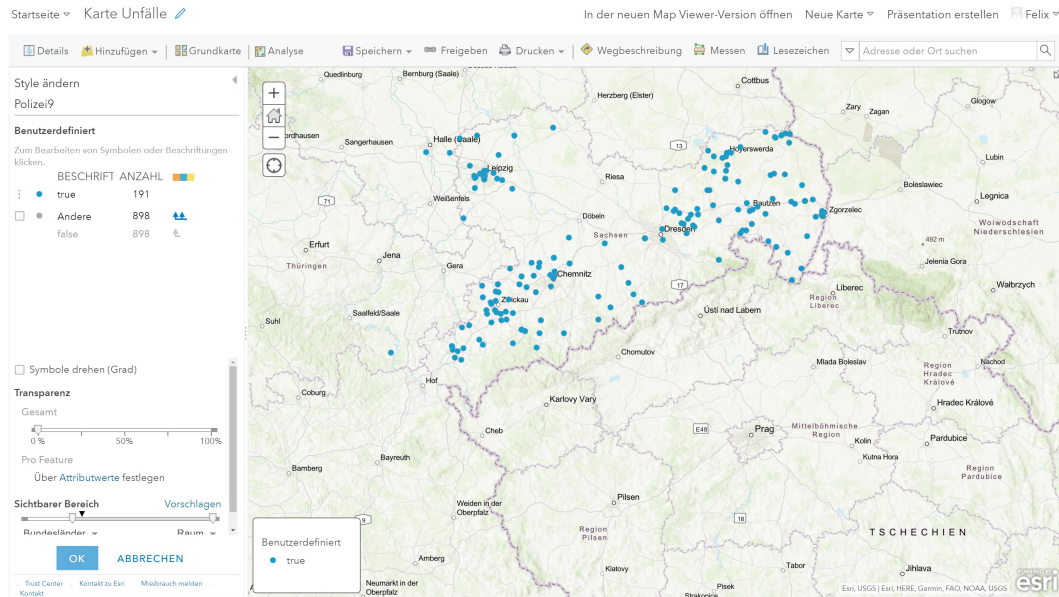


Abbildung 34: Verbergen der Punkte der Kategorie **false**.

Daraufhin kann aus diesen einzelnen Karten ein interaktives Dashboard erstellt werden. Dort befindet sich links eine Übersicht über Statistiken wie die Anzahl der Diebstähle, der Unfälle, der Fahrradunfälle und der Vorfälle, bei welchen Alkohol im Spiel war.

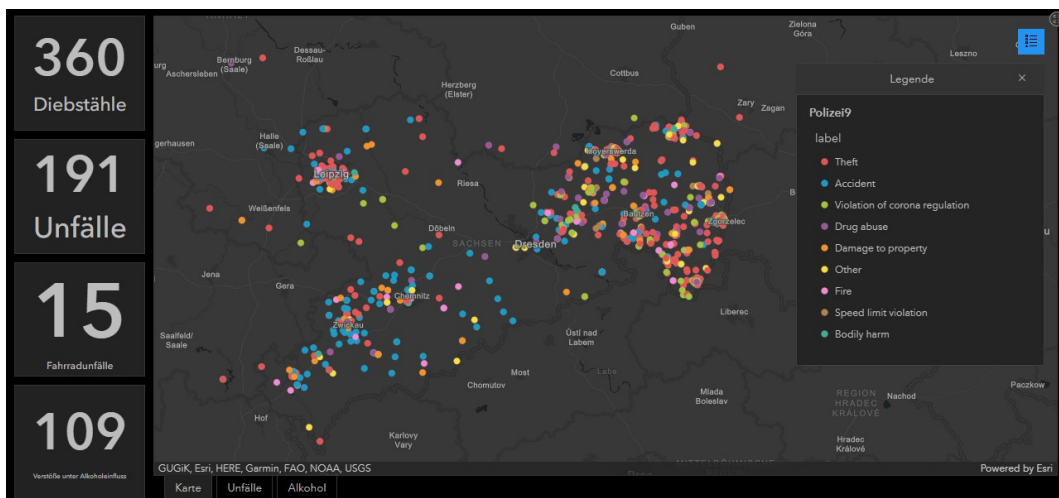


Abbildung 35: Entstandenes Dashboard.

