ETIM Klassifikation mit ML-Approach

Vorhersage der ETIM-Klasse basierend auf Daten aus der Geberit Datenbank

Studienarbeit FS 2021

Version: 1.0

Autoren:	Etienne Baumgartner, Nathanael Gall
Betreuer:	Prof. Oliver Augenstein
Industriepartner:	Geberit AG, Jona
Experte:	Nico Schmid
Themengebiet:	Machine Learning
Studiengang:	Informatik

Abstract

Situation

Das ETIM Klassifikationsmodel setzt sich auf dem internationalen Markt immer mehr durch. Die ETIM Klassifikation vereinfacht einerseits den Datenaustausch zwischen Händler und Hersteller und andererseits die Klassifizierung der Produkte. Die Geberit AG nutzt diesen Standard für ihren Produktekatalog und die Klassifizierung wird manuell durch einen Sachbearbeiter vorgenommen.

Ziel

Das Ziel der Arbeit ist die Entwicklung und der Vergleich von zwei Verfahren zur automatischen Bestimmung der ETIM Klasse auf der Basis von maschinellem Lernen

Methode / Vorgehen

Der Erfolg der Arbeit beruht auf der richtigen Verwertung der von Geberit zur Verfügung gestellten Produktdaten. Auf Grund der Analyse wurden zwei Ansätze entwickelt, um die ETIM Klassifizierung zu erlernen.

One-Hot-Encoding	Die Daten werden für das Trainieren auf einem Deep Neural Network One-Hot-Encoded. Um die Anzahl Dimensionen des Netzwerks möglichst klein zu halten, wird anhand des Bayes Errors die optimale Spaltenkombination ermittelt. Auf dieser Auswahl wird ausserdem ein Lookup Table erstellt, um eindeutige Datensätze direkt zu klassifizieren.
Textembedding	Viele der Spalten weisen einen grossen Anteil an deutschem Text auf. Es wird mit der fastText Library ein Textembedding trainiert, welches die Artikel den jeweiligen ETIM Klassen zuordnet.

Wesentliche Ergebnisse

Nach der Optimierung dieser zwei Ansätze stellt sich heraus, dass der textbasierte fastText Algorithmus genauere Resultate liefert. Mit dem fastText Model wird auf dem vorab abgetrennten finalen Testset eine Top-1-Accuracy von 0.961 und eine Top-3-Accuracy von 0.982 erreicht.

Die höhere Accuracy ist auf die bessere Verwertungder Spaltentexte zurückzuführen, weil dasTextembedding die Nähe zweier verschiedener Werteabbilden kann, während beim One-Hot-Encoding alleunterschiedlichen Werte äquidistant sind.

Empfehlungen

Die Empfehlung ist, die drei wahrscheinlichsten Klassen des fastText Top-3-Algorithmus zu präsentieren und die finale Entscheidung einem Benutzer zu überlassen

Schlüsselwörter Machine Learning, ETIM, Classification, fastText, Neural Network

Inhaltsverzeichnis

1 Einleitung

Auftrag

Diese Semesterarbeit befasst sich mit einem Klassifizierungsproblem, dass anhand maschinellen Lernens gelöst werden soll.

Die Ausgangslage stellt eine Ansammlung von ungefähr 25'000 Datensätzen dar. Die Aufgabe besteht nun darin verschiedene Algorithmen aufzubauen und eine möglichst gute positive Klassifizierungsrate auf den Daten zu erhalten. Es werden keine Beschränkungen bezüglich der Algorithmen-Wahl gesetzt.

Motivation

Das Interesse von Geberit liegt in der maschinellen Klassifizierung ihrer Produkte. Genauer gesagt geht es um die Zuordnung ihrer Produktepalette zum weltweit genutzten ETIM Standard.
Für diese Zuordnung wurde bisher eine manuelle Zuteilung anhand interner Produktdaten durchgeführt. Die Grundlage der manuellen Zuteilungen wurde in Form der 25'000 Datensätze zur Verfügung gestellt.
Geberit möchte ihre Möglichkeiten für eine optimierte, automatische Klassifizierung prüfen und übergibt dieses Anliegen in Form einer Semesterarbeit an die Hochschule OST. Der Auftrag setzt sich aus der Analyse der Daten, Aufbereitung der nutzbaren Informationen und dem Vergleich der möglichen Ansätze zusammen.

Ziel

Das Ziel der Arbeit ist die konzeptuelle Untersuchung der Möglichkeiten, den bisher manuellen Klassifizierungsprozess von Produkten zur respektiven ETIM Standard Klasse mit Hilfe von maschinellem Lernen zu erleichtern.
Hierbei geht es vor allem darum aufzuzeigen, dass ein maschineller Ansatz zur Klassifizierung einen Mehrwert gegenüber der manuellen Bearbeitung bieten kann. Damit lässt sich eine Zuordnung von neuen Produkten schneller und genauer durchführen. Dabei sollen verschiedene Ansätze untersucht und verglichen werden, um eine passende Lösung zu präsentieren.
Optimalerweise wird eine potenzielle Grundlage für eine spätere Software-Lösung in Form eines Prototyps bereitgestellt.

Anwendung der Resultate

Die Voraussetzung dieser Arbeit ist nicht die perfekte Zuordnung der Daten. Ein guter Algorithmus soll eine Einschränkung der möglichen Zuweisungen erlauben und unter anderem potenzielle ETIM Klassifizierungen vorschlagen. Der empfohlene Algorithmus soll einen menschlichen Sachbearbeiter in der Zuordnung der ETIM Klassen weitgehend durch diese vorgeschlagenen Klassifizierungen unterstützen.

Vorgehen

Zu Beginn werden die Daten von Geberit untersucht. Durch diese Analyse werden wichtige Informationen und grundlegende Charakteristiken der Daten aufgedeckt. Auf der Basis dieser Eigenschaften wird die Datenaufbereitung lanciert. Dabei werden die Daten so weit präpariert, dass ein optimales Lernen möglich ist und keine unerwünschten Nebeneffekte, zum Beispiel durch Ausreisser, auftreten. Die aufgearbeiteten Daten werden den jeweiligen Algorithmen eingespeist und die Resultate anschliessend verglichen. Auf Grund der Resultate wird zum Abschluss eine Empfehlung abgegeben und falls zeitlich möglich ein Prototyp in Form einer Anwendung bereitgestellt.

Zusammenfassend stellt sich diese Arbeit somit wie folgt zusammen:

1. Datenanalyse
2. Datenaufbereitung
4. Algorithmen-Implementation
5. Resultats Vergleich
6. Fazit und Empfehlung

2 Definitionen

Allgemein Bevor mit der eigentlichen Bearbeitung des Themas begonnen werden kann, wird anbei der ETIM Standard erklärt.

2.1 ETIM

Definition «ETIM ist ein zweistufiges Klassifikationsmodell, das Produktdaten aus dem Fachbereich Elektrotechnik strukturiert erfasst und einen standardisierten elektronischen Datenaustausch von Produktdaten zwischen Herstellern und Handel ermöglicht.» [1]

Zusammensetzung Einem Produkt wird eine eindeutige Artikelklasse zugeordnet. Diese Artikelklasse ist wiederum genau einer Artikelgruppe unterstellt. Jede Artikelklasse wird durch ein eindeutiges Set an Merkmalen wie *Name*, *Typ*, *Einheit*, usw., identifiziert.

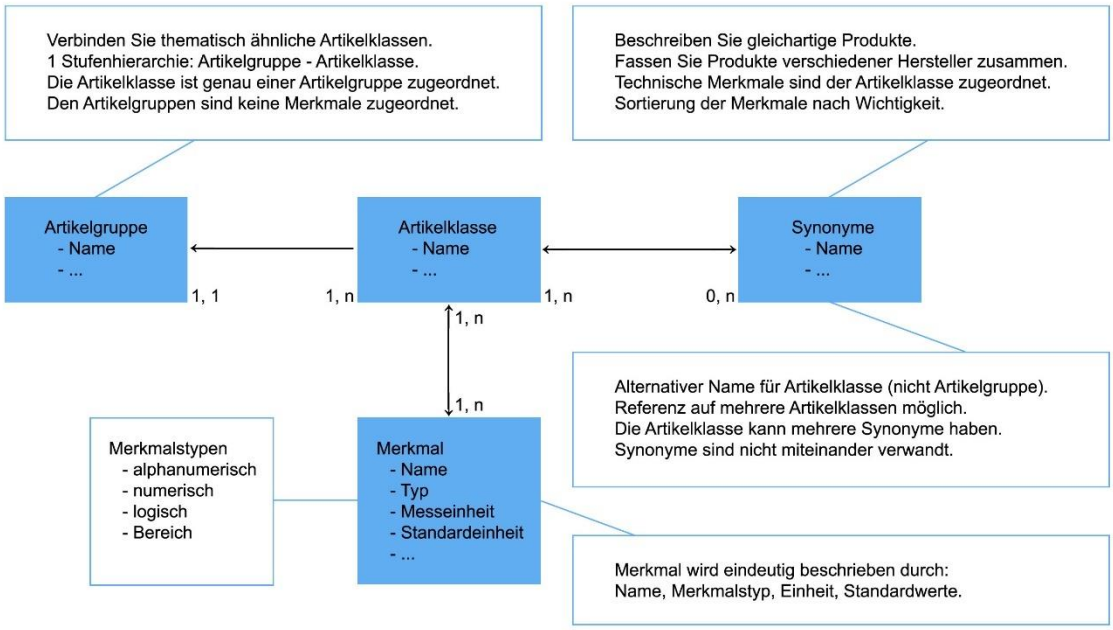

Abbildung 1: ETIM Zusammensetzung, Quelle https://www.etim.ch/de/klassifizierung/modell-informationen

ETIM Klasse In der Semesterarbeit ist nur die ETIM Klasse von Bedeutung.
Die ETIM Klasse vereinigt somit ähnliche Produkte verschiedener Hersteller zu einer Gruppierung. Die Produkte einer Klasse weisen ähnliche Spezifikationen auf und können anhand dieser auch klassifiziert werden.

ETIM Beschreibung Jedes Produkt hat neben der ETIM Klasse eine Beschreibung.

Format Das Format der ETIM Klasse ist immer gleich: EC + 6 Zeichen
Das Format der ETIM Beschreibung besteht aus maximal 80 Zeichen und ist textbasiert.

Relevanz

Die Daten von Geberit enthalten jeweils die ETIM Klasse und die dazugehörige ETIM Beschreibung. Diese liegen in einem 1:1 Verhältnis und daher ist nur die ETIM Klasse relevant.

Nutzen der Klassifizierung

Sobald einem Produkt eine ETIM Artikelklasse zugeordnet werden kann, können die Merkmale für den ETIM Standard automatisch aus der Produktdatenbank von Geberit extrahiert werden. Diese Zuordnung erfolgt momentan jedoch noch manuell und ist mühselig. Wenn sich ein maschineller Ansatz zur Klassifizierung als plausibel herausstellt, kann auch der Export des Produktkatalogs von Geberit viel effizienter durchgeführt werden.

3 Technologien und Umgebung

Sprache

Als Programmiersprache wurde Python gewählt.

Python hat eine reiche Ansammlung an Frameworks und Libraries, die auf Datenanalysen und maschinelles Lernen ausgelegt sind.
Ausserdem gibt Tools, um einfach und schnell Desktopanwendungen zu erstellen.

Python Pakete

Anbei ist eine Liste der Python Pakete aufgeteilt nach Bereich:

Bereich	Pakete
Analyse	pandas, matplotlib, tabulate, itertools
Aufbereitung	pandas, numpy, sklearn
Algorithmen	pandas, numpy, sklearn, tensorflow,pickle, joblib, fasttext
Prototyp	pandas, numpy, tkinter, pandastable, erstellte Algorithmen.

Genauere Angaben liegen in den jeweiligen Notebooks des Abgabeordners vor.

Technologien

Der gesamte Code für die Analyse, die Aufbereitung und den Aufbau der Algorithmen wurde in Jupyter Notebooks verfasst.

Der Prototyp ist eine Python Anwendung.

4 Datenanalyse

4.1 Ausgangslage

Allgemein

Die Daten wurden als CSV-Datei zur Verfügung gestellt und bestehen aus Informationen bezüglich 25'826 Artikel in der Geberit Datenbank. Jeder Datensatz hat die folgenden Informationen:

Datensätze

Spaltenname	Form	Type	Beschreibung
article_id	ART_{Nummer}	Text/Integer	
name	Combination of numbers and text	Text/Integer	
aricle_number	Nummer vom <Name>	Integer(.)	
etim_description	Beschreibung des Nutzungsraums	Text	
etim_id	EC{Nummer} 6 Digits	Text/Integer	
product_hierarchie	xx..yyy..zzzz + Textbeschreibung	Integer(..)/Text	
short_text	Text (+Dimension)	Text	
product_name	{Unternehmen} + Text	Text	
path	{Text}\{Text}\{Text}\...	Text(\)	
system_id	SYS_{Nummer} 6 Digits	Text/Integer	
system_name	Text	Text	2. Pfadtext
category_id	CAT_{Nummer} 6 Digits	Text/Integer	
category_name	Text	Text	3. Pfadtext
family_id	FAM_{Nummer} 6+ Digits	Text/Integer	
family_name	Text	Text	4. Pfadtext
group_id	GRP_{Nummer} 6+ Digits	Text/Integer	
group_name	Text	Text	5. Pfadtext
series_id	SER_{Nummer} 6+ Digits	Text/Integer	
series_name	Text	Text	6. Pfadtext
type_id	TYP_{Nummer} 6+ Digits	Text/Integer	
type_name	Text	Text	

Tabelle 1: Spaltenbeschreibung

Einfachheitshalber werden die Spaltennamen für die weitere Bearbeitung angepasst. Diese ist ein Schritt, der in der Datenaufbereitung durchgeführt wird, jedoch bereits jetzt angewendet wird.

4.2 Beobachtungen

Struktur Geberit

Während der ETIM Standard[1] auf zweistufiger Klassifizierung beruht, scheint Geberit grob einer sechsstufigen Struktur zu folgen. Artikel werden der Reihe nach einem System, einer Kategorie, einer Familie, einer Gruppe, einer Serie und zuletzt einem Typ zugeordnet. Diese Zuordnung korreliert mit der ETIM Klassifizierung, es gibt jedoch kein direktes Mapping.

ETIM Klassen

Die Produkte sind in 260 der 6'382 ETIM Klasseneingeteilt oder haben die Bezeichnung `'EC000000',` welche keiner Klasse des ETIM Standards entspricht. Die Aufteilung in ETIM Klassen ist annähernd Pareto verteilt, fällt jedoch stärker ab.

