Zürcher Hochschule
für Angewandte Wissenschaften

School of
Engineering

InIT Institut für
angewandte Informationstechnologie

MSE | MASTER OF SCIENCE
IN ENGINEERING

ADRESSBEREINIGUNG MIT OPEN DATA DER SCHWEIZER POST

PA1

Severin Holzer

Advisor: Kurt Stockinger

30. April 2020

Inhaltsverzeichnis

1 Einleitung

Viele Unternehmen besitzen Adressdaten über ihre Stakeholder, um Rechnungen oder sonstige Briefe und Pakete zu senden. Die Adressen werden über Formulare via Internet (digital), via Post (handschriftlich) oder über das Telefon übermittelt. Anschliessend übertragen die Sachbearbeiter/innen die Daten in das entsprechende System. Bei diesem Prozess kann es in den entsprechenden Schritten zu Fehlern kommen. Beispielsweise können Tippfehler, unleserliche Schriften oder Kommunikationsprobleme auftreten. Dies kann zu schlechter Qualität der Adressen im System führen.

Das Problem kann als Entity Matching Problem adressiert werden, das heisst, wenn die Daten unterschiedlich geschriebene Adressen enthalten, kann man sie einander nicht zuordnen. Das Ziel dieser Arbeit ist es, einen End-to-End Prozess aufzubauen, mit welchem möglichst viele Adressen der Rohdaten mit jenen aus den Referenzdaten verknüpft werden können. Kurz gesagt eine Adressbereinigung soll erfolgen. Um dies realisieren zu können, wurde der Prozess mit Python in drei Teilschritte gegliedert. Als erstes wurden die Daten geladen (Part 1), danach folgte die Vorbereitung (Part 2) und am Ende erfolgte das Matchen (Part 3). Zum Abschluss werden zwei Methoden gegenübergestellt, um den Prozess effizienter zu gestalten und die Anzahl benötigter Trainingsdaten zu verkleinern.

Diese Arbeit beschreibt zuerst die benötigten Datensätze (Referenz- und Testdatensatz). Anschliessend folgt die Beschreibung der Teilschritte des End-to-End Prozesses der Adressbereinigung, auf der Grundlage von echten Adressdaten.

2 Datensätze

Um eine Adressbereinigung zu ermöglichen, benötigt man einen Referenzdatensatz, welcher die korrekten Adressen enthält. Ausserdem einen Testdatensatz, bei welchem die Bereinigung durchgeführt werden soll.

2.1 Referenzdaten

Als Referenzdaten dienten die Daten aus dem Open Data Portal der Schweizer Post [1]. Auf diesem Portal werden grosse Mengen von unpersönlichen Informationen frei zur Verfügung gestellt und in regelmässigen Abständen aktualisiert. Für dieses Projekt wurden die Daten aus dem Strassenverzeichnis verwendet. Die Post stellt zusätzlich eine Beschreibung des Datenmodells [2] zur Verfügung. Um einen besseren Einblick zu erhalten, folgt im nächsten Abschnitt ein kurzer Überblick über dieses Modell und die Beziehungen zwischen den einzelnen Datensätzen.

2.1.1 Datenmodell Strassenverzeichnis

Die Post führt eine Datenbank mit sämtlichen Strassen-, Weiler- und Flurbezeichnungen in der Schweiz und Liechtenstein. Ausserdem enthält es Informationen zu den postalisch bedienten Gebäuden. Die Daten bilden eine gute Referenz bei der Adressbereinigung und es können noch weitere Eigenschaften gewonnen werden (Beispiel: Kanton, BFS-Gemeindenummer[1], Agglomerationsnummer usw.). Gemäss Dokumentation sind 10 Datensätze verfügbar, welche durch einen Primärschlüssel (PK) und einen Fremdenschlüssel (FK) miteinander verknüpft werden können. Eine kurze Beschreibung ist in der folgenden Tabelle vorhanden (**Hinweis**: Durchgestrichen bedeutet, dass die Daten auf dem Open Data Portal nicht vorhanden sind.):

Record-Art	Inhalt	Beschreibung
~~00~~	~~NEW_HEA~~	~~Enthält das Versionsdatum und einen eindeutigen Zufallscode.~~
01	NEW_PLZ1	Enthält alle für die Adressierung gültigen Postleitzahlen der Schweiz und des Fürstentums Liechtenstein.
02	NEW_PLZ2	Enthält alternative Ortsbezeichnungen und Gebietsbezeichnungen zur jeweiligen Postleitzahl.
03	NEW_COM	Enthält die politischen Gemeinden der Schweiz und des Fürstentums Liechtenstein. Diese Daten stammen aus der offiziellen Liste des Bundesamtes für Statistik (BFS).
04	NEW_STR	Enthält alle Strassenbezeichnungen aller Ortschaften der Schweiz und des Fürstentums Liechtenstein.
05	NEW_STRA	Logische alternative oder fremdsprachige Strassenbezeichnung zur offiziellen Strassenbezeichnung. Gebäudebezeichnungen ohne

[1] Das Bundesamt für Statistik (BFS) führt für jede Gemeine in der Schweiz eine Gemeindenummer. Als Beispiel steht die Nummer 261 für die Gemeinde Zürich.