3.6 Feedback-Loop

Zur schnelleren Erstellung von Trainingsdaten wurde ein Feedback-Loop entwickelt, mit welchem es möglich wird, den Output des EntityRecognizer-Modells als weitere Trainingsdaten zu nutzen. Dafür wird mithilfe eines Python-Skripts der Output des EntityRecognizers in JSONL-Annotations umgewandelt. Diese können anschließend in doccano importiert und von Menschen überprüft werden. Darauffolgend werden diese Daten aus doccano exportiert und als Trainingsdaten für das Modell genutzt. Bei dem Output des Modells handelt es sich in diesem Fall um eine CSV-Datei, welche nach folgendem Format aufgebaut ist:

| id | Text | Filename | Place | Date & Time | Weapon | Damage | Criminal | Injured | Victim | Crime |
|----|------|----------|-------|-------------|--------|--------|----------|---------|--------|-------|
|----|------|----------|-------|-------------|--------|--------|----------|---------|--------|-------|

Abbildung 36: Aufbau der CSV-Ausgabe des EntityRecognizers.

Hierbei enthalten die Spalten vier bis elf durch Kommata getrennte Strings, bei welchen es sich um die Textausschnitte handelt, die das Modell der jeweiligen Kategorie zugeordnet hat. Diese müssen nun geteilt und im Text gesucht werden. Anschließend kann über die Start- und Endposition (Zeichenanzahl) ein String erstellt werden, der dem Format JSON-Lines entspricht, welches sowohl als In- als auch als Output für doccano verwendet werden kann.

```

1  import csv, os
2  results = ""
3  # Öffnen der Datei mit AI Model Outputs
4  with open("results2.csv", "r", encoding="utf8") as file:
5      reader = csv.reader(file)
6      labels = []
7      for row in reader:
8          col_count = len(row)
9          # für die erste Zeile
10         if reader.line_num == 1:
11             # die ersten drei Spalten enthalten keine Daten
12             for i in range(3, col_count):
13                 labels.append(row[i])
14             # für alle anderen Zeilen
15         else:
16             text = row[1]
17             text = text.replace("\n", " ")
18             # als ID eignet sich die erste Spalte nicht, da
19             # für jede Datei mehrere Spalten vom AI Model
20             ↪ erzeugt
21             # werden können. Daher muss die ID aus dem
22             ↪ Dateinamen
23             # verwendet werden.
24             id = int(row[2].split(".")[0])
25             jsonlabels = []
26             # die ersten drei Spalten enthalten keine Daten
27             for i in range(3, col_count):
28                 # jedes Feld kann mehrere, durch Kommata
29                 ↪ getrennte, Werte enthalten
30                 parts = row[i].split(",")
31                 for part in parts:
32                     # Finden der Startposition des gesuchten
33                     ↪ Abschnitts im Text
34                     start = text.find(part)
35                     # Berechnen der Endposition
36                     end = start + len(part)
37                     # Prüfen, dass die Länge nicht 0 ist
38                     if not (start == 0 and end == 0) and not
39                     ↪ len(part) == 0 and not start == end:
40                         problem = False
41                         # Für jedes der bereits existieren
42                         ↪ JSON-Labels:
43                         for jsonlabel in jsonlabels:
44                             start1 = jsonlabel[0]
45                             end1 = jsonlabel[1]