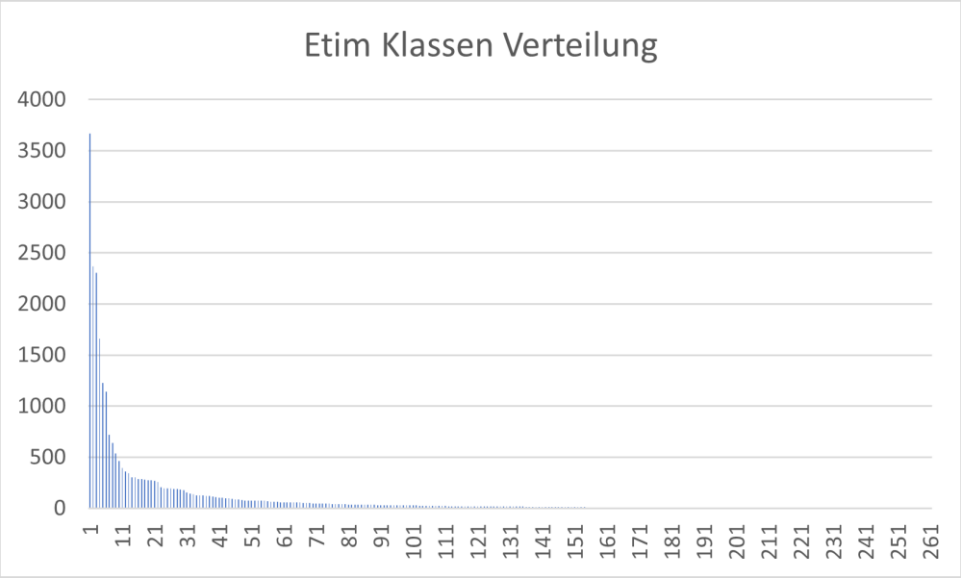

Abbildung 2: ETIM Verteilung

[1] https://prod.etim-international.com, Zugriff: 01.06.2021

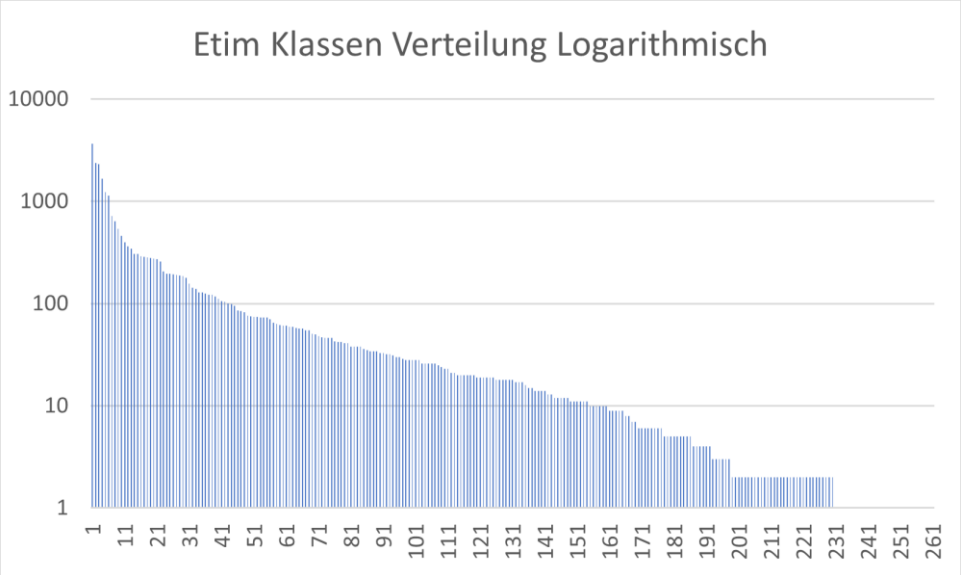

Abbildung 3: ETIM Verteilung logarithmisch

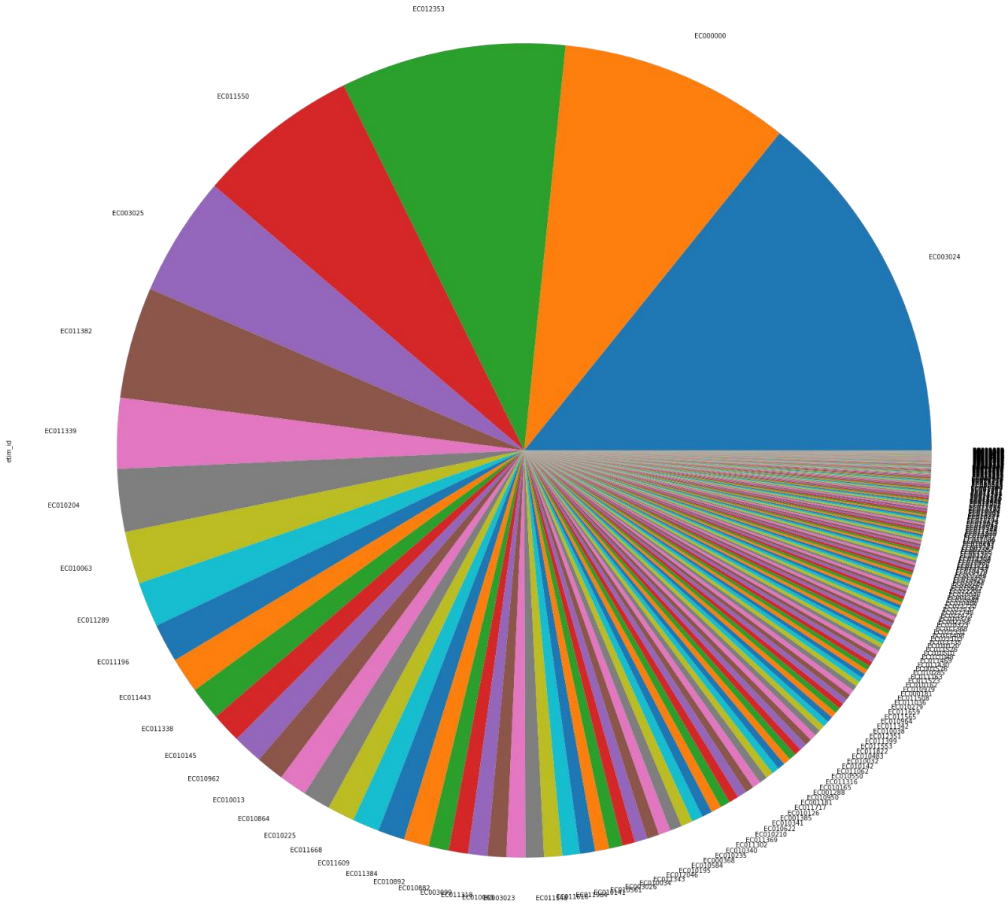

Abbildung 4: ETIM Vorkommen

In den obenstehenden Abbildungen werden die ETIM Klassen oder `etim_ids` und ihre numerischen Vorkommen in unterschiedlichen Darstellungen aufgezeigt. Daraus wird die Annahme gewonnen, dass eine Mindestanzahl an Vorkommen vorausgesetzt werden kann, bevor die Daten von Algorithmen verarbeitet werden.

Unique Values

column_name	unique_values	column_name	unique_values
article_id	25826	**category_id**	57
name	25826	**category_name**	57
aricle_number	25826	**family_id**	225
etim_description	260	**family_name**	132
etim_id	261	**group_id**	470
product_hierarchie	930	**group_name**	296
short_text	23535	**series_id**	1029
product_name	8712	**series_name**	519
path	8784	**type_id**	1774
system_id	7	**type_name**	590
system_name	7		

Tabelle 2 Unique Values

article_id, name und article_number

Die Spalten `aritcle_id`, `name` und `article_number` enthalten ausschliesslich einzigartige Werte. Ein Algorithmus, welcher auf diesen Spalten trainiert, erreicht eine Genauigkeit von 1.0 und würde jede `etim_id` aus dem Datenset perfekt zuweisen. Sobald jedoch ein neues Produkt hinzukommt, ist der Algorithmus ahnungslos.

product_name

Die Spalte `product_name` enthält ebenfalls ausschliesslich einzigartige Werte. Diese setzen sich jedoch zusammen aus einzelnen Worten wie dem Namen des Herstellers, Materialien und Grössenangaben des Artikels. Diese könnten theoretisch extrahiert werden.

etim_id etim_description

Überraschenderweise haben die `etim_id` und die `etim_description` unterschiedlich viele unique Values. Dies ist darauf zurückzuführen, dass die nichtexistierende `etim_id 'EC000000'` ebenfalls dazugezählt wird, jedoch keine `etim_descritpion` dafür existiert.

Wie bereits angesprochen liegen `etim_id` und `etim_description` in einer 1:1 Beziehung zueinander.

short_text

Der `short_text` besteht ebenfalls aus vielen einzigartige Werte.
Im Gegensatz zu `aritcle_id`, `name` und `article_number` ist es jedoch zu voreilig diese Spalte abzuschreiben. Die Werte bestehen ausschliesslich aus Text, woraus potenziell viele Informationen gewonnen werden können.

path

Aus der Tabelle 1 geht hervor, dass der `path` aus den Spalten `[]_ids` und `[]_names` besteht. Der `path` ist somit eine mögliche Information Akkumulation bestehend aus den `[]_name` Spalten.
Die Grundkonstellation ist die Folgende:
`Systeme\system_name\category_name\family_name\...`
Der Rest des `path` ist besteht aus abgeänderten oder umgeschriebenen Variationen der `group_name`, `series_name` und `type_name` Spalten.
Dies ist der Fall, da der `path` die Ordnerstruktur auf dem Geberit Server widerspiegelt.

product_hierarchie

Die erste Hälfte der `product_hierarchie` setzt sich zusammen aus drei Zahlen, die eine Systematische Hierarchie widerspiegelt, nach welcher Geberit ihre Produkte einordnet, aber weder mit ihrer Ordnerstruktur noch mit dem ETIM System übereinstimmt.
Die zweite Hälfte ist eine Liste von Wörtern, welche aus den Technischen Daten des Artikels generiert wird.

[]_id, []_name:

Es ist sinnvoll diese Spalten genauer zu untersuchen. Unteranderem weisen sie eine akzeptable Menge von einzigartigen Werten auf und sind die Bausteine aus denen der `path` zusammengesetzt ist. Dabei fällt auf, dass sich die Menge der einzigartigen Werte stetig erhöhen, je «tiefer» die jeweilige Position im `path`.

Ebenfalls auffällig ist die unterschiedlichen Anzahl der einzigartigen Werte für ein `[]_id` und `[]_name` Paar. Eine nähere Betrachtung der Korrelationen zwischen `[]_id` und `[]_name` führt zu folgendem Ergebnis:

1. Alle `[]_id` Werte können **eindeutig** dem jeweiligen `[]_name` zugeordnet werden.

2. `[]_name` Werte können **unterschiedlichen** `[]_id` zugeordnet werden.

Das heisst, dass sich ein Wert aus `[]_id` stets auf den selben `[]_name` Wert abbilden lässt. Umgekehrt gilt jedoch, dass ein `[]_name` Wert mehreren `[]_id` Werten zugeordnet werden kann, z.B.:

1. Die `type_id 'TYP_101063'` wird immer auf den `type_name 'Anschlusswinkel für Gaszähler'` abgebildet.

2. Der `type_name 'Anschlusswinkel für Gaszähler'` wird aber auf die beiden `type_ids 'TYP_101063'` und `'TYP_100973'` abgebildet.

Dies führt zu der Annahme, dass die `[]_id` potenziell mehr Informationen trägt als der `[]_name`.

Untenstehend ist eine Tabelle mit Anzahl der Fälle in denen der Wert aus `[]_namen` mehr als einem Wert aus `[]_id` zugeordnet ist.

[]_name	Anzahl []_ id Zuteilungen
system_name	0
category_name	0

family_name	16
group_name	54
series_name	185
type_name	250

Tabelle 3: Zuteilungen der Namen und Ids

NaN

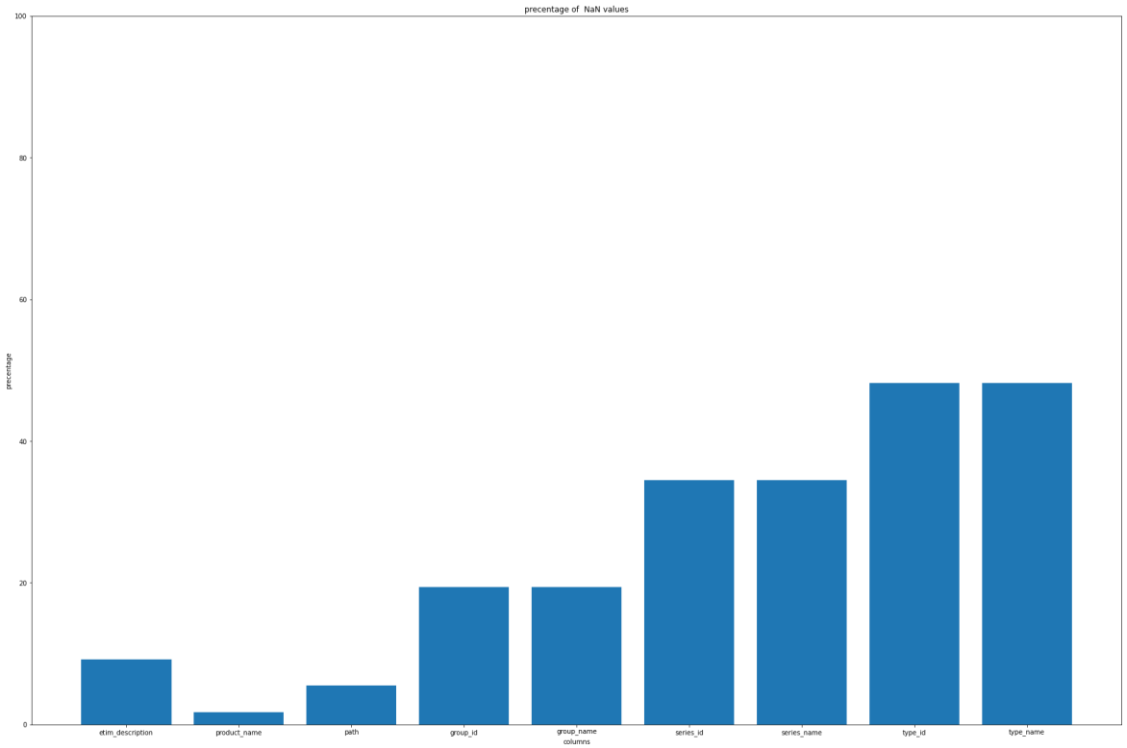

Abbildung 5: Prozentanteil der NaN Werte

Eine weitere wichtige Beobachtung beschreibt die Anzahl der `NaN` Werte in den jeweiligen Spalten.
Interessant ist, dass sich die `NaN` Werte mit wenigen Ausnahmen auf die `group_name`, `series_name` und `type_name` Spalten beschränken. Diese Entdeckung unterstützt die Annahme, dass der `path` im Wesentlichen aus den Informationen der Spalten `system_name, category_name` und `family_name` bestehen.

Eine weitere spannende Beobachtung ist, dass ungefähr 50% der `type_id` und `type_name NaN` Werte sind.

Der Prozentsatz der `NaN` Werte für die `etim_description` ist damit zu erklären, dass der `etim_id 'EC00000'` keine `etim_description` zugeordnet werden kann.