		Strasse/Hausnummer, Gebiets-, Flur- oder Weilerbezeichnungen werden wie Strassennamen behandelt.
06	NEW_GEB	Enthält Hausnummer und Hauskey.
07	NEW_GEBA	Enthält alternative Hausbezeichnung und alternativen Hauskey.
08	NEW_BOT_B	Enthält Boteninformationen auf Stufe Hausnummer (Briefzustellung).
~~12~~	~~NEW_GEB_COM~~	~~Verknüpfung zwischen Gebäude und Gemeindeinformationen~~

Tab. 1: Datensätze mit Beschreibung [2].

Um die Beziehungen besser zu verstehen, wurde das Datenmodell in der nächsten Abbildung visualisiert (**Hinweis**: Ein Kreuz bedeutet, dass die Datensätze in dieser Arbeit nicht berücksichtigt wurden.):

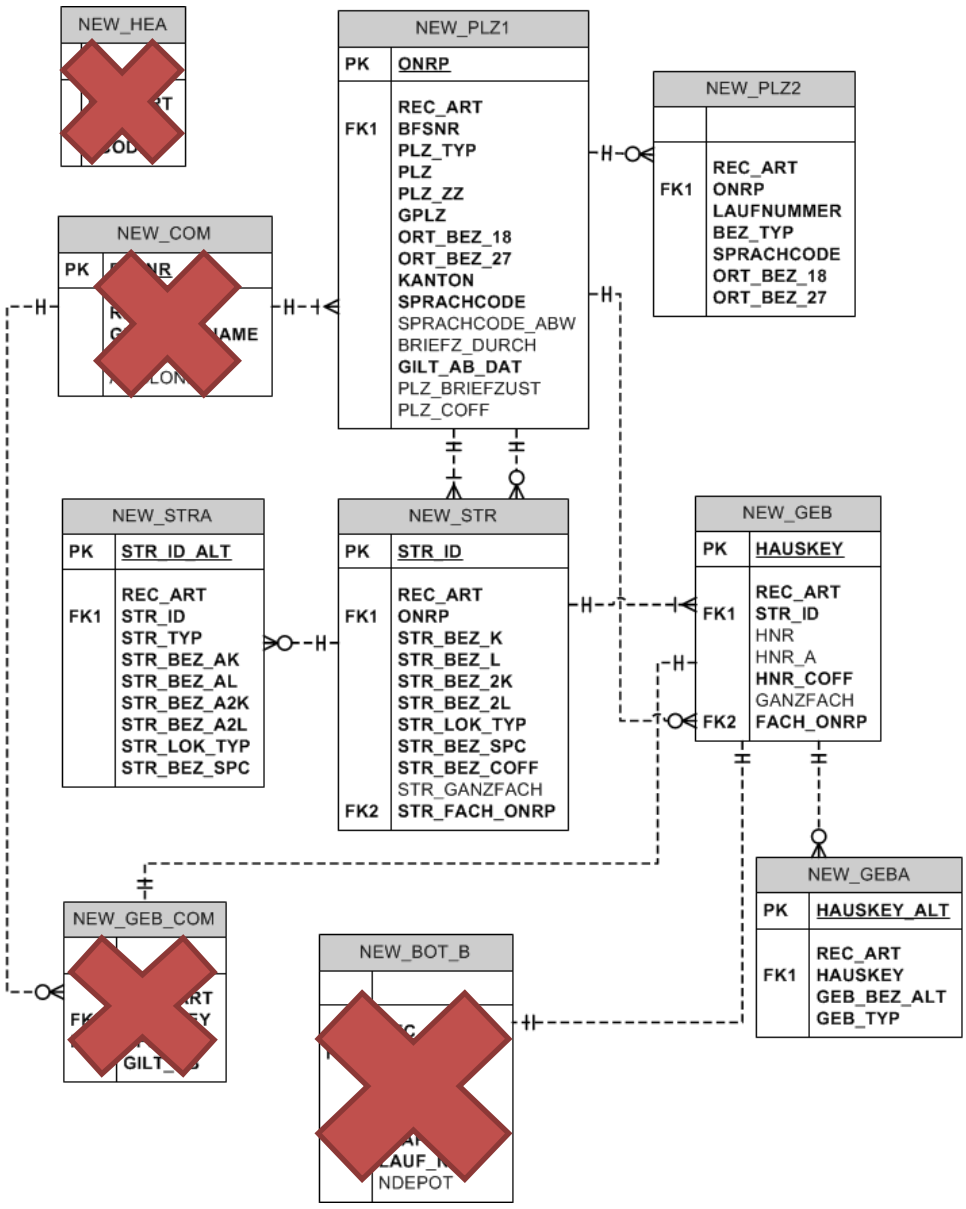

Abb. 1: Datenmodell mit den Beziehungen (PK, FK) zwischen den Datensätzen [2].