```

```

40         # Finden der ersten Startposition
41         minstart = min(start1, start)
42         # Falls sich Labels überschneiden,
         ↳ darf jeweils nur das Erste
         ↳ hinzugefügt werden.
43         # Dies passiert realistisch jedoch
         ↳ nur aller etwa 20 Texte.
44         if start == minstart and end - start
         ↳ >= start1 - minstart:
45             problem = True
46         if start1 == minstart and end1 -
         ↳ start1 >= start - minstart:
47             problem = True
48         if not problem:
49             jsonlabels.append([abs(start),
         ↳ abs(end), labels[i-3]])
50         # Anhängen der neuen Zeile mit JSON-Labels an den
         ↳ results String
51         line = ('{"id": ' + str(id) + ', "text": "' +
         ↳ text.replace('"', '"') + '", "labels": '
52         + str(jsonlabels).replace('"', '"') + '}')
53         results += line + "\n"
54     with open("newannotations.json", "w", encoding="utf8") as file:
55         file.write(results)

```

Abbildung 37: Feedback-Loop.

4 Ergebnisse

4.1 Vergleich verschiedener Backbones

Die EntityRecognizer-Netze wurden mithilfe verschiedener Backbones trainiert und jeweils deren benötigte Dauer in Sekunden zum Finden der optimalen Learning-Rate, Trainieren des Modells sowie der Anwendung auf etwa 1.700 Beispieldaten gemessen. Des Weiteren wurde die endgültige Genauigkeit des Modells in Prozent erfasst.

| Modell | Modellname |
|--------|--|
| 1 | spaCy ^{[HuggingFace, 2020][ExplosionAI, 2021]} |
| 2 | bert-base-german-cased ^{[HuggingFace, 2020][deepset, 2021]} |
| 3 | bert-base-german-dbmdz-cased ^{[HuggingFace, 2020][Schweter und Baiter, 2021]} |
| 4 | xlm-mlm-ende-1024 ^[HuggingFace, 2020] |
| 5 | xlm-clm-ende-1024 ^[HuggingFace, 2020] |
| 6 | distilbert-base-german-cased ^{[HuggingFace, 2020][Sanh et al. 2019]} |

| Modell | lr_find() Dauer | fit() Dauer | extract_entities() Dauer | Genauigkeit |
|--------|-----------------|-------------|--------------------------|-------------|
| 1 | 12,7 s | 738,7 s | 26,4 s | 100 % |
| 2 | 89,8 s | 6535,1 s | 311,5 s | 83 % |
| 3 | 105,5 s | 6347,3 s | 302,1 s | 84 % |
| 4 | 67,2 s | 4792,8 s | 225,9 s | 88 % |
| 5 | 82,6 s | 6488,3 s | 324,5 s | 81 % |
| 6 | 48,8 s | 3110,3 s | 139,8 s | 81 % |

| Modell | lr_find() Dauer | fit() Dauer | extract_entities() Dauer | Genauigkeit |
|--------|-----------------|-------------|--------------------------|-------------|
| 1 | 12,04 % | 11,30 % | 8,14 % | 100 % |
| 2 | 85,12 % | 100 % | 95,99 % | 83 % |
| 3 | 100 % | 97,13 % | 93,10 % | 84 % |
| 4 | 63,70 % | 73,34 % | 69,62 % | 88 % |
| 5 | 78,29 % | 99,28 % | 100 % | 81 % |
| 6 | 46,26 % | 47,59 % | 43,08 % | 81 % |

Abbildung 38: Modellvergleich EntityRecognizer.