4.3 Bayes Error Analyse

Motivation

Der Bayes Error soll per Definition die kleinstmögliche Error Rate berechnen. Aus den gewonnenen Informationen werden Entscheidungen für die weitere Entwicklung der Algorithmen hervorgehen.

Fehlerursache

Mit den gegebenen Daten gibt es nur eine einzige mögliche Ursache für unvermeidbare Fehler. Wenn es in den Trainingsdaten mehrere Datensätze gibt, welche identischen Input aber unterschiedliche `etim_ids` haben, gibt es keine bessere Strategie, als die Häufigste dieser `etim_ids` auszugeben.

Probleme

Die Geberit Daten haben ein Problem, was den Bayes Error betrifft. Es gibt einige Spalten, die einzigartig auf allen Datensätzen sind. Dies führt zu einem Bayes Error von Null. Deshalb wird der Bayes Error auf diesen Spalten nicht berechnet.

Vorgehen

Für die Entwicklung eines Algorithmus braucht es Informationen bezüglich der Spalten auf denen gelernt werden soll. Für dieses Projekt wird ein Bayes Algorithmus aufgebaut, sodass einzelne irreduzible Error Analysen auf einer bestimmten Anzahl Spalten durchgeführt werden können. Dieser Algorithmus liefert Informationen für eine bestmögliche Auswahl an Spalten, auf denen trainiert werden sollte.

Auswahlverfahren

Für die Auswahl der Spalten präsentieren sich einige Kriterien:

- Anzahl Kolonnen
- Anzahl einzigartigen Werte
- Bayes Error Rate

Hinweis

Für den Bayes Algorithmus wurden die Spalten `aritcle_id`, `name` und `article_number` bereits ausgenommen. Sie weisen eine einzigartige Zuordnung zu den `etim_ids` auf und sind daher nicht brauchbar.

Algorithmus

Der Algorithmus nimmt eine Menge von Inputspalten und gruppiert alle Datensätze nach Kombinationen dieser Spaltenwerte. Jede dieser Gruppen wird nach der `etim_id` gruppiert. Für jede Kombination der Spaltenwerte des Inputs wird diese `etim_id` Gruppe gelöscht, welche die höchste Kardinalität aufweist. Die restlichen Datensätze werden von einem beliebigen Algorithmus, der die Inputdaten perfekt gelernt hat, nie korrekt klassifiziert. Teilt man die Anzahl Datensätze die übrig geblieben sind durch die gesamte Anzahl Datensätze, ergibt sich daraus der Bayes Error.

Resultat

Component 0	uniques	Bayes Error
system_id	7	0.749090064
category_id	57	0.557384032
family_name	132	0.428599086
family_id	225	0.353403547
type_name	590	0.31749944
group_name	296	0.28394469

product_hierarchie	930	0.25811198
group_id	470	0.220424429
series_name	519	0.173180394
series_id	1029	0.098385857
type_id	1774	0.045559788
product_name	8712	0.011586207
path	8784	0.007129687
short_text	23535	0.002671726

Tabelle 4: Bayes Error mit einzelnen Komponenten

Die Zusammenstellung der einzelnen Spalten und ihren Bayes Errors ist logisch nachvollziehbar. Je mehr einzigartige Werte in einer Spalte, desto einfacher ist eine Zuordnung der `etim_id`.

Somit wäre es auf den bestehenden Daten theoretisch möglich eine korrekte Zuordnung mit Wahrscheinlichkeit von 0.997328274 zu erreichen, falls man sich ausschliesslich auf den `short_text` fokussiert.

Jedoch wäre es keinem Algorithmus möglich eine Zuordnung durchzuführen, wenn ein neues Produkt mit neuem `short_text` hinzugefügt werden soll.

Für die ursprüngliche Auswahl kommen also nur diejenigen Spalten in Frage, welche nicht zu viele einzigartigen Werte aufweisen.

Deshalb wird ab hier für die Auswahl der Spalten die Anzahl der einzigartigen Werte auf 1000 beschränkt.

Component 0	unique	Component 1	unique	Bayes Error
series_name	519	type_name	590	0.058376156
group_id	470	type_name	590	0.063803005
product_hierarchie	930	type_name	590	0.066547166
group_name	296	type_name	590	0.072142239
product_hierarchie	930	series_name	519	0.089930822
group_id	470	series_name	519	0.09122191

Tabelle 5: Bayes Error mit zwei Komponenten

Diese Tabelle enthält alle 2-er Kombinationen, die unter einem Bayes Error von 0.1 liegen.

Es kann bereits eine leichte Tendenz herausgelesen werden. Die Kombinationen bestehen ausschliesslich aus den Spalten `product_hierarchie`, `group_id`, `series_name` und `type_name`. Es ist zu erwarten, dass sich dieser Trend auch mit 3-er Kombinationen fortsetzen wird.

Component 0	Component 1	Component 2	Bayes Error

product_hierarchie	series_name	type_name	0.029085303
group_id	series_name	type_name	0.035952637
group_name	series_name	type_name	0.040043057
product_hierarchie	group_id	type_name	0.042246873
product_hierarchie	group_name	type_name	0.044921721
family_id	series_name	type_name	0.047790339
family_name	series_name	type_name	0.047995889

Tabelle 6: Bayes Error mit drei Komponenten

Für obenstehende Auswahl der 3-er Kombinationen wurde der Bayes Error auf 0.05 gesenkt.

Die vorhergesagte Tendenz bezüglich der Spalten erfüllt sich mit einer Ausnahme. Eine Kombination mit `family_id` schafft es ebenfalls, die Kriterien zu erfüllen.

Ab diesem Punkt ist es möglich eine Voraussage bezüglich den 4-er Kombinationen zu machen. Dem bereits in den 2-er Kombinationen bemerkten Trend folgend, wird die beste Kombination aus `product_hierarchie, group_id, series_name` und `type_name` bestehen.

Component 0	Component 1	Component 2	Component 3	Bayes Error
product_hier.	group_id	series_name	type_name	0.02583423
product_hier.	group_name	series_name	type_name	0.026372443
product_hier.	family_name	series_name	type_name	0.028468654
product_hier.	family_id	series_name	type_name	0.028468654
product_hier.	system_id	series_name	type_name	0.029085303
product_hier.	category_id	series_name	type_name	0.029085303

Tabelle 7: Bayes Error mit vier Komponenten

Die Voraussage trifft zu und somit ergibt sich Folgendes als beste Kombination:

- Spalten: `product_hierarchie, group_id, series_name, type_name`
- Anzahl einzigartigen Werte: < 1000
- Bayes Error: 0.0258342303552207

Die gesamten Daten zur Bayes Analyse sind im Abgabeordner unter `analysis/bayes_error_all_data_kombinationen.xlsx` abgelegt.

Schlussfolgerung

Anhand der Bayes Error Analyse können die vier Spalten `product_hierarchie, group_id, series_name, type_name` als aussagekräftigste Kombination identifiziert werden.

Es gibt auf den originalen Daten also einen theoretischen irreduziblen Fehler von 0.02583423, solange die ausgewählten Spalten als Feature Spalte genutzt werden.

Die Bayes Analyse hat einen weiteren Vorteil.
Es ist möglich den Bayes Error auf einem der Trainingsset zu berechnen, um eine obere Grenze für die Trainingsaccuracy von einem Algorithmus zu bestimmen.

4.4 Schlussfolgerungen

EC000000

Alle Datensätze, die eine `etim_id` von `'EC000000'` aufweisen, werden aus den Daten entfernt.
Geberit verwendet die ETIM Klasse `'EC000000'` für alle Artikel, welche aus unbestimmten Gründen nicht durch einen Sachbearbeiter manuell klassifiziert werden können. Dies geschieht einerseits durch Human Error oder ist darauf zurückzuführen, dass die passende ETIM Klasse im ETIM Standard noch nicht vorhanden ist.
Es wäre interessant, ein Algorithmus zu trainieren, der vorhersagt, in welchen Fällen für einen Artikel keine ETIM Klasse existiert. Es gibt anhand der verfügbaren Daten jedoch keine Indizien dafür, ob dieser Fall vorliegt oder ob es sich schlichtweg um einen Human Error handelt. Basierend auf dieser Überlegung werden die `'EC000000'` für das Training ignoriert.

Einzigartige Werte

Die Spalten `article_id`, `name` und `article_number` weisen für jeden Datensatz einzigartige Werte auf und haben für die Problembehandlung keinen Nutzen.

system_id
system_name

Diese beiden Spalten sind redundant. Sie weisen eine eindeutige Zuweisung, also eine 1:1 Beziehung zueinander auf.

category_id
category_name

Diese beiden Spalten sind redundant. Sie weisen eine eindeutige Zuweisung, also eine 1:1 Beziehung zueinander auf.

Top etim_id

Die acht meist vorkommenden `etim_ids` machen ungefähr 50% der Zuteilungen aus. Viele `etim_ids` weisen ein Vorkommen von weniger als zwei auf. Diese Zeilen werden ignoriert.

Bayes Error

Aus der Bayes Analyse geht hervor, dass `product_hierarchie, group_id, series_name, type_name` voraussichtlich gut geeignete Spalten für das Trainieren bilden.

path

Der `path` besteht im Wesentlichen aus den Informationen der `system_name, category_name` und `family_name` Spalten. Diese Spalte weist daher eine akkumulierte Informationsansammlung auf.

Duplikate

Viele Datensätze sind in ihren Spaltenwerten identisch. Die Duplikationen werden nach einer optimalen Spaltenauswahl entfernt. Dies ist nötig, damit ein Algorithmus einen mehrfach vorkommenden Dateninput nicht unproportional gut lernt.

NaN Werte `NaN` Werte werden ohne Informationsverlust mit dem String: `'no information'` ersetzt.

5 Datenaufbereitung

5.1 Encoding

Allgemein

Da ausschliesslich mit kategorischen Daten gearbeitet wird, ist es unumgänglich mit Encodings zu arbeiten. Die Art des Encodings ist abhängig von der Wahl der Algorithmen. Für die Benutzung der meisten maschinell lernenden Algorithmen sind numerischen Werten vorauszusetzen.

One-Hot-Encoding

Dafür bietet sich das One-Hot-Encoding an. Es bildet jeden einzigartigen Wert auf einen einzigartigen n-dimensionalen Vektor aus Nullen und Einsen ab.

Damit können die kategorischen textbasierten Informationen auf ein numerisch basiertes System abgebildet werden, die jeweils denselben Abstand zueinander haben. Das ist vorteilhaft, da lernende Algorithmen dadurch keine Ordnung auf kategorischen Daten herauslesen können.

Dieses Encoding wird für den Aufbau und Training eines Neuralen Netzwerks benutzt.

Embeddings

Eine zweite Möglichkeit ist der Gebrauch eines Textembeddings. Die Aufgabe von Textmbeddings besteht darin, aus textbasierten Informationen Zusammenhänge zu erlernen und diese auf einen Vektor abzubilden.

Da ein Grossteil der Spaltenwerte deutschsprachiger Text ist, könnte ein schon trainiertes deutsches Embedding verwendet werden. Dieses wäre zum Beispiel auf der Gesamtheit der deutschen Wikipedia Seite trainiert worden.

Dies hat jedoch zwei Nachteile:

1. Das Training ist «unsupervised». Das heisst der Zusammenhang zwischen dem Resultierenden Vektor und der `etim_id` noch muss noch separat erlernt werden.
2. In den Daten sind die meisten Wörter aus dem Sanitärbereich. Im schlimmsten Fall sogar Wörter, welche es gar nicht gibt wie zum Beispiel ‘50x50’. Dies führt dazu, dass ein ungemein grosses Model gebraucht wird, damit in sich so ähnlichen Texten wesentliche Unterschiede erkannt werden.

Aus den obig genannten Gründen ergibt es Sinn ein eigenes Embedding mit der fastText Library von Facebook[2] zu erstellen.

5.1.1 Dimensionen

Allgemein

Während beim Textembedding die Anzahl Dimensionen willkürlich festgelegt werden können, ist dies beim One-Hot-Encoding nicht so einfach möglich.

Durch die Datenanalyse wird aufgezeigt, wie viele einzigartige Werte in jeder Spalte vorhanden sind. Die Kodierung all dieser Werte mit dem oben beschriebenen One-Hot-Encodings führt zu einer riesigen Anzahl an Dimensionen.

[2] https://fasttext.cc/, Zugriff: 01.06.2021

Dimensionalität

Um diesen riesigen Dimensionen entgegenzuwirken, werden durch die Analyse des Bayes Errors diejenigen Spalten ausgewählt, welche mit möglichst geringer Anzahl verschiedener Werte, einen möglichst geringen Bayes Error erreichen. Dies Spalten weisen das beste Verhältnis von Information zur Grösse der Dimensionen auf.

Resultat

Mit Hilfe der Bayes Error Analyse, wurden `etim_id`, `product_hierarchie`, `group_id`, `series_name`, `type_name` und, wie noch aufgezeigt wird, `path` als beste Spaltenkombination identifiziert und somit werden die Dimensionen der One-Hot-Encodings drastisch reduziert.

5.2 Übersicht der Aktionen

Übersicht

In diesem Abschnitt werden allgemeine Tatsachen der Daten und die daraus resultierenden Aktionen durch die Aufbereitung beschrieben. Diese Informationen gehen direkt aus den Schlussfolgerungen der Datenanalyse hervor und werden hier übersichtlich dargestellt.

Die Aktionen finden in der nachfolgenden Sektion bei der Unterteilung der Datensets ihre Anwendung.

5.2.1 Löschen

EC000000

Dies stellt die einfachste Entscheidung für die Aufbereitung dar. Datensätze mit einer `etim_id 'EC000000'` werden im maschinellen Lernen nicht gebraucht. Diese 2367 Zeilen werden daher aus dem Datensatz gelöscht.

Vorkommen

Alle `etim_ids`, die weniger als 2-mal im Datenset vorkommen, sind nicht aussagekräftig genug und werden deshalb entfernt.

Duplikate

Duplikate der Datensätze müssen gelöscht werden, da sie den Lernprozess der Algorithmen negativ beeinflussen können.

5.2.2 Ersetzen

NaN

Beim Import entstehen aus nicht ausgefüllten Zellen `NaN` Zellen.
Diese Zellen werden durch den String `'no information'` ersetzt.

Einamalig vorkommende Keys

Die Werte, die einmalig pro Spalte vorkommen, haben entweder keinerlei oder zu viel Einfluss auf die Entscheidung des Labels.
Diese Zellen werden mit dem String `'unique'` ersetzt.

5.2.3 Auswählen

article_id Die `article_id` besteht nur aus einmalig vorkommenden Werten.
Sie hat daher keine Relevanz für die Problemlösung.

article_number Die `article_number` besteht nur aus einmalig vorkommenden Werten.
Sie hat daher keine Relevanz für die Problemlösung.

Hinweis Aus der Datenanalyse geht hervor, dass neben `article_id` und `article_number` noch weitere Spalten als redundant oder als nicht relevant eingestuft wurden.