Als erster Schritt wurden die Daten zu einem einzigen Datensatz verknüpft, um eine gute Grundlage für die Adressbereinigung zu erhalten. Nähere Details folgen im nächsten Abschnitt.

2.1.2 Kombinierter Referenzdatensatz

Als erster Schritt wurden die 6 Datensätze heruntergeladen und die nicht benötigten Attribute gelöscht. Anschliessend fand eine Bereinigung durch Entfernen von Duplikaten statt. Wie in der Tab. 1 beschrieben, existierte zu jedem Datensatz einer, welche jeweils alternative Angaben zu den entsprechenden Attributen besitzt. Dies ermöglichte es noch weitere Adresskombinationen zu generieren. Dadurch konnte bereits beim einfachen Inner-Join (siehe Kap. 3.1) die Anzahl nicht gefundener Matches von ca. 18'700 auf ca. 16'300 reduziert werden. Dies bedeute eine Reduktion um ca. 12.8%. Diese Verbesserung zeigte sich auch während dem Blocking (siehe Kap. 3.2), wobei ähnliche Adresspaare bereits bei den ersten Blockern gefunden wurden.

Es wurden drei Datensätze erstellt und diese am Ende miteinander kombiniert und normalisiert.

Ortsbezeichnungen (NEW_PLZ)

Mit Hilfe der Datensätze NEW_PLZ1 und NEW_PLZ2 konnte für jede Ortschaft eine kurz/lang-Bezeichnung sowie deren Alternative gewonnen werden. Dadurch wurde der Datensatz von ca. 6’100 auf ca. 8’100 Records erweitert.

ONRP	BFSNR	POSTLEITZAHL	ORTBEZ18_x	ORTBEZ27_x	ORTBEZ18_y	ORTBEZ27_y
6122	5409	1884	Arveyes	Arveyes	NaN	NaN
1298	6417	2024	St-Aubin-Sauges	St-Aubin-Sauges	St-Aubin	St-Aubin
4476	261	8057	Zürich	Zürich	Zurich 57	Zurich 57
1723	351	3015	Bern	Bern	Berna	Berna
4583	2920	8243	Altdorf SH	Altdorf SH	NaN	NaN
5272	3372	9633	Bächli (Hemberg)	Bächli (Hemberg)	NaN	NaN
8487	4289	4800	Zofingen PFinance	Zofingen PostFinance	NaN	NaN
376	6621	1209	Genève	Genève	Genf	Genf
5357	3238	9443	Widnau	Widnau	NaN	NaN
3829	5251	6877	Coldrerio	Coldrerio	Villa Coldrerio	Villa Coldrerio

Abb. 2: Verknüpfte Daten zwischen NEW_PLZ1 (Blau: ORTBEZ18_x, ORTBEZ27_x) und NEW_PLZ2 (Grün: ORTBEZ18_y, ORTBEZ27_y) über ONRP.

Für den Datensatz NEW_PLZ wurden alle Ortsbezeichnungen in der neu generierten Variable ORTBEZ gespeichert.

Strassenbezeichnungen (NEW_STRASSE)

Gleich wie bei den Ortschaften bietet die Post zu den Strassenbezeichnungen einen Datensatz mit alternativen Bezeichnungen an. Das Verknüpfen zwischen NEW_STR und NEW_STRA erhöhte den Datensatz von ca. 176'100 auf 250'700 Records.

STRID	ONRP	STRBEZK	STRBEZL	STRBEZ2K	STRBEZ2L	STRBEZAK	STRBEZAL
63576	2890	Rütiweg	Rütiweg	Rütiweg	Rütiweg	Motzet	Motzet
54877	3172	Fliederweg	Fliederweg	Fliederweg	Fliederweg	NaN	NaN
40070	3994	Bottaholds	Bottaholds	Bottaholds	Bottaholds	NaN	NaN
85324	5396	Hellwies	Hellwies	Hellwies	Hellwies	NaN	NaN
51026711	5259	Steinstrasse	Steinstrasse	Steinstrasse	Steinstrasse	Tobelstrasse	Tobelstrasse
51529	2987	Kirchgasse	Kirchgasse	Kirchgasse	Kirchgasse	NaN	NaN
51001707	5253	Bruggli	Bruggli	Bruggli	Bruggli	NaN	NaN
88503	3421	Grodstrasse	Grodstrasse	Grodstrasse	Grodstrasse	NaN	NaN

Abb. 3: Verknüpfte Daten zwischen NEW_STR (Blau: STRBEZK, STRBEZL, STRBEZ2K, STRBEZ2L) und NEW_STRA (Grün: STRBEZAK, STRBEZAL, STRBEZA2K, STRBEZA2L) über STRID und ONRP.

Im neuen Datensatz NEW_STRASSE wurden diese Einträge in die neu generierte Variable STRBEZ geschrieben.