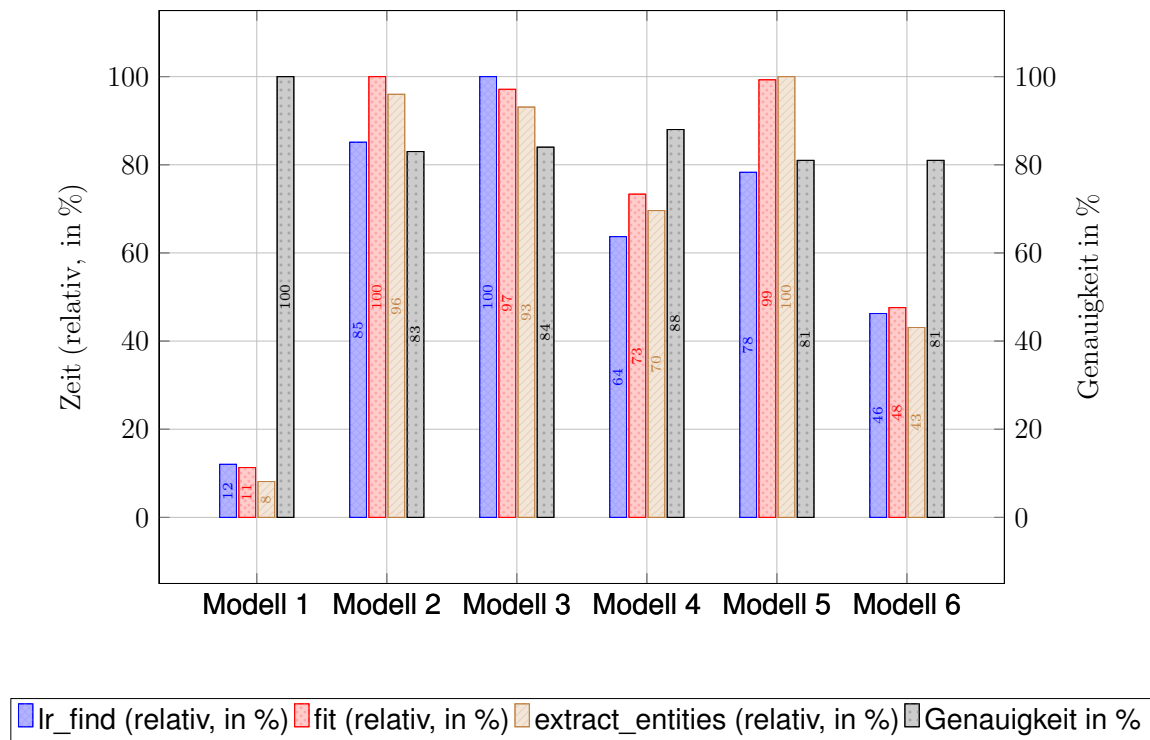


Abbildung 39: Relativer Modellvergleich.

Für diese Messwerte ist nur ein relativer Vergleich untereinander sinnvoll, da die Ergebnisse stark von der vorhandenen CPU-Leistung abhängen. Anhand von Abbildung 39 ist deutlich ersichtlich, dass es sich bei Modell 1 sowohl um das mit Abstand schnellste als auch um das genaueste handelt. Alternativ bietet Modell 4 die zweithöchste Genauigkeit, allerdings ist dieses Modell auch deutlich langsamer.

5 Fazit

5.1 Zusammenfassung des Vorgehens

Der erste Schritt war das Sammeln der Polizeiberichte, welcher im Falle der Polizei Sachsen teilweise manuell erfolgen musste. Daraufhin folgte das Erstellen der Trainingsdaten mithilfe von doccano. Anschließend konnten die Daten aus doccano exportiert und dann als JSONL an den EntityRecognizer bzw. TextClassifier übergeben werden. Nach dem Training wurden die jeweiligen Modelle gespeichert und auf die Berichte angewendet. Die exportierten CSV-Dateien wurden anschließend zusammengeführt und geokodiert. Nun folgten die Bereinigung und der Upload der Daten zu ArcGIS Online, wo diese zu Karten und einem interaktiven Dashboard verarbeitet wurden.

5.2 Erlangte Erkenntnisse

Durch diese Besondere Lernleistung wurden die Kenntnisse des Autors über die genauere Funktionsweise von KIs, Neuronalen Netzen und Natural Language Processing vertieft. Außerdem wurden tiefere Erkenntnisse über die Programmiersprache Python erlangt, welche nicht Bestandteil dieser Arbeit sind, wie beispielsweise die Funktionsweise von ThreadPools. Es ist tatsächlich bereits mit wenigen hundert Trainingsdaten möglich, in kurzer Zeit ein NLP-Modell zu trainieren, welches Ergebnisse mit einer hervorragenden Genauigkeit erzielt.