Diese Spalten wurden hier bewusst nicht gelistet, da sie für das Textembedding durchaus einen Mehrwert haben können. Vor allem die Spalten `name` und `product_name` weisen einen hohen potenziellen textbasierten Informationsgehalt auf.

5.3 Unterteilung Test-, Validierungs- und Trainingsset

Daten zur Beurteilung der Studienarbeit Vor der Datenübergabe an die Studierenden wurden 1000 Datensätze entfernt, anhand derer das Resultat der Arbeit gemessen werden kann. Diese Daten werden den Studierenden vorenthalten. Im Rahmen dieser Arbeit wird davon ausgegangen, dass die übrig gebliebenen `geberit_data_original` alle Daten darstellen.
Die Auswertung mit diesen 1000 Datensätzen wird jedoch im Anhang **Fehler! Verweisquelle konnte nicht gefunden werden.** noch aufgeführt.

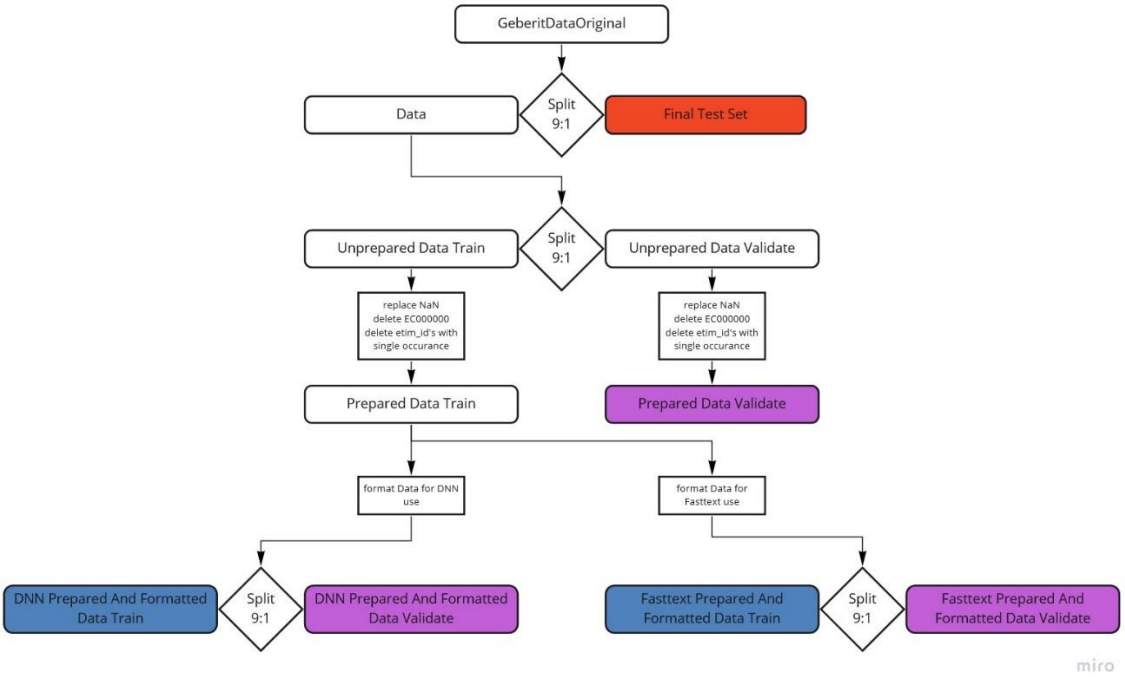

Abbildung 6: Graphische Darstellung der Datenaufbereitung erstellt mit http://www.miro.com

Mehrstufige Parameter

Die Entwicklung der Algorithmen spielt sich auf drei Ebenen ab:

1. Das Training der Modelle.
2. Das Optimieren der Hyperparameter.
3. Das Entscheiden wie gut die einzelnen Modelle sind und welche davon Geberit empfohlen werden sollen auf `prepared_data_validate`.

Testset

Um sich ein Bild zu verschaffen, wie sich ein Algorithmus auf neuen Daten verhält, ist ein Test-, oder Validierungsset nötig mit Daten, mit welchen der Algorithmus bis zu diesem Punkt noch nicht interagiert hat.

final_test_set

Um am Ende der Arbeit eine gute Metrik zu haben, wie gut der Algorithmus ist, wird das `final_test_set` im Verhältnis 1:9 von `geberit_data_original` abgetrennt. Es umfasst 2583 unbehandelte Daten.

Unprepared Data

Die aus dem ersten Split verbleibenden 23'243 Datensätze werden nun weiter in `unprepared_data_validate` und `unprepared_data_train` aufgeteilt. Der zweite Split wird ebenfalls im Verhältnis 1:9 durchgeführt.

Prepared Data

Auf `unprepared_data_validate` und `unprepared_data_train` wird nun eine erste Datenaufbereitung durchgeführt.

1. `NaN` Werte werden durch `'no information'` ersetzt
2. Records mit `etim_id 'EC000000'` werden gelöscht
3. `etim_ids` mit weniger als 2 Vorkommen werden gelöscht

Die beiden bearbeiteten Sets werden nun als `prepared_data_validate` und `prepared_data_train` abgespeichert.

`Prepared_data_validate` wird gebraucht, um die Modelle, welche aus den verschiedenen Ansätzen entstehen, zu vergleichen.

`Prepared_data_train` stellt die Grundlage des Trainierens aller Modelle dar.

Falls ein Algorithmus Hyperparameter optimiert, muss das Validierungsset dafür ebenfalls aus `prepared_data_train` kommen.

Lookup_prepared_and_formatted_data

Da der Lookup Table keine Hyperparameter hat und direkt trainieren kann, müssen dafür keine weiteren Massnahmen getroffen werden.

dnn_prepared_and_formatted_data

Das DNN ist der erste Algorithmus, welcher zusätzliche Datenaufbereitung benötigt:

1. Spaltenauswahl:
   `etim_id, product_hierarchie, group_id, path, series_name, type_name, path.`
2. Löschen der Duplikate der ausgewählten Spalten.
3. Einzigartige Werte ersetzten durch `'unique'`

Da die Daten sehr unregelmässig verteilt sind, tendiert das DNN dazu, seltene Inputs zu ignorieren und stattdessen die häufigsten Inputs zu lernen. Der 2. Schritt soll dem entgegenwirken. Er verhindert, dass derselbe Input mehrmals pro Durchlauf trainiert wird.

Die im 3. Schritt als `'unique'` gekennzeichneten Werte werden in einem separatem Dictionary festgehalten, damit sie bei der Trainingsvalidierung ebenfalls ersetzt werden können.

Für das Training der Hyperparameter muss von dem formatierten Trainingsset ein Validierungsset abgetrennt werden. Um zu verhindern, dass im Validierungsset `etim_ids` vorkommen, welche im Trainingset nicht vorkommen, werden die Daten auf jeder `etim_id` separat im Verhältnis 9:1 auf `dnn_prepared_and_formatted_data_train` und `dnn_prepared_and_formatted_data_validate` aufgeteilt.

fasttext_prepared_and_formatted_data

Für fastText werden die Daten folgendermassen formatiert:

1. Konkatenieren des Inhalts der Spalten `name, short_text, product_name, path, system_name, category_name, group_name, series_name` und `type_name` sowie die zweite Hälfte der `product_hierarchie` zu einem String `text`.
2. Umwandeln zur Form:
   `'__label__' ++ etim_id ++ ' ' ++ text`
3. Zufälliges Aufteilen der Daten im Verhältnis 9:1 auf `fasttext_prepared_and_formatted_data_train.txt` und `fasttext_prepared_and_formatted_data_validate.txt`.

Zusammenfassung

Anbei eine Zusammenfassung der Aktionen auf den jeweiligen Sets:

Daten Set	Aktionen
original_data	Split
data	Split
final_test_set	-
unprepared_data_train	-
unprepared_data_validate	-
prepared_data_train	Löschen & Ersetzen
prepared_data_validate	Löschen & Ersetzen
dnn_prepared_and_formatted_data_train	Auswahl & Löschen
dnn_prepared_and_formatted_data_validate	Auswahl & Löschen
fasttext_prepared_and_formatted_data_train	Auswahl
fasttext_prepared_and_formatted_data_validate	Auswahl

Tabelle 8: Datenbearbeitung Aktionen

Übersicht und numerische Informationen

Die angrenzende Tabelle enthält numerische Informationen bezüglich der verschiedene Sets:

Daten Set	# Records	# etim_id
original_data	25826	261
data	23243	255
final_test_set	2583	176
unprepared_data_train	20918	254
unprepared_data_validate	2325	170
prepared_data_train	18982	218
prepared_data_validate	2109	169
dnn_prepared_and_formatted_data_train	8213	218
dnn_prepared_and_formatted_data_validate	1027	203
fasttext_prepared_and_formatted_data_train	17083	218
fasttext_prepared_and_formatted_data_validate	1899	172

Tabelle 9: Datenbearbeitung numerische Details

Nutzung der Sets

Folgend wurde der Verwendungszweck all der Datensets aufgelistet:

Daten Set	Verwendung
data	Finales Trainieren der Modelle
final_test_set	Finale Validierung der Modelle
unprepared_data_train	Zwischenschritt
unprepared_data_validate	Zwischenschritt
prepared_data_train	Basis der Trainingsdaten, Lookup
prepared_data_validate	Validierung Trainingsmodelle, bestimmen, welche Kombination davon sinnvoll sind.
dnn_prepared_and_formatted_data_train	Training des DNN Models
dnn_prepared_and_formatted_data_validate	Hyperparameter Optimierung des DNN Models
fasttext_prepared_and_formatted_data_train	Training des fastText Models
fasttext_prepared_and_formatted_data_validate	Hyperparameter Optimierung des fastText Models

Tabelle 10: Datenbearbeitung Zweck

6 Algorithmen

6.1 Definitionen

Top 1

Ein Top-1 Modell liefert jeweils die wahrscheinlichste EIMT Klasse für einen bestimmten Input.

Bei jedem Top-1 Ergebnis entscheidet jeweils der Algorithmus, welche `etim_id` ausgewählt wird. Aus der Problemstellung geht jedoch heraus, dass dies bisher durch manuelle Bearbeitung gemacht wurde.

Top 3

Bei den Top-3 Modellen ist ein Benutzer gezwungen, sich am Entscheidungsprozess zu beteiligen, indem ihm drei `etim_id` zur Auswahl präsentiert werden.

Top-3 Algorithmen sind somit eine Kombination aus maschineller Vorarbeit und menschlicher Schlussentscheidung.

Resultate und Metriken

Das Resultat eines Algorithmus setzt sich aus folgenden Metriken zusammen:

Resultat	Beschreibung
Trainingsaccuracy	Die Trainingsaccuracy beschreibt die Genauigkeit des Models auf den Trainingsdaten
Validationaccuracy	Die Validationaccuracy beschreibt die Genauigkeit des Models auf den Validierungsdaten
Bayes/Trainings Differenz	Die Bayes/Trainings Differenz beschreibt Differenz zwischen der Genauigkeit des Models auf den Trainingsdaten und der theoretisch bestmöglichen Genauigkeit auf den Trainingsdaten.
Top 1 Accuracy	Ein Top-1 Resultat enthält jeweils nur die wahrscheinlichste ETIM Klasse und beschreibt die vollautomatische Klassifizierung der Produkte. Die Top-1-Accuracy beschreibt die Genauigkeit des Top-1 Resultats auf den Testdaten `prepared_data_validate`.
Top 3 Accuracy	Die Top-3-Accuracy beschreibt den Recall für k=3 den Testdaten `prepared_data_validate`. Oder in anderen Worten, die Wahrscheinlichkeit, dass wenn ein Modell die drei wahrscheinlichsten `etim_ids` ausgibt, die Richtige dabei ist.

Tabelle 11 Verwendete Metriken

6.2 Übersicht

Benchmark

Nach der Analyse und Aufbereitung der Daten, wird zuerst ein Benchmark Algorithmus erstellt. Das Ziel dieses Algorithmus ist, eine grobe Einschätzung zu machen, wie gut die Daten lernbar sind.

Lookup Table Ein Lookup Table auf den Daten soll alle eindeutig zuweisbaren Fälle übernehmen und kann in Kombination mit weiteren Modellen als Vorarbeiter agieren.

DNN Ein Deep Neural Network oder DNN soll die Zuweisungen der `etim_ids` mit Hilfe von One-Hot-Encodings auf ausgewählten Spalten erlernen.

fastText Der fastText Algorithmus soll die Zuweisung der `etim_ids` anhand des hohen Informationsgehaltes der Spaltenwerte durch Text Embeddings erlernen.

Kombinationen Kombinationen und Varianten der oben genannten Algorithmen werden in die Analyse miteinbezogen.

6.3 Benchmark

Data Set Für den Benchmark wird das Data Set `prepared_data_train` benutzt.

Data Selection Der Benchmark läuft ausschliesslich auf den Kolonnen `etim_id` und `product_hierarchie`.

Vorgehen Für die Benchmark wird die `product_hierachie` One-Hot encodiert, um damit ein Neuronales Netzwerk zu trainieren.

Ausführung Diese Daten werden dann im Verhältnis 3:1 in ein Trainings-, und Validierungsset aufgeteilt trainiert:

Epochen	30
Batchsize	128
Optimizer	Adam
Learning rate	0.01

Netzwerk

```
model = Sequential()
model.add(Dense(800, activation='relu', kernel_initializer='he_normal', input_shape=(n_features,)))
model.add(Dense(700, activation='relu', kernel_initializer='he_normal'))
model.add(Dense(300, activation='relu', kernel_initializer='he_normal'))
model.add(Dense(260, activation='relu', kernel_initializer='he_normal'))
model.add(Dense(260, activation='softmax'))
```

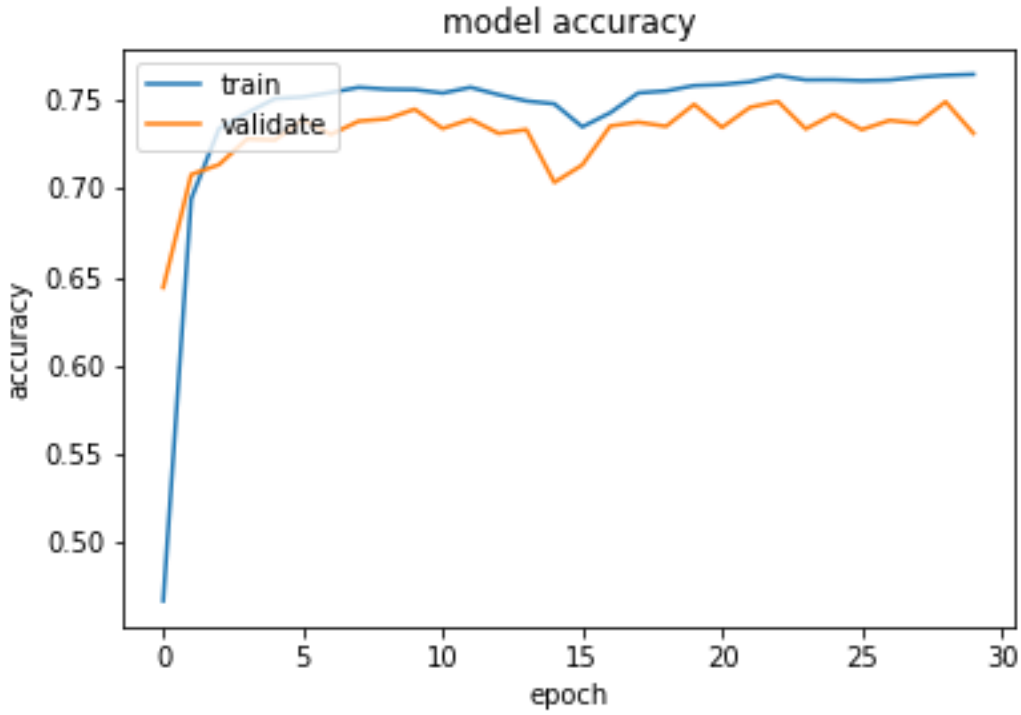

Abbildung 7: Benchmark Train/Validate Accuracy

Train Accuracy

Der Benchmark erreicht eine Trainingaccuracy von 0.767.