Gebäudebezeichnungen (NEW_GEBAEUDE)

Im Datensatz NEW_GEB waren die Hausnummern gespeichert. Zusätzlich enthielt NEW_GEBA Hausbezeichnungen, welche später beim Kombinieren aller Datensätze in die Strassenbezeichnung eingefügt wurden. In diesem Fall fand keine Vermehrung der Records statt, da es sich nicht um alternative Hausnummern handelte.

HAUSKEY	STRID	HNR	HNRA	GEB_BEZ_ALT
58067425	58009147	20.0	NaN	NaN
76464351	76155408	9.0	A	NaN
50042884	23204	19.0	NaN	NaN
58138018	58070809	11.0	NaN	NaN
50213558	91580	9.0	NaN	NaN
76299452	31669	20.0	NaN	NaN
8087025	76124037	7.0	NaN	Chalet St.Odile
76311599	76120309	4.0	NaN	NaN

Abb. 4: Verknüpfte Daten zwischen NEW_GEB (Blau: HNR, HNRA) und NEW_GEBA (Grün: GEB_BEZ_ALT) über STRID und HAUSKEY.

Der neue Datensatz NEW_GEBAEUDE enthielt schlussendlich ca. 1’872’300 Einträge.

Alle Adressen (AdressesCH)

Über die Verknüpfung zwischen den Variablen ONRP und STRID konnten die drei vorbereiteten Datensätze NEW_PLZ, NEW_STRASSE und NEW_GEBAEUDE miteinander kombiniert werden. Da auch Adressen vorkommen konnten, bei welchen nicht alle Angaben verfügbar waren, wurden zusätzlich Records generiert, in welchen nur die Ortsbezeichnungen vorhanden waren. Dadurch wurde es ermöglicht, Adressen bei denen Strassen nicht zugeordnet werden konnten, einer Referenzadresse zuzuweisen. Als Resultat entstand der neue Datensatz AdressesCH, welcher in dieser Arbeit als Referenzdatensatz für die Adressbereinigung benötigt wurde.

ONRP	BFSNR	POSTLEITZAHL	ORTBEZ	STRID	STRBEZ	HAUSKEY	HNR	HNRA
680	2054	1470	Estavayer-le-Lac	90338.0	Chasseral, ch. du	NaN	NaN	NaN
583	5749	1373	Chavornay	15072945.0	Chemin du Vieux Moulin	15172954.0	2.0	NaN
1568	6730	2826	Corban	30021953.0	Eglise, rue de l'	30034707.0	4.0	NaN
713	5669	1521	Curtilles	76128339.0	Vagnaires, chemin des	76375599.0	5.0	NaN
1165	6021	1957	Ardon	15008312.0	Avenue Neuve	76359996.0	29.0	NaN
1135	5409	1885	Chesières	76132847.0	Les Olympiades	15095250.0	76.0	NaN
4388	261	8004	Zurich	64825.0	Zypressenstrasse	57009976.0	56.0	NaN
3230	1024	6020	Emmenweid	75927.0	Schönbühlring	15190.0	13.0	NaN
4795	297	8493	Blitterswil	58006984.0	Campingplatz	58155718.0	11.0	NaN
1111	5411	1865	Col-du-Pillon	76144599.0	PB Rochers	76397643.0	1117.0	NaN

Abb. 5: Referenzdatensatz AdressesCH mit allen Adressen der Schweiz und Liechtenstein.

Der kombinierte Datensatz enthielt alle Adressen (6’184'000 Records) der Schweiz und Liechtenstein.

2.2 Testdaten

Als Testdatensatz konnten echte Daten verwendet werden. Insgesamt waren ca. 1'023'700 Einträge vorhanden, welche einer Adresse im Referenzdatensatz zugeordnet werden sollten.

FzStammnummer	FhStrasse1	FhHausNr1	FhHausNr2	FhPLZ1	FhOrt1	BfsNummerCd	FhOrtLand
315711162	Albisstrasse	11	NaN	8800	Thalwil	141	CH
400031721	Neuguetstrasse	4	NaN	8618	Oetwil am See	157	CH
682243818	Am Aepplihoger	14	NaN	8816	Hirzel	295	CH
183658670	Gerichtshausstrasse	3	NaN	8340	Hinwil	117	CH
600842687	Höhestrasse	40	NaN	8702	Zollikon	161	CH
647984319	Dorfstrasse	7	C	8471	Rutschwil (Dägerlen)	214	CH
612347062	Im Wisli	14	NaN	8805	Richterswil	138	CH
323679460	Moosmattstrasse	16	a	8953	Dietikon	243	CH

Abb. 6: Testdatensatz für die Adressbereinigung.