5.3 Ausblick

Die Spalte mit Datum und Uhrzeit kann mithilfe eines Sequence To Sequence NLP-Modells in eine standardisierte Form, z. B. nach ISO 8601, gebracht werden. Dadurch wären zusätzlich zu den räumlichen Analysen auch zeitliche Analysen möglich. Außerdem könnte man überprüfen, inwiefern sich dieses Modell auf Berichte aus anderen Bundesländern auswirkt, wenn keine neuen Trainingsdaten bereitgestellt werden.

6 Literaturverzeichnis

- deepset (2021). *Open Sourcing German BERT — Insights into pre-training BERT from scratch*. online. deepset. URL: <https://www.deepset.ai/german-bert> (besucht am 31.08.2021).
- Devlin, Jacob, Ming-Wei Chang, Kenton Lee et al. (2018). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. online. Google. URL: <https://arxiv.org/abs/1810.04805> (besucht am 10.02.2021).
- Devlin, Jacob, Ming-Wei Chang, Research Scientists et al. (Nov. 2018a). *Open Sourcing BERT: State-of-the-Art Pre-training for Natural Language Processing*. online. Google. URL: <https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html> (besucht am 03.05.2021).
- (Nov. 2018b). *Open Sourcing BERT: State-of-the-Art Pre-training for Natural Language Processing*. online. adaptiert. Google. URL: https://1.bp.blogspot.com/-RLAbr6kPNUo/W9is5FwUXmI/AAAAAAAAADeU/5y9466Zoyoc96vqLjbruLK8i_t8qEdHnQCLcBGAs/s1600/image3.png (besucht am 03.05.2021).
- Esri (2021a). *Named Entity Extraction Workflow with arcgis.learn*. online. Arcgis.com. URL: <https://developers.arcgis.com/python/guide/how-named-entity-recognition-works/> (besucht am 03.09.2021).
- (2021b). *What is GIS?* online. URL: <https://www.esri.com/en-us/what-is-gis/overview> (besucht am 05.09.2021).
- ExplosionAI (2021). *Available trained pipelines for German*. online. spaCy. URL: <https://spacy.io/models/de> (besucht am 31.08.2021).
- Hammond, Mark et al. (Mai 2021). *pywin32*. online. URL: <https://pypi.org/project/pywin32/> (besucht am 01.09.2021).

- Howard, Jeremy und Sebastian Ruder (2018). *Fine-tuned Language Models for Text Classification*. online. URL: <http://arxiv.org/abs/1801.06146> (besucht am 02.09.2021).
- HuggingFace (2020). *Pretrained models*. online. Huggingface.co. URL: https://huggingface.co/transformers/pretrained_models.html (besucht am 31.08.2021).
- Mendel, Brock et al. (2021). *pandas: powerful Python data analysis toolkit*. online. pandas. URL: <https://pypi.org/project/pandas/> (besucht am 03.09.2021).
- Nakayama, Hiroki (2019). *Welcome to doccano — Text Annotation for Humans*. online. doccano. URL: <https://doccano.github.io/doccano/> (besucht am 31.08.2021).
- pandas Entwicklerteam (2021). *pandas.DataFrame*. online. pandas. URL: <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html> (besucht am 18.11.2021).
- Peters, Matthew E. et al. (2018). *Deep contextualized word representations*. online. URL: <https://aclanthology.org/N18-1202.pdf> (besucht am 02.09.2021).
- Roldós, Inés (März 2020). *Named Entity Recognition: Concept, Tools and Tutorial*. online. URL: <https://monkeylearn.com/blog/named-entity-recognition/> (besucht am 01.09.2021).
- Sachsen, Polizei (Aug. 2021). *Polizei Sachsen - Polizeidirektion Dresden - Quad gestohlen, u.a. Meldungen*. online. sachsen.de. URL: https://www.polizei.sachsen.de/de/MI_2021_82801.htm (besucht am 03.08.2021).
- Sanh, Victor et al. (2019). *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*. online, archived. Huggingface.co. URL: <https://web.archive.org/web/20201112022945/https://github.com>

com/huggingface/transformers/tree/master/examples/distillation
(besucht am 31.08.2021).