Validation Accuracy

Ohne weitere Optimierung kann der Benchmark Algorithmus Daten aus dem Validierungsset mit einer Accuracy von 0.749 richtig klassifizieren. Wenn also ein Algorithmus diese Baseline nicht klar übertrifft, wird er verworfen.

6.4 Lookup Table

6.4.1 Algorithmus

Daten

Für den Aufbau des Lookup Table wird `prepared_data_train` benutzt.
Diese Daten enthalten keine `NaN` Werte, keine EC000000 und keine `etim_ids` mit weniger als zwei Vorkommen. Diese Umstände bilden eine passende Grundlage für den Lookup Table.

Daten Auswahl

Die Basis des Lookup bildet folgende Kombination von Spalten:

```
etim_id,product_hierarchie, group_id, series_name,
type_name
```

Die Auswahl ist auf die Analyse des Bayes Errors gestützt.

Lookup Dictionary Der Aufbau des Lookup Table verläuft wie folgt:

1. Daten auf die oben beschriebenen Spalten beschränken
2. Aussonderung der Duplikate
3. `etim_id` wird in einem Set abgelegt und anschliessend im Dictionary mit den Spaltenwerten als Key abgespeichert

Resultat Es gibt drei mögliche Resultate für einen Lookup:

1. Einen `etim_id`
2. Mehrere `etim_ids`
3. Key Error

Eine etim_id Die Kombination aus `product_hierarchie, group_id, series_name` und `type_name` verweist auf eine einzelne `etim_id` und ist klar zuweisbar.

Mehrer etim_ids Die Kombination aus `product_hierarchie, group_id, series_name` und `type_name` auf verweist auf mehrere `etim_id's` und ist nicht klar zuweisbar.

Key Error Die Kombination aus `product_hierarchie, group_id, series_name` und `type_name` verweist auf keine `etim_id` und ist daher nicht zuweisbar. Die Kombination wird nicht im Datenset des Lookup Table gefunden und ist dem Algorithmus nicht bekannt.

Nutzen Die Aufgabe des Lookup Table besteht darin, alle eindeutig zugewiesenen `etim_ids` zurückzugeben und alle anderen Fälle an einen weiteren Algorithmus weiterzureichen.

6.4.2 Analyse

Data Set Da es sich beim Lookup Table ausschliesslich um Auswendiglernen handelt, wird eine Analyse direkt auf dem `prepared_data_validate` durchgeführt.

Certainity Rate Um die Performance des Lookup Table näher zu betrachten, wird die Certainity Rate eingeführt. Sie beschreibt wie sicher der Lookup bei seinen Entscheiden sein muss, bevor ein Resultat zurückgegeben wird.

Die Resultate mit mehreren `etim_ids` wird wie folgt behandelt:

> Berechnen des prozentualen Vorkommens der einzelnen `etim_ids`. Falls die häufigste vorkommende `etim_id` prozentual über der Certainity Rate liegt wird diese gewählt, ansonsten bleibt sie nicht zugewiesen.

Performance des Lookup Die Analyse des Lookup Table führt zu folgenden übergreifenden Ergebnissen:
Hierbei wurde in 0.05 Schritten von 0.5 bis 1.0 über die Certainty Rate iteriert.

Certainty Rate	Richtige Zuordnung	Falsche Zuordnung	Mehrere Werte	Key Error
0.0	0.8360	0.0621	0.0	0.1019
0.5	0.8222	0.0375	0.0384	0.1019
0.55	0.8137	0.0313	0.0531	0.1019
0.60	0.8070	0.0247	0.0664	0.1019
0.65	0.8028	0.0223	0.0730	0.1019
0.70	0.7961	0.0185	0.0835	0.1019
0.75	0.7914	0.0156	0.0910	0.1019
0.80	0.7838	0.0147	0.0996	0.1019
0.85	0.7776	0.0138	0.1067	0.1019
0.90	0.7643	0.0119	0.1219	0.1019
0.95	0.7435	0.0090	0.1456	0.1019
1.00	0.7421	0.0090	0.1470	0.1019

Tabelle 12: Lookup Perfomance

* Gerundet auf 2 Nachkommastellen

Richtige Zuordnung

Die *Richtige Zuordnungs* Rate bestimmt den Prozentsatz, der eindeutig richtig zugeteilten `etim_id` zu den jeweiligen Kolonnenwerte.

Falsche Zuordnung

Die *Falsche Zuordnungs* Rate bestimmt den Prozentsatz der eindeutig falsch zugewiesenen `etim_id` auf die jeweiligen wirklichen `etim_id`. Der Wert der `etim_id` aus dem Lookup entspricht also nicht dem eigentlichen Wert.

Dieses Vorkommen kann auf eine potenzielle Falschzuordnung zurückzuführen sein oder aus mehreren Werten wurde zwar die häufigste ausgewählt, jedoch war diese Auswahl falsch.

Key Error

Der *Key Error* entspricht einer Kombination aus Spaltenwerten, die in dieser Form nicht im Trainingset vorhanden war.

Mehrere Werte

Die *Mehreren Werte* beschreiben jene Kombinationen, welche sich nicht eindeutig auf eine `etim_id` zuweisen lassen und die häufigste `etim_id` hat die Certainty Rate nicht erreicht.

Schlussfolgerung

Der Lookup Table erreicht seine obere Grenze bei 0.836 richtigen Zuordnungen, solange ihm totale Freiheiten bei Entscheiden, also die jeweils wahrscheinlichste `etim_id`, gegeben werden. Jedoch ist seine Error Rate in diesem Falle mit 0.0621 zu hoch.

Die Aufgabe des Lookup soll darin bestehen, alle eindeutig zuweisbaren `etim_ids` in einem ersten Durchlauf zu klassifizieren. Den Rest soll an einen Algorithmus weitergegeben werde, der zuverlässigere Entscheidungen treffen kann.

Diese Aufgabe erfüllt der Lookup Table bei einer Certainity Rate von 1.0, wobei folgendes gilt:

Nicht klassifiziert	0.1470 (Mehrere Werte) + 0.1019 (Key Errors) = 0.2489
Klassifiziert (Richtig und Falsch)	1 - 0.2489 = 0.7511
Precision	0.7421 / (0.7421 + 0.0090) = 0.988

Tabelle 13: Lookup Schlussfolgerung

Das Resultat ist eine Accuracy von 98.8% auf denjenigen Daten, die eindeutig zugeordnet werden und 24.89% der Datensätze, die an einen zweiten Algortihmus zur Klassifizierung weitergegeben werden.

6.4.3 Erweiterung der Spaltenauswahl

Ausgangslage

Aus dem obigen Abschnitt geht hervor, dass 14.70 % der Daten nicht eindeutig einer `etim_id` zugeordnet werden können. Auf dem `prepared_data_train` Set sind das 307 Datensätze.

Eine weitere Analyse auf den nicht zugewiesenen Daten ist sinnvoll, um zu verstehen, wie diese Zuordnungen verbessert werden könnten.
Dies spielt vor allem für den Algorithmus eine Rolle, an den diese nicht durch den Lookup zuordbaren Daten weitergegeben werden, sprich dem DNN.

Vorgehen

Es werden zusätzlich alle Spalten untersucht, die nicht im Key des Lookup Table vorkommen. Daraus lässt sich auf potenziell informative Spalten schliessen, die bei einer Klassifizierung der Daten helfen.

Vorab müssen die Datensätze nach den Lookup Table Keys gruppiert werden. Anschliessend werden alle möglichen Werte für die Kolonnen `path, product_name, system_id, category_id, category_name, family_id, family_name, group_name, series_id` und `type_id` des jeweiligen Keys aufgelistet und abgespeichert.

Jeder Key muss untersucht werden, um zu überprüfen, ob in einer der Spalten mehr als ein Wert vorhanden ist. Dies wäre ein Indiz dafür, dass wenn diese Spalte zum Entscheidungsprozess hinzugezogen würde, bessere Resultate folgen.

Resultat

Die untenstehende Tabelle zeigt auf, welche Kolonne wie oft mehrere Werte enthalten.

Kolonne	Anzahl Fälle
path	20

product_name	20
system_id	0
category_id	1
category_name	1
family_id	1
family_name	0
group_name	0
series_id	0
type_id	0

Tabelle 14: Lookup Analyse

Zum Beispiel werde für 20 verschiedene Einträge im Lookup Table mehrere Einträge in der Kolonne `path` gefunden.

Schlussfolgerung

Durch diese Analyse werden die Kolonnen `path` und `product_name` als zusätzliche potenzielle Informationsquelle identifiziert.

In der Bayes Analyse wurden bereits Informationen über die Spalten gesammelt.

Column	Unique Values	Bayes Error
product_name	8712	0.011586207
path	8784	0.007129687

Tabelle 15: Bayes Error

Da `path` einen kleineren irreduziblen Fehler aufweist, ist es sinnvoll diese Spalte für den Neuralen Netzwerk Algorithmus im nächsten Abschnitt miteinzubinden. Ausserdem besteht der `path` im Wesentlichen aus den Informationen der `system_name`, `category_name` und `family_name` Spalten. Dementsprechend werden Informationen von Kombinationen dieser Spalten ebenso einfliessen.
Das Hinzufügen wird sicherlich den Trainings Error des DNN verkleinern, hat jedoch auch den Nachteil, dass es eine starke Erhöhung der Dimensionen zur Folge hat. Das liegt an den 8784 einzigartige Werte in der Spalte `path`.

6.5 Neuronales Netzwerk

6.5.1 Algorithmus

Daten

Als Trainingset wird `dnn_prepared_and_formatted_data_train` benutzt.
Als Validierungsset wird `dnn_prepared_and_formatted_data_validate` benutzt.
Als Testset wird `prepared_data_validate` benutzt.

Daten Auswahl

Die Basis des Neural Netzwerk bilden folgende Spalten:

`etim_id`, `product_hierarchie`, `group_id`, `series_name`, `type_name` und `path`

Die Auswahl ist auf die Analyse des Bayes Errors und des Lookup Table gestützt. Diese Spalten werden aus den `dnn_prepared_and_formatted_data_train` extrahiert und One-Hot codiert.

Encoder

Als One-Hot-Encoder wird `der OneHotEncoder` von `sklearn` verwendet. Die Einstellung `unknown='ignore'` führt dazu, dass Werte, welche im Trainingsset nicht anzutreffen sind, auf den Nullvektor abgebildet werden. Dieser `OneHotEncoder` wird als `Xencode.joblib` abgespeichert.
Für die `etim_id's` wird der `LabelEncoder` von `sklearn` verwendet. Dieser wird unter `yencode.joblib` gespeichert.

Netzwerk

Das Netzwerk wird durch manuelles Anpassen der Hyperparameter auf eine gewisse Nezwerk-Dimension eingestellt. Diese Einstellung erlaubt eine möglichst hohe Trainingsgenauigkeit. Das resultierende trainierte Modell wird unter `dnn_model.h5` abgespeichert.

Output Layer

Das Output Layer ist ein Softmax Layer, welches zu Beginn des Trainings mit der relativen Häufigkeit der `etim_ids` im Trainingsset gewichtet wird, um eine schnellere Konvergenz des Netzwerks zu erreichen.

Bayes Error

Die Analyse des Bayes Error auf `prepared_and_formatted_data_train` ergibt folgendes Resultat:

Comp. 0	Comp. 1	Comp. 2	Comp. 3	Comp. 4	Bayes Error
product_hier.	group_id	series_name	type_name	path	0.0657494

Der Algorithmus soll folglich eine Trainings-Accuracy von 0.934 anstreben.

Ausführung

Epochen 30

Batchsize 32

Optimizer Adam

Netzwerk

```
model = Sequential()
model.add(Dense(1900, activation='relu', kernel_initializer='he_normal', input_shape=(n_features,)))
model.add(Dense(1300, activation='relu', kernel_initializer='he_normal'))
model.add(Dense(900, activation='relu', kernel_initializer='he_normal'))
model.add(Dense(450, activation='relu', kernel_initializer='he_normal'))
model.add(Dense(250, activation='relu', kernel_initializer='he_normal'))
model.add(Dense(148, activation='relu', kernel_initializer='he_normal'))
model.add(Dense(180, activation='relu', kernel_initializer='he_normal'))
model.add(Dense(y_LabelEncoder.classes_.shape[0], activation='softmax'))
```

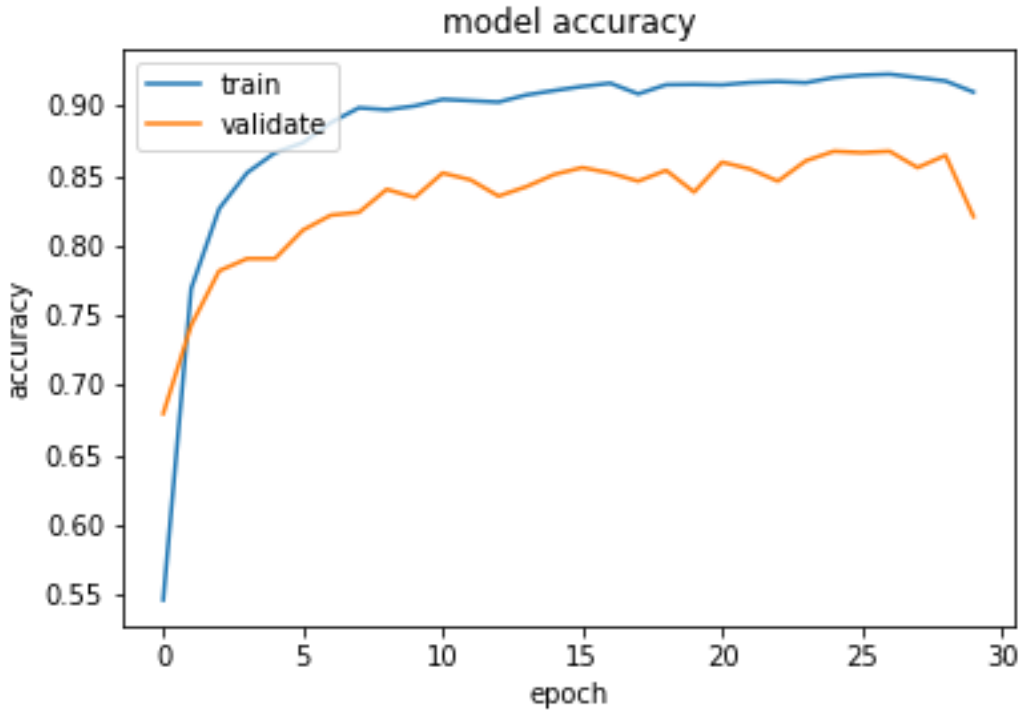

Abbildung 8: DNN Train/Validate Accuracy

Resultate

Training	Validation	Bayes	Bayes/Train Diff.	Top 1
0.910	0.821	0.934	0.024	0.881

Trainings & Validierungs Accuracy

Diese Konfiguration ergibt ein Model mit einer Trainingsaccuracy von 0.910 und einer Validierungsaccuracy von 0.821.