2.3 Normalisierung

Damit die beiden Datensätze miteinander verglichen werden konnten, mussten sie dieselbe Struktur und Eigenschaften besitzen. Aus diesem Grund wurde eine Funktion definiert, mit welcher die beiden Datensätze normalisiert wurden. Es kamen die folgenden Regeln zur Anwendung:

Regel	Erläuterung
NaN-Werte mit leeren Werten auffüllen (dtype=object)	Um später die Zusammenführung zu einer neuen Variablen zu ermöglichen (sonst erscheint der Wert NaN im neu generierten String).
NaN-Werte mit 0 auffüllen (dtype≠object)	Damit das Programm erkennt, dass keine Hausnummer vorhanden ist und trotzdem Differenzen gebildet werden können.
Kleinschreibung auf alle Strings anwenden	Die Ähnlichkeit zwischen Texten wird dadurch erhöht.
Erzeugen einer eindeutigen ID (Laufnummer)	Nötig für einzelnen Programmcode in Python, um eine Eindeutigkeit zu erkennen.
Neues Attribut FhHausNrGes1/ HNRGes, welches die Hausnummer und den Zusatz enthält	Bei gewissen Adressen steht die Hausnummer im Zusatz. Zusätzlich können Datensätze geladen werden, bei welchen die Hausnummer und der Zusatz zusammen in einem Feld stehen.

Tab. 2: Beschreibungen und Erläuterungen der angewendeten Regeln bei der Normalisierung.

Durch die Normalisierung wurden die beiden Datensätze AdressesCH_norm_nodup und FZ_norm_nodup erzeugt. Diese bildeten die Grundlage für das Data Matching (siehe Kap. 3).

ONRP	BFSNR	POSTLEITZAHL	ORTBEZ	STRID	STRBEZ	HAUSKEY	HNR	HNRA	HNRGes	Ad_ID
1096	5407	1854	leysin	76147331	rollier, avenue	15085115	9		9	1156357
515	6644	1290	versoix	48546	yung, avenue louis-	76020380	41	a	41a	1062104
3456	1402	6390	engelberg	21750	am dürrbach	80933	6	a	6a	2486677
1112	5409	1867	villy vd	15065684	le crêtel d'antagnes	0	0			5969110
1430	371	2504	biel/bienne	44433	hintergasse	29045486	39		39	296466
968	2206	1723	marly	48903	pfaffenwil, route de	15019135	9		9	172699
3426	1702	6330	heiligkreuz b. cham	928436	hallenbad	197929	0			5452041
2170	567	3713	kien	76125467	haltenrain	0	0			5983744
150	5586	1003	losanna	30379	terreaux, rue des	0	0			6009373
3102	4068	5607	hägglingen	53361	rebberg	34190579	5		5	544225

Abb. 7: Referenzdatensatz (AdressesCH_norm_nodup) nach der Normalisierung.

In den beiden Abb. 7 und Abb. 8 kann die angewendete Normalisierung begutachtet werden. Bei den Textattributen sieht man die Texte kleingeschrieben. Zusätzlich wurden unbekannte Hausnummern (NaN) durch die Zahl 0 ersetzt. Als Gesamthausnummer wurden die Attribute HNRGes und FhHausNrGes1 aus der Hausnummer und dem

dazugehörigen Zusatz erstellt. Die Variable Ad_ID bildet die eindeutige ID, welche als Primärschlüssel verwendet wurde.

FzStammnummer	FhStrasse1	FhPLZ1	FhOrt1	BfsNummerCd	FhHausNrGes1
640335993	müllerwis	8335	hittnau	173	1
599338585	unter-hilti	8444	henggart	31	8
184645865	hohmattstrasse	8173	neerach	88	19
220369020	neuguetstrasse	8624	grüt (gossau zh)	115	7
219843628	bahnhof zürich oerlikon	8050	zürich	261	
411767851	st gallerstrasse	8404	winterthur	230	180
200308698	watterstrasse	8156	oberhasli	90	1a
184209500	pionierstrasse	8403	winterthur	230	7
184087248	meienhof	8421	dättlikon	215	3
649662198	juchstrasse	8192	glattfelden	58	20

Abb. 8: Testdatensatz (FZ_norm_nodup) nach der Normalisierung.

Damit der Input des Testdatensatzes flexibel bleibt, ging man davon aus, dass die Hausnummer und der Zusatz in einem Feld vorhanden sind. Aus diesem Grund wurde die neue Variable FhHausNrGes1 erstellt und die Ursprünglichen gelöscht. Falls keine Einträge für die Hausnummer und den Zusatz vorhanden waren, wurde ein leerer Inhalt für die neue Variable FhHausNrGes1 erzeugt. Als Primärschlüssel dient in dieser Tabelle die FzStammnummer.

Bei der Normalisierung der Referenzdaten wurden nochmals Duplikate entdeckt und entfernt. Gründe dafür sind, dass ein Eintrag beispielsweise als klein- und grossgeschrieben existierte. Nach dem Normalisieren führte dies zu den genannten Duplikaten, welche entfernt wurden. Am Ende beinhaltet der normalisierte Referenzdatensatz ca. 6’171'400 verschiedene Adressen. Die Anzahl im Testdatensatz änderte sich nach dem Normalisieren nicht. Verantwortlich dafür ist das Attribut FzStammnummer, welches als Schlüssel definiert war.