Schweter, Stefan und Johannes Baiter (2021). *dbmdz BERT models*. online. dbmdz. URL: <https://github.com/dbmdz/berts> (besucht am 31.08.2021).

Vasani, Dipam (März 2019). *How do pre-trained models work?* online. towardsdatascience. URL: <https://towardsdatascience.com/how-do-pretrained-models-work-11fe2f64eaa2> (besucht am 18.11.2021).

Ward, Ian (Aug. 2021). *JSON Lines*. online. URL: <https://jsonlines.org/> (besucht am 03.08.2021).

Wikipedia-Autoren (Apr. 2021a). *Artificial intelligence*. online. Wikipedia. URL: https://en.wikipedia.org/wiki/Artificial_intelligence (besucht am 02.05.2021).

– (Mai 2021b). *Artificial neural network*. online. Wikipedia. URL: https://en.wikipedia.org/wiki/Artificial_neural_network (besucht am 02.05.2021).

– (Mai 2021c). *Deep learning*. online. Wikipedia. URL: https://en.wikipedia.org/wiki/Deep_learning (besucht am 02.05.2021).

– (Juni 2021d). *ISO 6709*. online. Wikipedia. URL: https://en.wikipedia.org/wiki/ISO_6709 (besucht am 05.09.2021).

– (März 2021e). *Künstliche Intelligenz*. online. Wikipedia. URL: https://de.wikipedia.org/wiki/K%C3%BCnstliche_Intelligenz (besucht am 14.03.2021).

7 Abkürzungsverzeichnis

BERT Bidirectional Encoder Representations from Transformers

GPT Generative Pre-Training

KI Künstliche Intelligenz

LR Learning-Rate

NER Named Entity Recognition bzw. Named Entity Recognizer

STS Sequence To Sequence

TC Text Classification

NLP Natural Language Processing

8 Verzeichnis der Fremd- und Fachworterklärungen

JSONL / JSON Lines JSON-ähnliches Dateiformat, bei dem jede Zeile als einzelnes JSON-Objekt eingelesen wird^[Ward, 2021].

doccano Das Open-Source-Werkzeug doccano dient zur Textannotation. Es bietet Annotationsfunktionen für Text Classification, Named Entity Recognition und Sequence To Sequence^[Nakayama, 2019].

Learning-Rate Ein Tuning-Parameter, welcher die Schrittgröße jeder Iteration bestimmt, während eine Annäherung auf ein Minimum einer Verlustfunktion durchgeführt wird. Die Learning-Rate spiegelt somit auch die Lerngeschwindigkeit des Modells wider^[Esri, 2021a].

Geokodierung Umwandlung von Adressen in Textform in Koordinaten nach Standard ISO 6709 (dezimale Gradzahlen)^[Wikipedia-Autoren, 2021d].

9 Verzeichnis der Abbildungen, Tafeln und Tabellen

| | | |
|----|---|----|
| 1 | Vergleich des Aufbaus von BERT, einem bidirektionalen, und GPT, einem unidirektionalen Netz ^[Devlin, Chang, Scientists et al. 2018b] . | 6 |
| 2 | Beispiel für unscrapbares HTML von der Website der Polizei Sachsen ^[Sachsen, 2021] | 10 |
| 3 | Script zum Speichern der Daten der Polizei-Website. | 11 |
| 4 | Erstellung einer mit doccano importierbaren Datei. | 12 |
| 5 | Erstellte Labels in doccano. | 12 |
| 6 | Beispiel eines annotierten Berichts. | 13 |
| 7 | Beispielzeile aus dem exportierten Datensatz. | 14 |
| 8 | Einlesen der Daten aus der JSONL-Datei. | 15 |
| 9 | Erstellung eines EntityRecognizer. | 15 |
| 10 | Finden der optimalen Learning-Rate. | 15 |
| 11 | Trainieren des Netzes. | 16 |
| 12 | Speichern des Modells. | 16 |
| 13 | Extrahieren der Informationen aus Dateien im Reports-Ordner. | 16 |
| 14 | Speicherung der Ergebnisse. | 17 |