Bayes Error vs. Trainings Accuracy

Das Ziel des Trainings ist, sich dem Bayes Error von 0.934 so nahe wie möglich anzunähern. Die Differenz ist mit 0.024 jedoch zu hoch und muss verbessert werden.

Test Accuracy

Auf `prepared_data_validate` hat dieses Model eine Top-1-Accuracy von 0.881.

6.5.2 Verbesserungen

Metaparameter

Um das Netzwerk und den Lernprozess weiter zu verbessern, wurden diverse Metaparameter angepasst:

```
regularisation_parameter = 0.002
dropout_rate = 0.2
batch_size = 256
epochs = 72
```

Es wurde ein Dropout Layer zwischen allen Layern im DNN ausser dem Softmax Layer eingefügt, welches während dem Training alle Nodes des vorherigen Layers mit der Wahrscheinlichkeit `dropout_rate` ignoriert.

Für den `regularisation_parameter` wurde in allen Layern ausser dem Softmax Layer ein L1 Regularisierer mit dem Gewicht `regularisation_parameter` eingefügt.

Lernprozess

Die Parameter, welche beeinflussen wie schnell und wie genau das DNN die Trainingsdaten lernt, sind die `epochs` und die `batch_size` Parameter. Diese wurden von Hand durch Ausprobieren festgelegt.

Regularisierung

Die Parameter, welche beeinflussen wie gut das DNN Informationen aus den Trainingsdaten generalisiert, sind die `dropout_rate` und die der `regularisation_parameter`.

Der Algorithmus optimiert diese Parameter, indem systematisch verschiedene Kombinationen ausprobiert werden. Ausgewählt werde diejenigen, welche die höchste Validierungsaccuracy aufweisen.

Ausführung

Epochen	72
Batchsize	256
Regularisation	0.002
Dropout	0.2
Optimizer	Adam

Netzwerk

```
model = Sequential()
model.add(Dense(1900, activation='relu', kernel_initializer='he_normal', input_shape=(n_features,) , activity_regularizer=l1(regularisation_parameter)))
model.add(Dropout(dropout_rate))
model.add(Dense(1300, activation='relu', kernel_initializer='he_normal', activity_regularizer=l1(regularisation_parameter)))
model.add(Dropout(dropout_rate))
model.add(Dense(900, activation='relu', kernel_initializer='he_normal', activity_regularizer=l1(regularisation_parameter)))
model.add(Dropout(dropout_rate))
model.add(Dense(450, activation='relu', kernel_initializer='he_normal', activity_regularizer=l1(regularisation_parameter)))
model.add(Dropout(dropout_rate))
model.add(Dense(250, activation='relu', kernel_initializer='he_normal', activity_regularizer=l1(regularisation_parameter)))
model.add(Dropout(dropout_rate))
model.add(Dense(148, activation='relu', kernel_initializer='he_normal', activity_regularizer=l1(regularisation_parameter)))
model.add(Dropout(dropout_rate))
model.add(Dense(180, activation='relu', kernel_initializer='he_normal', activity_regularizer=l1(regularisation_parameter)))
model.add(Dense(y_LabelEncoder.classes_.shape[0], activation='softmax'))
```

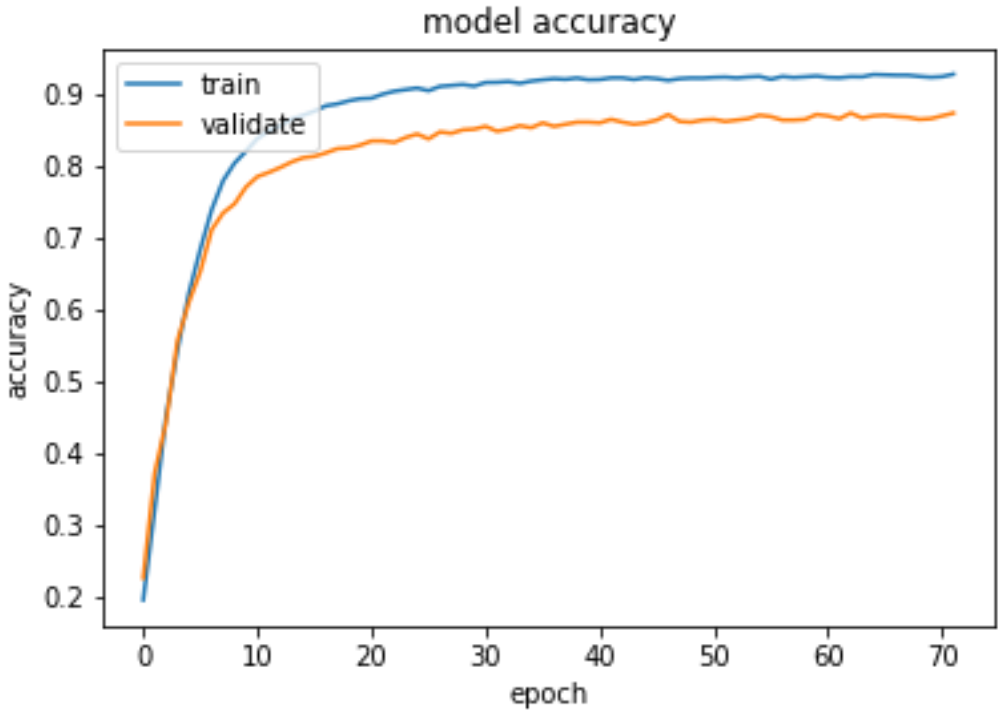

Abbildung 9: DNN Optimierte Train/Validate Accuracy

Wahrscheinlichkeit

Training	Validation	1-Bayes Err.	Bayes/Train Diff.	Top 1
0.928	0.874	0.934	0.006	0.911

Trainings & Validierungs Accuracy

Durch die optimierten Metaparametern erreicht das Model eine Trainings-Accuracy von 0.928 und eine Validierungs-Accuracy von 0.874.

Bayes Error vs. Training Accuracy

Eine Annäherung der Trainings-Accuracy an den Bayes Error von 0.006 ist zufriedenstellend.

Test Accuracy

Auf `prepared_data_validate` hat dieses Model eine Top-1-Accuracy von 0.911.

Validierungs vs. Test Accuracy

Es ist auffallend, dass das trainierte Model auf dem Validierungsset bedeutend schlechter abschneidet als auf dem Testset. Das ist auf die unterschiedliche Häufigkeit der etim_ids im `dnn_prepared_and_formatted_data_validate` gegenüber dem `prepared_data_validate` zurückzuführen. Durch die Datenaufbereitung, spezifisch durch das Löschen der Duplikate und dem Train-Validate Split, wurden die Anzahl Vorkommen von häufigen ETIM Klassen deren der seltenen angepasst.

6.6 Text Embedding mit fastText

6.6.1 Algorithmus

Idee

Der letzte Algorithmus braucht einen anderen Ansatz, um die Dimensionen der Trainingsdaten zu reduzieren. Anstelle des One-Hot-Encodings wird ein Textembedding benutzt. Da viele der Spalten in den ursprünglichen `prepared_data_train` Deutschen Text beinhalten, werden die Textinformationen extrahieren, um ein Textembedding darauf zu trainieren.

Daten

Als Trainingsset wird `fasttext_prepared_and_formatted_data_train` benutzt.
Als Validierungsset wird `fasttext_prepared_and_formatted_data_validate` benutzt.
Als Testset wird `prepared_data_validate` benutzt.

Daten Auswahl

Die Basis des fastText bilden folgende Spalten:

```
product_hierarchie, short_text, product_name, path, sys
tem_id, category_id, family_id, family_name, group_id,
group_name, series_id, series_name, type_id, type_name
```
und `name`

Die Auswahl wird in folgendes Format umgewandelt:

```
'__label__' ++ etim_id ++ ' ' ++ X
```

(Wobei `X` ein String mit allen textbasierten Informationen aus dem jeweiligen Datensatz ist)

Abgespeichert werden diese Strings in einem `.txt` File und später für das Training und die Validation hinzugezogen.

Vorgehen

Um das Textembedding zu trainieren, wird die Open Source Library fastText benutzt. Mit der Funktion `train_supervised` wird ein Model trainiert, das für jedes `X` und jede `etim_id` die Wahrscheinlichkeit voraussagt, dass `X` als `etim_id` klassifiziert wird. Das Model wird mit Learning Rate `lr=1.0` und der Anzahl Epochen `epoch=25` erstellt und anschliessend unter `fasttext_model.bin` abgespeichert.

Bayes Error

Eine Analyse anhand des Bayes Error macht im Fall des fastText Algorithmus keinen Sinn. Dies ist auf die Spaltenauswahl zurückzuführen. Da die Spalte `name` in der Auswahl enthalten ist erreicht der Bayes Error einen Wert von 0.0, da es sich in dieser Spalte ausschliesslich einzigartige Werte finden lassen.

Zusammenfassung

Training	Validation	Top 1
1.0	0.966	0.956

Trainings & Validierungs Accuracy

fastText erreicht ohne Optimierung eine Trainingsaccuracy von 1.000 und eine Validierungsaccuracy von 0.966

Test Accuracy Das fastText Model erreicht eine Top-1-Accuracy auf `prepared_data_validate` von 0.967.

6.6.2 Verbesserungen

Metaparameter FastText bietet die Funktion an, optimale Metaparameter zu bestimmen. Hierfür wird `fasttext_prepared_and_formatted_data_validate` verwendet.
Dafür werden beim Trainieren und anschliessend beim Validieren gewisse Metriken wie der F1 Score automatisch abgeglichen, um die optimale Parameter Einstellung zu finden.

Optimierte Parameter Nach einer Optimierungszeit von fünf Stunden sehen die Metaparameter wie folgt aus:

```
lr=5.0
dim=167
ws=5
epoch=100
minCount=1
minn=3
maxn=6
neg=5
wordNgrams=1
loss='softmax'
bucket=72689
thread=3
lrUpdateRate=100
t=0.0001
label='__label__'
```

Zusammenfassung

Training	Validation	Top 1
1.0	0.974	0.963

Trainings & Validierungs Accuracy Die Trainingsaccuracy ist 1.000. Dies ist jedoch nicht verwunderlich, da unter anderem die Spalte `name` benutzt wird, welche für jeden Artikel einzigartig ist. Die Spannende Metrik ist die Validierungsaccuracy von 0.974, welche misst, wie gut das Model generalisiert.

Test Accuracy Mit optimierten Metaparametern erreicht fastText auf `prepared_data_validate` eine Top-1-Accuracy von 0.9634.

6.7 Kombinationen

6.7.1 Top 3

Problem

Bei jedem Top-1 Ergebnis entscheidet jeweils der Algorithmus, welche `etim_id` ausgewählt wird. Aus der Problemstellung geht jedoch hervor, dass dies bisher durch manuelle Bearbeitung gemacht wurde.

Idee und Vorgehen

Um die Genauigkeit der Klassifizierung zu erhöhen, wird der Top-3-Algorithmus eingeführt. Es werden keine Änderung an den Model vorgenommen. Der Top-3-Algorithmus gibt nicht nur die wahrscheinlichste ETIM Klasse zurück, sondern erweitert diese Rückgabe auf die wahrscheinlichsten drei ETIM Klassen. Dadurch entsteht eine Kombination aus maschineller Vorarbeit und der endgültigen Entscheidung durch einen Sachbearbeiter. Dadurch kann die theoretische Genauigkeit des Algorithmus verbessert werden.

Daten

Die Datengrundlage bildet das in den jeweiligen Algorithmen beschriebene Schema.
Es werden keine weiteren Trainingsdurchläufe unternommen, weshalb nur die Top-1-Accuracy und die Top-3-Accuracy präsentiert werden.

Als Validierungsset wurde auf das `prepared_data_validate` Set zurückgegriffen, wie bereits in der Validierung des DNNs und des Lookup Table.

Zusammenfassung

DNN_Top 1	DNN_Top 3	fastText_Top 1	fastText_Top 3
0.921	0.951	0.963	0.985

Test Accuracy

Das DNN erhöht seine Accuracy gegenüber dem Top-1 Resultat auf 0.951.
Der fastText Algorithmus erhöht seine Accuracy gegenüber dem Top-1 Resultat auf 0.985.

Zusätzliches

Zu beachten ist, dass hier die Accuracy den Fall beschreibt, dass unter den Top-3 `etim_id` die richtige dabei ist. Das bedeutet, dass ein Benutzer die falsche Entscheidung treffen kann und somit die wirkliche Accuracy verschlechtert.
Jedoch bietet diese Idee die Möglichkeit unterschiedliche Entscheidungen vom User und vom Algorithmus miteinander zu vergleichen. Diese Erkenntnisse können genutzt werden, um die Algorithmen in Zukunft zu verbessern und sich der Auswahl des Benutzers anzupassen.

6.7.2 Lookup und DNN

Idee und Vorgehen

Der Lookup Table hat eine sehr hohe Präzision auf den Inputkeys, welche jeweils nur eine ETIM Klasse zurückgeben. Die vorhergesagten `etim_id` stimmen in 0.988 der Fälle.
Der Lookup Table wird genutzt, um alle eindeutig zuweisbaren `etim_id` zu klassifizieren. Die nicht zuweisbaren Datensätze werden dem DNN übergeben.

Daten

Die Datengrundlage bildet das in den jeweiligen Algorithmen beschriebene Schema.
Es werden keine weiteren Trainingsdurchläufe unternommen, weshalb nur die Top-1-Accuracy und die Top-3-Accuracy präsentiert werden.

Als Validierungsset wurde auf das `prepared_data_validate` Set zurückgegriffen, wie bereits in der Validierung des DNNs und des Lookup Table.

Zusammenfassugn

Top 1	Top 3
0.921	0.953

Test Accuracy

Die Kombination des Lookup Table und des DNN erreicht eine Top-1-Accuracy von 0.921 und eine Top-3-Accuracy von 0.953.

7 Vergleich der Resultate

7.1 Analyse der Resultate

Datenset

Die Analyse der Ergebnisse wurde auf dem `prepared_data_validate` durchgeführt. Dieses Set bildet das zu den Trainingsdaten komplementäre Validierungsset.