Die Attribute mit den BFS-Gemeindenummern (BfsNummerCd / BFSNR) werden für das Data Matching nicht verwendet, sondern nur ein Vergleich (siehe Kap. 3.1) durchgeführt.

3 Data Matching

In diesem Teil soll erklärt werden, wie die beiden normalisierten Datensätze miteinander kombiniert wurden. Das heisst jeder Eintrag im Testdatensatz soll einem im Referenzdatensatz zugordnet werden, dadurch wäre eine Bereinigung der Adressen erfolgt.

Inner-Join ➔ Blocking ➔ Pair Comparison ➔ Klassifizierung

Abb. 9: Übersicht über die einzelnen Schritte im Data Matching.

Jeder einzelne Schritt und die dazugehörigen Zwischenresultate werden in den folgenden Abschnitten erläutert.

3.1 Inner-Join

Um die bereits korrekten Adressen im Testdatensatz zu eliminieren, wurde durch die jeweilige Verknüpfung zwischen Postleitzahl, Strasse, Gesamthausnummer und dem Ort zwei neue Datensätze erstellt. Im Datensatz FZ_join_in_01 (ca. 1'007'400 Einträge) wurden die Adressen gespeichert, welche im Referenzdatensatz gefunden wurden. Das heisst die Adressen hatten eine eindeutige Übereinstimmung in den entsprechenden Attributen. Mit einem kurzen Vergleich wurden die Adressen dargestellt, welche unterschiedliche Einträge in der BFS-Gemeindenummer haben. Dabei konnten 21 Records identifiziert werden. Bei diesen fand bereits eine Bereinigung statt, da die Nummer von der Post übernommen wurde.

FzStammnummer	FhStrasse1	FhPLZ1	FhOrt1	BfsNummerCd	FhHausNrGes1	BFSNR
555333791	alte sihltalstrasse	8135	sihlwald	136	9	295
217190976	tüfweg	8044	gockhausen	261	3	191
204686798	rotbuechstrasse	8165	schleinikon	99	12	98
410701042	rotbuechstrasse	8165	schleinikon	99	12	98
599966109	chlosterwis	8427	freienstein	68	13	57
682072969	müllerwis	8606	greifensee	198	14	194
222538490	rotbuechstrasse	8165	schleinikon	99	7a	98
212321000	meisenrain	8044	gockhausen	261	61	191
618255268	meisenrain	8044	gockhausen	261	61	191
959354765	meisenrain	8044	gockhausen	261	61	191

Abb. 10: FZ_join_in_01 mit unterschiedlichen BFS-Gemeindenummern (BFSNummerCd ≠ BFSNR).

Um die Differenzen zwischen den beiden Gemeindenummern noch besser zu erläutern, betrachteten wir den rot markierten Eintrag (Abb. 10). Die Postleitzahl 8044 wird

vom Bundesamt für Statistik (BFS) teilweise der Gemeinde Zürich (261) oder Dübendorf (191) zugeordnet. So kann es schnell vorkommen, dass die Daten falsch erfasst werden (wie im Testdatensatz ersichtlich). Von der Post wurde die entsprechende Adresse richtigerweise zur Gemeinde Dübendorf (191) zugewiesen und somit wurde eine erste Bereinigung der Daten durchgeführt. Das Resultat wurde auf der Homepage Schweizer Regionen[2] überprüft. Dort sah man, dass der Ort 8044 Gockhausen zur Gemeinde Dübendorf gehört.

Der zweite Datensatz FZ_join_out_01, welcher durch den Inner-Join die nicht gefundenen Adressen herausgibt, besass ca. 16'300 Einträge.

FzStammnummer	FhStrasse1	FhPLZ1	FhOrt1	BfsNummerCd	FhHausNrGes1
647396355	bernerstr nord	8064	zürich	261	208
627382613	forrenhof	8194	hüntwangen	61	
220565210	talackerstrasse	8152	glattbrugg	66	19
400175925	c/o gemeinde bonstetten	8906	bonstetten	3	
414072690	alte landstrasse	8942	oberrieden	137	78b
680854816	seebergstrasse	8103	unterengstringen	249	1
414122553	baumackerstrasse	8050	zürich	261	46/pf
333385569	gentenwisstrasse	8332	russikon	178	523
410336078	bühlhalde	8132	egg b. zürich	192	2
184453666	pfinstweidstrasse	8005	zürich	261	31a

Abb. 11: FZ_join_out_01 mit den nicht gefundenen Adressen

Es gibt sehr viel Gründe warum die Adressen nicht gefunden wurden. Wenn man die Abb. 11 betrachtet, können beispielsweise die Strassen (bernerstr nord) falsch geschrieben oder mit dieser Beschriftung (c/o gemeinde bonstetten) offiziell nicht vorhanden sein. Es kann auch alles bis auf die Gesamthausnummer (46/pf) korrekt sein, jedoch konnte die Adresse dann auch nicht eindeutig zugewiesen werden.