| | | |
|----|--|----|
| 15 | Einlesen der Trainingsdaten aus der TXT-Datei. | 17 |
| 16 | Erstellung eines TextClassifiers. | 18 |
| 17 | Finden der optimalen Learning-Rate. | 18 |
| 18 | Training des Modells. | 18 |
| 19 | „Auftauen“ des Modells. | 18 |
| 20 | Training des Modells (Feinjustierung). | 19 |
| 21 | Speichern des Modells. | 19 |
| 22 | Anwendung des Modells auf Beispieldatensatz. | 19 |
| 23 | Laden der Inhalte aus den TXT-Dateien. | 20 |
| 24 | Erstellung einer Liste für die Ergebnisse sowie einer Methode, welche die Umwandlung eines Tupels in ein Dictionary durchführt, das anstatt des Textes den Dateinamen enthält. . | 20 |
| 25 | Anwendung des Modells auf alle Texte. | 21 |
| 26 | Speicherung der Ergebnisse. | 21 |
| 27 | Einlesen der pandas-Dataframes aus CSV-Dateien. | 21 |
| 28 | Zusammenführung der beiden Dataframes mithilfe der Spalte filename. | 22 |
| 29 | Geokodierung der Adressen mithilfe eines Geocoders für die Region Sachsen. | 22 |

| | | |
|----|--|----|
| 30 | Beispieldatensatz des SpatialDataframes. | 22 |
| 31 | Bereinigung der Daten und Veröffentlichung nach ArcGIS Online. | 23 |
| 32 | Auswahl eines Attributs für die Darstellung der Punkte des Feature-Layers. | 24 |
| 33 | Script in der Esri Sprache Arcade. | 24 |
| 34 | Verbergen der Punkte der Kategorie <code>false</code> | 25 |
| 35 | Entstandenes Dashboard. | 25 |
| 36 | Aufbau der CSV-Ausgabe des EntityRecognizers. | 26 |
| 37 | Feedback-Loop. | 28 |
| 38 | Modellvergleich EntityRecognizer. | 29 |
| 39 | Relativer Modellvergleich. | 30 |

Danksagung

Der besondere Dank des Autors gilt dem externen Betreuer Thomas Paschke, welcher jederzeit für Fragen zur Verfügung stand und bei der Suche nach Lösungen für Probleme geholfen hat. Des Weiteren möchte der Autor dem internen Betreuer Herrn Karl für die Erläuterungen zum Verfassen einer wissenschaftlichen Arbeit seinen Dank aussprechen, vor allem für die Hinweise zur Zitierweise. Auch möchte der Autor Herrn Hausmann für die Herstellung des Kontakts zu Herrn Paschke sowie für die schnelle Hilfe zum Erstellen eines Geocoders für Sachsen, danken. Zusätzliche dankt der Autor Esri Deutschland für die Betreuung der Besonderen Lernleistung sowie für die Bereitstellung eines ArcGIS Online-Zugangs, einer ArcGIS Pro-Lizenz und eines Office 365 Accounts.

Selbstständigkeitserklärung

Ich versichere, dass ich die Arbeit selbstständig angefertigt, nur die angegebenen Hilfsmittel benutzt und alle Stellen, die dem Wortlaut und dem Sinn nach anderen Werken entnommen sind, durch Angabe der Quellen als Entlehnung kenntlich gemacht habe.

Mit der schulinternen Verwendung der Arbeit bin ich einverstanden.

Leipzig, 27. Januar 2022