Metriken

Algorithmus	Accuracy
dnn_top_1	0.911332385
lookup_dnn_top_1	0.921289711
dnn_top_3	0.951161688
lookup_dnn_top_3	0.953532479
fastText_top_1	0.963489806
fastText_top_3	0.985775249

Tabelle 16: Algorithmen Vergleich Accuracy

Die obenstehende Tabelle weist den fastText Top-3-Algorithmus als genausten Algorithmus aus.

Es ist möglich einem Benutzer eine Auswahl von drei `etim_ids` zu präsentieren, in der die richtige Klassifizierung mit einer Wahrscheinlichkeit von 98.58% vorhanden ist.

Ursachen

Dafür, dass fastText so viel besser abschneidet, als das DNN gibt es zwei Erklärungen:

1. Das DNN trainiert auf weniger Spalten als fastText es tut. Aufgrund der begrenzten Rechenkapazität, welche zur Verfügung stand, musste die Spaltenanzahl für das DNN auf ein Minimum reduziert werden.
2. Das Textembedding kann die Nähe zweier verschiedener Werte abbilden, während beim One-Hot-Encoding alle verschiedenen Werte äquidistant sind. Dies resultiert darin, dass das DNN Werte, welche beim Training nicht aufgetreten sind, auf `Null` abgebildet. Das Textembedding merkt hingegen, wenn zum Beispiel zwei `short_texts` bis auf ein Wort identisch sind.

Metriken auf Klassen Stufe

Eine genauere Übersicht der Metriken sind im Excel Arbeit Sheet `Metriken.xlsx` enthalten. Dabei sind noch weitere Informationen bezüglich der Accuracy, des Recall und der Precision auf individuellen `etim_ids` vorhanden. Die Angaben geben Ausschluss darüber, welche Klassen im Einzelfall eine gute Zuteilung erfahren und bei welchen die Algorithmen Mühe haben.

etim_id	Lookup Accuracy	DNN Accuracy	fastText Accuracy
EC010890	0.0	0.0	0.0
EC000406	lookup undecided	0.0	0.0
EC000474	lookup undecided	0.0	0.0
EC010042	lookup undecided	0.0	0.0

EC010081	lookup undecided	0.0	0.0
EC010170	lookup undecided	0.0	0.0
EC010394	lookup undecided	0.0	0.0
EC010786	lookup undecided	0.0	0.0
EC010827	lookup undecided	0	0
EC010391	lookup undecided	0.5	0

Tabelle 17: Schlechteste etim_id Accuracy

Die obenstehende Tabelle zeigt zum Beispiel die zehn `etim_ids`, die für die Algorithmen nicht zuordbar sind. Dabei handelt es sich im Normalfall um `etim_ids`, die ein sehr geringes Vorkommen aufweisen.

Nichtsdestotrotz wären diese Informationen bei einer definitiven Umsetzung von Seitens Geberit von Nutzen, da schwierig zu klassifizierende `etim_ids` als Spezialfall behandelt werden könnten.

7.2 Rücksprache mit Geberit

Top-1 vs Top-3

Bei der Besprechung der erhobenen Metriken möchte Geberit sich nicht auf vollautomatisierte Klassifizierung verlassen, sondern immer auf noch einen Menschen in die Entscheidung einbeziehen. Aus diesem Grund macht es wenig Sinn, einen Algorithmus zu produzieren, welche nur eine Klasse ausgibt. Eine sinnvollere Lösung wäre ein Tool, welches die drei Wahrscheinlichsten Klassen ausgibt, von welchen dann ein Mensch die richtige auswählen kann.

fastText

Von den Algorithmen, welche drei ETIM Klassen herausgeben ist fastText Top-3 mit 0.985 Accuracy klar der Beste.

8 Lösung

8.1 Klassifizierung

Ziel

Das Ziel der Arbeit ist einen Algorithmus zu finden, der Geberit bei der Klassifizierung von `etim_ids` unterstützt und den grösstmöglichen Mehrwert bietet.

Top-3-Algorithmus

Aus der Rücksprache mit Geberit ging hervor, dass vor allem ein Interesse am Top-3 Algorithmus besteht. fastText stellt für Geberit die eindeutige Wahl dar. FastText liefert in der Top-1 Version bereits die besten Resultate mit einer Genauigkeit von 96.35% und bietet in der Top-3 Version neben potenziell verbesserten Resultaten einen noch grösseren Mehrwert. Es erlaubt die Kombination von maschineller Vorarbeit und einem finalen menschlichen Entscheid mit einer obere Genauigkeitsgrenze von 98.57%.

Empfehlung

Die Empfehlung der Autoren ist der fastText Top-3-Algorithmus.

8.2 Auswertung

Lernen

Alle Hyperparameter wurden in den vorherigen Kapiteln trainiert und sind somit für die jeweiligen Sub Sets optimiert. Eine Überprüfung der momentanen Modelle der Algorithmen macht noch keinen Sinn, da es eine signifikante Anzahl von `etim_ids` im `final_test_set` gibt, auf denen durch die mehrfachen Splits niemals trainiert wurde. Deshalb werden alle Modelle mit denselben Parametern nochmal trainiert, aber auf allen Daten, welche nicht im `final_test_set` sind.

Datenset

Die finale Auswertung wird auf dem unberührten `final_test_set` durchgeführt.

Die neuen Modelle der jeweiligen Algorithmen werden mit den durchs Training optimierten Hyperparameter auf allem ausser `final_test_set` trainiert.

Für den Lookup und den fastText Algorithmus wurde dafür das `final_prepared_and_formatted_data_train` Set zusammengestellt und für das DNN das `final_prepared_data_train` Set. Diese beiden Sets tragen dieselben Informationen und unterscheiden sich nur in der Daten Aufbereitung für die jeweiligen Algorithmen.

EC000000

Da die Modelle bewusst '`EC000000`' nicht ausgeben, sind diese Datensätze alle mit Sicherheit falsch. Die Anzahl Datensätze im `final_test_set` ist 250 und bilden somit einen Anteil von 9.679%

Unbekannte etim_ids

Abgesehen von '`EC000000`' gibt es auch andere `etim_ids`, welche durch die zufällige Partition im `final_test_set` sind, aber beim Trainieren nie angetroffen werden. Dies sind 6 oder 0.232%, welche ebenfalls automatisch falsch klassifiziert werden.

Metriken

Zur finalen Bewertung der Algorithmen wird jeweils die Top-1-Accuracy und die Top-3-Accuracy folgender Daten erhoben.

1. Das `final_test_set`: Dieses Set gibt pessimistische Metriken zurück, welche definitiv nicht zu Gunsten des Algorithmus manipuliert wurde.
2. Das `final_test_set` ohne `'EC000000'`: Diese Metriken und insbesondere die Top-3-Accuracy stellen der Meinung der Autoren nach das genauste Analog dazu dar, wie gut der Algorithmus, in der Praxis, auf noch nicht klassifizierten Daten, abschneidet.
3. Das `final_test_set` ohne `'EC000000'` und unbekannte `etim_ids`: Dieses Set gibt einem die höchstmöglichen Metriken zurück für ungesehene Daten.

Resultate

Der fastText Algorithmus gibt folgende Metriken:

Resultat	Top-1	Top-3
final_test_set	0.868	0.887
final_test_set ohne 'EC000000'	0.961	0.982
final_test_set ohne 'EC000000' und unbekannte etim_ids	0.963	0.985

Das DNN gibt folgende Metriken:

Resultat	Top-1	Top-3
final_test_set	0.835	0.872
final_test_set ohne 'EC000000'	0.925	0.966
final_test_set ohne 'EC000000' und unbekannte etim_ids	0.927	0.968

Der Kombinationsalgorithmus aus Lookup Table und DNN gibt folgende Metriken:

Resultat	Top-1	Top-3
final_test_set	0.839	0.872
final_test_set ohne 'EC000000'	0.928	0.966
final_test_set ohne 'EC000000' und unbekannte etim_ids	0.931	0.968

8.3 GUI

8.3.1 Funktionalität

Allgemein

Das GUI wurde mit Hilfe des `tkinter` Python Paket erstellt. Es bietet die Möglichkeit die trainierten Modelle der Algorithmen in ihren Variationen auf geladenen Daten anzuwenden. Der angezeigte Datensatz widerspiegelt stets den resultierenden Datensatz nach einer ausgewählten Aktion.

Laden

Das Tool bietet die Möglichkeit einen Datensatz als csv Datei direkt einzulesen und zu betrachten. Diese Funktionalität wird durch das `pandastable` Paket zur Verfügung gestellt.

Überprüfung

Durch die Auswahl «check Classification» wird eine Überprüfung der geladenen Datensätze initiiert. fastText Top-1 übernimmt diese Aufgabe als Klassifizierungsalgorithmus. Der resultierende Datensatz besteht aus den Daten, bei denen die Klassifizierung eine andere `etim_id` erwartet hätte.

Die Idee hierbei ist, dass somit schnell und übersichtlich potenziell falsch klassifizierte Daten identifiziert werden können.

Lookup «Lookup» klassifiziert die Daten mit Hilfe des Lookups Algorithmus.

DNN «DNN» klassifiziert die Daten mit Hilfe des DNN Top-1-Algorithmus.

fastText «fastText» klassifiziert die Daten mit Hilfe des fastText Top-1-Algorithmus.

Lookup_DNN «Lookup_DNN» klassifiziert die Daten mit Hilfe des Lookup_DNN Kombinations Algorithmus.

DNN Top 3 «DNN Top 3» klassifiziert die Daten mit Hilfe des DNN Top-3-Algorithmus.
Hierbei wird jeweils ein zusätzliches Fenster geöffnet, um die Benutzerauswahl entgegenzunehmen.

fastText Top 3 «fastText Top 3» klassifiziert die Daten mit Hilfe fastText Top-3-Algorithmus.
Hierbei wird jeweils ein zusätzliches Fenster geöffnet, um die Benutzerauswahl entgegenzunehmen

Speichern Nachdem die jeweilige Funktion durchgeführt wurde, gibt es die Möglichkeit den resultierenden Datensatz wieder abzuspeichern. «Save as…» speichert den sichtbaren Datensatz an beliebiger Stelle ab.

Kontrolle Der Prototyp erlaubt minimale Kontrolle über die Daten. Top-1-Algorithmen befüllen den Datensatz nach ihrem besten Wissen und Gewissen. Top-3-Algorithmus erlauben etwas mehr Interaktion. Dabei kann beispielsweise die jeweilige Reihe ausgewählt werden, auf der eine Klassifizierung gewünscht wird. Dies ist vor allem in Kombination mit «check Classification» nützlich, da man dabei gleich Vorschläge für potenziell falsch klassifizierte Daten erhalten und diese korrigieren kann.

Log Ein weiteres nützliches Feature ist die Log Funktionalität. Falls bei einem Top-3-Algorithmus der Benutzer nicht diejenige Auswahl trifft, die der Algorithmus als die Wahrscheinlichste ansieht, wird der gesamte Datensatz inklusive der durch den Benutzer ausgewählten und der wahrscheinlichsten `etim_id` in einem Log abgespeichert. Dies erlaubt eine Analyse, in welchen Daten sich der Algorithmus und ein Benutzer regelmässig widersprechen und erlaubt damit mögliche Veränderungsansätze und kostbare Erkenntnisse.

Showcase

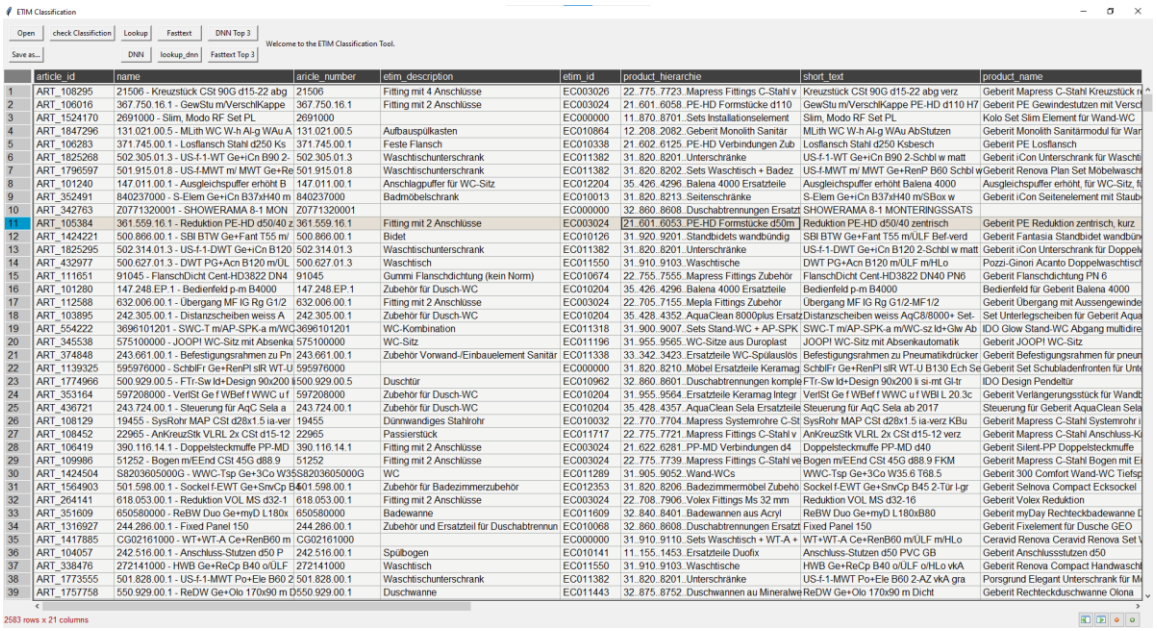

Abbildung 10: GUI Vorschau

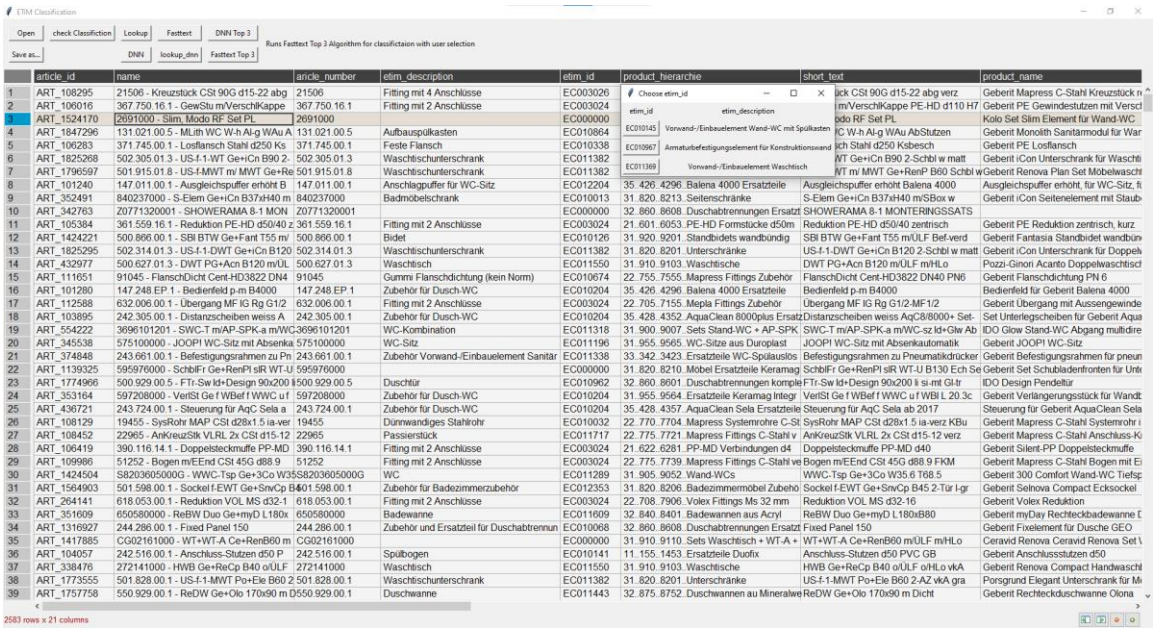

Abbildung 11: GUI Vorschau mit Auswahl

9 Bewertung, Methodenreflexion, Empfehlungen

Bewertung

Sowohl der Industriepartner als auch die Studenten sind mit dem resultierenden Algorithmus zufrieden. Neben einem anwendbaren Model wird ein Prototyp abgeliefert, der eine effiziente Klassifizierung ermöglicht. Durch die simple Abhandlung der Klassifizierung eines Produkts in wenigen Klicks wird der Aufwand eines Sachbearbeiters und die potenzielle menschliche Fehlerrate reduziert.