Der Datensatz FZ_join_out_01, mit den nicht eindeutig identifizierten Adressen, bildete den Startpunkt für die Adressbereinigung. Mit verschiedenen Methoden wurde versucht die richtige Adresse aus dem Referenzdatensatz zu eruieren.

[2] Übersicht der BFS-Gemeindecodes: https://www.schweizer-regionen.ch/bfs/191

3.2 Blocking

Das Ziel in diesem Schritt ist es, die Berechnungszeit während dem Vergleich der Adresspaare zwischen dem Test- und Referenzdatensatz zu verkleinern. Ohne das Blocking [3] würde die Zeitkomplexität [4] O(16'300 × 6'171'400 = ~10^{11}) betragen und die Laufzeit des Prozesses massiv verlängern. Das Blocking bildet Paare und entfernt anschliessend die offensichtlich falschen Einträge über ein Ähnlichkeitsmass. Das Problem in diesem Schritt bildet die Einstellung der richtigen Grenze. Bei zu hoher Grenze könnten richtige und bei zu tiefer zu wenig falsche Paare entfernt werden. Ausserdem hatten wir mit dem Memory vom Jupyter Notebook zu kämpfen. Dieses Problem konnte mit der Reduktion der Referenzdaten (siehe unten) und dem vorgelagerten Blocker 1 (siehe Kap. 3.2.1) beseitigt werden.

Um den Referenzdatensatz stark zu reduzieren konzentrierte man sich während dem Blocking auf die Postleitzahl, den Ort und die Strasse. Es wurden die Attribute der Hausnummern und anschliessend alle Duplikate gelöscht. Dadurch ergaben sich neu ca. 540'900 Einträge und somit eine Reduktion um ca. 91.2%. Die Hausnummern wurden nach dem Blocking über ID_STR wieder angehängt und für den Pair Comparison (siehe Kap. 3.2.3) benötigt. Um den Blocker 2 (siehe Kap. 3.2.2) anwenden zu können, wurde für den Test- und Referenzdatensatz das Attribut PLZ_ORT_STR erstellt, in welchem die Postleitzahl, der Ort und die Strasse als String gespeichert waren.

ONRP	BFSNR	POSTLEITZAHL	ORTBEZ	STRID	STRBEZ	PLZ_ORT_STR	ID_STR
457	6638	1242	peney ge	48131	mandement, route du	1242 peney ge mandement route du	378174
700	5678	1510	bressonnaz	33058	caserne de moudon	1510 bressonnaz caserne de moudon	379885
1674	351	3000	berna 6	0		3000 berna 6	341274
2596	4252	4303	kaiseraugst	35003374	rosenweg	4303 kaiseraugst rosenweg	37565
5374	3271	9470	räfis	12544	bühlstrasse	9470 raefis buehlstrasse	371613
4391	261	8006	zürich	64894	gaugerstrasse	8006 zuerich gaugerstrasse	228435
3455	1402	6388	altzellen	76129732	aeschi	6388 altzellen aeschi	355677
3221	1059	6010	kriens	15892	gartenstrasse	6010 kriens gartenstrasse	45985
3970	3901	7000	cuera	39973	rossbodenstrasse	7000 cuera rossbodenstrasse	305531
4626	4801	8267	berlingen	58069698	ackerstrasse	8267 berlingen ackerstrasse	231735

Abb. 12: Der Referenzdatensatz nach dem Entfernen der Informationen über die Hausnummer. Jeder Eintrag hat einen eindeutigen Schlüssel (ID_STR) und ein neues Attribut PLZ_ORT_STR.

Wie bereits im vorhergehenden Abschnitt beschrieben, wurden zwei Blocker verwendet und zusammengefügt. Am Ende folgte das Hinzufügen der Hausnummern und nochmals ein Vergleich zwischen den gefundenen Paaren.

3.2.1 Blocker 1: Inner-Join 1

Um die Berechnungszeit und den benötigten Speicher zu reduzieren, wurde ein vorgelagerter Blocker erstellt. Dieser verwendete die Methode mit der exakten Übereinstimmung. In einem ersten Schritt wurden alle Einträge mit der Übereinstimmung in der Postleitzahl, der Strasse und dem Ort eruiert. Mit den Übriggebliebenen folgte ein Match über die Strasse und den Ort und zum Schluss über die Strasse und die Postleitzahl.