Das Ziel der Verbesserung der ETIM Klassifizierung durch maschinelles Lernen ist erfüllt.

Methodenreflexion

Microsoft Teams	Pandemie bedingt wurden Meetings jeweils über Microsoft Teams gehalten. Daraus ergab sich natürlich, dass Sharepoint für das Verwalten der Notebooks Modelle und Dokumentation benutzt wurde. Dies war insbesondere beim Schreiben des Berichts sehr angenehm, da man Änderungen der anderen Kollaboratoren in Real Time sieht. Ein Nachteil, dieses Systems ist Versionierung. Sicher zu stellen, dass alle Beteiligten Parteien, immer dieselben Versionen trainierter Modelle haben ist schwierig. Das Rekonstruieren älterer Versionen von Notebooks oder Modellen war mit Mehraufwand verbunden.
Jupyter Notebooks	Jupyter Notebooks ermöglichten es, im Python Code direkt mit Markdown zu dokumentieren, was die Zusammenarbeit stark erleichterte. Des Weiteren ermöglichen sie einzelne Codeblöcke zu ändern und laufen zu lassen, ohne dabei zeitaufwändige runtime Operationen wie zum Beispiel das Laden von Daten oder trainieren von Models zu wiederholen.
Tensorflow/DNN	Der Grund warum ein DNN mit One-Hot-Encoding als Ansatz gewählt wurde, war das Interesse der Studenten ein Neurales Netzwerk in der Arbeitswelt anzuwenden. Der Ansatz war rückblickend im Vergleich von fastText für die Problemstellung jedoch nicht effizient genug.
fastText	Nach der Datenanalyse war klar, dass ein Textembedding ein vielversprechender Ansatz war. Es gäbe diverse Alternativen zu fastText, welche voraussichtlich ähnlich gute Resultate produziert hätten. An fastText war ausschlaggebend, dass es ‘lightweight’, ‘selfcontained’ und 'opensource’ ist.

Empfehlungen

Gebrauch GUI	Das fastText Model ist so gut trainiert, dass der Gerbrauch davon über das GUI empfohlen wird, um Artikel zu klassifizieren. Die anderen Algorithmen sind um einen so grossen Faktor schlechter, dass es sich nicht lohnt, etwas anderes als fastText einzusetzen. Zu beachten ist, dass das GUI die Resultate der Arbeit darstellen soll und daher alle Algorithmen und ihre Kombinationen enthält.
EC000000	Es wäre sinnvoll, mit dem GUI über alle Artikel der Datenbank zu gehen, welche momentan ‘EC000000’ zugeordnet sind.

	Dadurch würden die Einträge eliminiert, welche nicht zugeordnet wurden und für den Algorithmus als Human Error angesehen werden. Andererseits könnte das Model inklusive den 'EC000000' Einträgen neu trainiert werden. Somit würde der Algorithmus ebenfalls in Mapping auf 'EC000000' lernen und dieses vorschlagen, wenn die Wahrscheinlichkeit hoch ist, dass dem Produkt keine ETIM Klasse zugeordnet werden kann.
Verbesserung des fastText Models	Die Logfunktion des GUIs ermöglicht es, Statistiken darüber zu erschliessen, ob das Model mit der Zeit schlechter wird. Zum Beispiel falls der endgültige Entscheid des Benutzers nicht der wahrscheinlichsten Vorhersage des Models entspricht. Diese Informationen könnten genutzt werden, um am Algorithmus Verbesserungen vorzunehmen. Ein alternativer Ansatz wäre das periodische Trainieren der Modelle auf der vollumfänglichen Datenbank.

10 Literatur und Quellenverzeichnis

Herkunft der Vorlage

Das Dokument wurde auf der Basis einer Vorlage für Technische Berichte erstellt. Die Vorlage ist ein Element des „Werkzeugkastens Technische Berichte“ der Hochschule für Technik Rapperswil. Sie orientiert sich an Prinzipien des Strukturierten Schreibens.

10.1 Quellenverzeichniss

[1] "ETIM Schweiz," [Online]. Available: https://www.etim.ch/de/. [Accessed 28 Mai 2021].

10.2 Abbildungen

10.3 Tabellen

Anhang

I Auswertung der reservierten Daten für die Bewertung

Allgemein

Hier ist die Auswertung mit den 1000 Datensätzen, welche am Anfang zur Bewertung der Arbeit entfernt wurden.

Vorgehen

Das Vorgehen hier ist analog zum Testen mit dem `final_test_set` und es werden dieselben Metriken erhoben.

Resultate

Der fastText Algorithmus gibt folgende Metriken:

Resultat	Top-1	Top-3
final_test_set	0.882	0.900
final_test_set ohne 'EC000000'	0.967	0.987
final_test_set ohne 'EC000000' und unbekannte etim_ids	0.969	0.989

Das DNN gibt folgende Metriken:

Resultat	Top-1	Top-3
final_test_set	0.843	0.876
final_test_set ohne 'EC000000'	0.924	0.961
final_test_set ohne 'EC000000' und unbekannte etim_ids	0.926	0.963

Der Kombinationsalgorithmus aus Lookup Table und DNN gibt folgende Metriken:

Resultat	Top-1	Top-3
final_test_set	0.846	0.875
final_test_set ohne 'EC000000'	0.928	0.959
final_test_set ohne 'EC000000' und unbekannte etim_ids	0.930	0.962

II Bedienungsanleitung GUI

Allgemein

Das Tool wurde nicht als eigenständige Applikation zur Verfügung gestellt. Es werden lediglich die gesamten Python Files und Skripts bereitgestellt.

Startup

Um das graphische Tool zu starten:

1. Navigiere in den Ordner `prototyp_gui`
2. Installiere `requirements.txt`

```
pip install -r requirements.txt
```

3. Starte das Programm

```
python application.py
```

Updaten der Modelle

Das Tool lädt alle Modelle aus `prototyp_gui/algorithms/models`.
Falls neue Modelle zur Verfügung stehen, genügt ein Austauschen der Modelle im oben genannten Ordner.

III Abgabeordner

Allgemein Der Abgabeordner enthält jegliche Daten und Algorithmen, welche für die Arbeit benutzt wurden.

Top Level

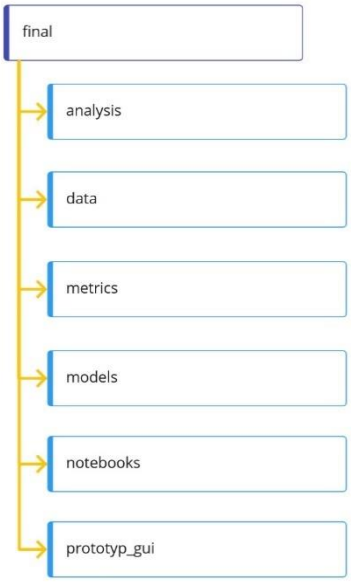

Abbildung 12 erstellt mit http://www.miro.com

Analysis Enthält Daten und Metriken aus der Datenabalyse.

Data

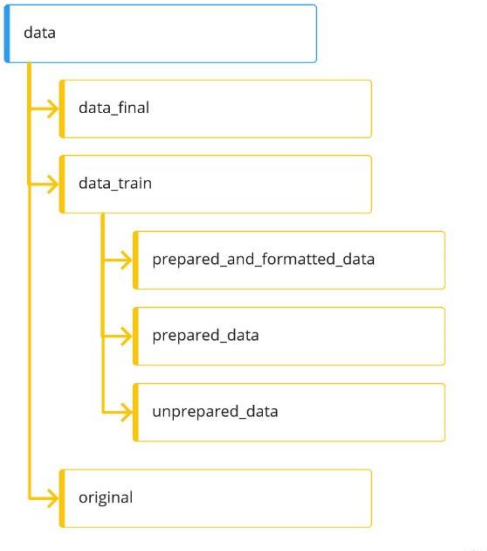

Abbildung 13 erstellt mit http://www.miro.com

Der Data Unterordner enthält alle benutzen Datensätze. Diese sind aufgeteilt nach `original`, `data_final` und `data_train`. Im `data_train` befinden sich die jeweiligen Unterteilungen der Daten nach der Datenaufbereitung.

Metrics

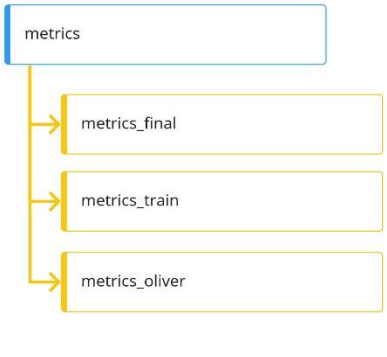

Abbildung 14 erstellt mit http://www.miro.com

Die Metriken enthalten alle Metriken der verschiedenen Algorithmen.

`metrics_train` enthält die Metriken der Trainingsalgorithmen auf dem `prepared_data_validate`.

`metrics_final` enthält die Metriken der finalen Algorithmen auf dem `final_test_set`.

`metrics_oliver` enthält die Metriken der finalen Algorithmen auf den Daten, die vor dem Beginn der Arbeit bereits abgetrennt wurden.

Models

Abbildung 15 erstellt mit http://www.miro.com

Die Models enthalten die jeweiligen Versionen der Algorithmen.

`models_train` enthält die Modelle aller Algorithmen, die auf dem `prepared_data` oder einer Variation davon trainiert sind.

`models_final` enthält alle Modelle aller Algorithmen, die auf allen Daten ausser dem `final_test_set` trainiert sind.

Notebooks

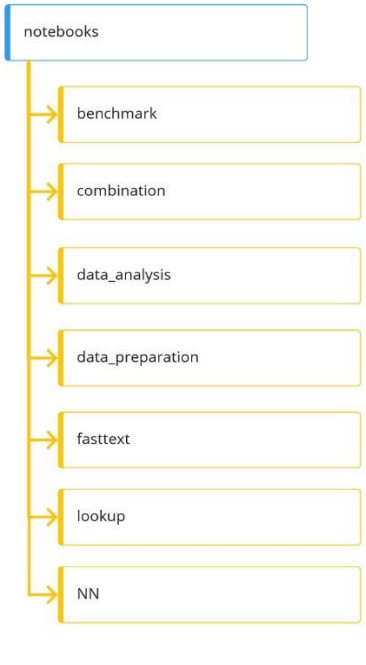

Abbildung 16 erstellt mit http://www.miro.com

Die Notebooks enthalten alle Jupyter Notebooks, die während der Arbeit erstellt wurden.

Prototyp GUI

Der Prototyp GUI Ordner enthält die Python Anwendung.

IV Management Summary

ETIM Klassifikation mit ML-Approach

Vorhersage der ETIM-Klasse basierend auf Daten aus der Geberit Datenbank

Autoren:	Etienne Baumgartner, Nathanael Gall
Betreuer:	Prof. Oliver Augenstein
Experte:	Nico Schmid
Themengebiet:	Machine Learning
Industriepartner:	Geberit AG, Jona

Ausgangslage

Das ETIM Klassifikationsmodel setzt sich auf dem internationalen Markt immer mehr durch. Die ETIM Klassifikation vereinfacht einerseits den Datenaustausch zwischen Händler Hersteller sowohl als auch die Klassifizierung eines Produkts im internationalen Elektrotechnik Fachbereich. Die Geberit AG nutzt diesen Standard für ihren Produktekatalog und die Klassifizierung wird manuell durch einen Sachbearbeiter vorgenommen.

Gesucht sind Möglichkeiten den Klassifizierungsprozess der Produkte mit Hilfe von maschinellem Lernen einerseits zu beschleunigen und anderseits potenzielle menschliche Fehler zu vermeiden.

Vorgehen / Technologie

Die Suche nach passenden Lösungen verlangt die genaue Analyse der bereits vorhanden, manuell klassifizieren Produktdaten. Spezifisch wird eine Analyse auf den Daten durchgeführt, um allgemein gültige Charakteristiken zu extrahieren. Um die Eigenschaften und gewonnen Informationen vollumfänglich auszunutzen, wird im zweiten Schritt eine Datenaufbereitung vorgenommen.

Auf der Basis der verarbeiteten Produktdaten werden zwei Modelle aufgebaut.

One-Hot-Encoding:
Die Daten werden für das Trainieren auf einem Deep Neural Network One-Hot-Encoded. Um die Anzahl Dimensionen des Netzwerks möglichst klein zu halten, wird anhand des Bayes Errors die optimale Spaltenkombination ermittelt. Auf dieser Auswahl wird ausserdem ein Lookup Table erstellt, um eindeutige Datensätze direkt zu klassifizieren

Textembedding:
Die hohe Anzahl und der hohe Informationsgehalt in den textbasierten Spalten ermöglichen den Aufbau einer Text Klassifizierung. Die Produkte werden durch den fastText Algorithmus und anhand von Textembeddings zugeordnet.

Ergebnisse

Nach der Optimierung dieser zwei Ansätze stellt sich heraus, dass der textbasierte fastText Algorithmus genauere Resultate liefert. Es wurde eine Pythonanwendung geschrieben, welche fastText benutzt, um dem Nutzer für jeden Datensatz eine Auswahl von drei potenziellen ETIM Klassen auszugeben. Die Wahrscheinlichkeit, dass sich die korrekte ETIM Klasse in dieser Auswahl befindet ist 98.2%.

Ausblick

Wenn man diese Anwendung zur Klassifizierung benutz, kann man das Model in Zukunft noch verbessern.

Vom Algorithmus abweichende Entschlüsse des Sachbearbeiters können analysiert werden und bieten die Möglichkeit das Model basierend auf den Entscheidungen des Benutzers entsprechend anzupassen.