FzStammnummer	ID_STR	PLZ_ORT_STR_x	PLZ_ORT_STR_y
557079324	238757	8824 schönenberg zh sonnenrain	8824 schönenberg zh sonnenrain
161031409	143294	8005 zürich förrlibuckstrasse	8005 zürich förrlibuckstrasse
925731559	73993	8911 rifferswil sidlerweid	8911 rifferswil sidlerweid
217285707	83533	8152 glattbrugg stelzenstrasse	8152 glattpark opfikon stelzenstrasse
419161706	66974	8353 elgg bahnhofstrasse	8353 elgg bahnhofstrasse
213679673	254387	8152 opfikon wright strasse	8152 glattpark opfikon wright strasse
222542534	64330	8022 zürich dreikönigstrasse	8002 zürich dreikönigstrasse
925827060	230641	8196 wil zh dorfstrasse	8196 wil zh dorfstrasse
671061755	233116	8403 winterthur pionierstrasse	8400 winterthur pionierstrasse
323637386	228602	8008 zürich drahtzugstrasse	8008 zürich drahtzugstrasse

Abb. 13: Der Datensatz OhneHNRGes_in mit den Methoden des Blocker 1.

Die rot eingefärbten Einträge in der Abb. 13 wurden durch den Match zwischen Postleitzahl und Strasse erzeugt und die Anderen durch den exakten Match zwischen Postleitzahl, Ort und Strasse. Der Datensatz OhnreHNRGes_in enthielt ca. 10'100 Einträge. Insgesamt konnten mit dieser Methode die nicht gefundenen Adressen (OhneHNRGes_out) auf ca. 6'200 reduziert werden. Dies bedeutet eine Reduktion um ca. 62.0%. Dieser Datensatz bildete die Grundlage für den Blocker 2.

FzStammnummer	PLZ_ORT_STR_x	FhHausNrGes1	PLZ_ORT_STR_y
218033945	8152 glattbrugg thurgauerstrasse	11	8152 glattpark opfikon thurgauerstrasse

Abb. 14: Der Datensatz OhneHNRGes_in mit einem falschen Paar.

Der Nachteil des Blocker 1 ist, dass Adressen falsch bereinigt werden können. Dies kann in der Abb. 14 beobachtet werden. Die Thurgauerstrasse 11 befindet sich nicht in 8152 Glattbrugg, sondern in 8050 Zürich. Die Daten wurden im Testdatensatz fälschlicherweise mit der falschen Postleitzahl und Ort angegeben.

3.2.2 Blocker 2: Regelbasiert

Um die restlichen Adressen im Testdatensatz einer Adresse im Referenzdatensatz zuordnen zu können, wurde ein regelbasierter Blocker definiert. Dieser konnte aus dem Package py_entitymatching [5] verwendet werden. Als Ähnlichkeitsmass wurde der Otsuka-Ochiai coefficient [6] verwendet, welcher das gleiche Konzept wie die Cosine Similarity verwendet. Als Variablen dienten die PLZ_ORT_STR und als Grenze 33%. Diese wurde bewusst so gewählt, da das neue Attribut aus drei Anderen (Postleitzahl, Ort und Strasse) gebildet wurde. Falls nun in einem Eintrag die Postleitzahl richtig geschrieben wurde jedoch die Strasse und der Ort nicht, erreicht man mindestens eine Übereinstimmung von 33%. Dem entsprechenden Eintrag, aus dem Testdatensatz, werden alle Adressen, aus dem Referenzdatensatz, mit der gleichen Postleitzahl zugeordnet. Somit sollte der gesuchte Match im neuen Datensatz enthalten sein.

$$K = \frac{|A \cap B|}{\sqrt{|A| \cdot |B|}}$$

Formel 1: Berechnung für den Otsuka-Ochiai coefficient.

Der Blocker 2 identifizierte ca. 1’615'300 Paare, welche die verwendete Übereistimmung erfüllten. Ausserdem konnte für jeden Eintrag mindestens ein möglicher Kandidat aus dem Referenzdatensatz zugewiesen werden.

FzStammnummer	ID_STR	PLZ_ORT_STR	PLZ_ORT_STR
659374845	250195	8058 zürich operation center	8087 zürich
200334612	228489	8058 zürich borddienststrasse	8006 zürich kornhausstrasse
218922843	228640	8048 zürich max högger strasse	8008 zürich august forel strasse
309880407	422523	8494 bauma tösstalstrasse	8483 brünggen tösstalstrasse
678570114	228512	8052 zürich buhnrainweg	8006 zürich vogelsangstrasse
212945638	232864	8404 winterthur rudolf diesel strasse	8400 winterthur general guisan strasse
220568957	64297	8058 zürich frachtstrasse	8002 zürich ulmbergstrasse
218065454	230110	8180 bülach wiesentalstrasse	8180 bülach bäretsmoosstrasse
183259120	240620	8910 affoltern am albis knonaueramt	8910 affoltern am albis semperweg
416976703	147152	8413 neftenbach seuzacherstrasse	8413 neftenbach im hüdel

Abb. 15: Der Datensatz Block mit der regelbasierten Methode des Blocker 2.

Alle Paare in der Abb. 15 zeigen eine Übereinstimmung in mindestens einem Attribut (Postleitzahl, Ort oder Strasse).

3.2.3 Kandidatenset

Nach den beiden Blockern folgte die Erstellung eines Kandidatensets. Dieses sollte alle möglichen Paare zwischen dem Test- und Referenzdatensatz enthalten und bildete die Grundlage für den Vergleich (siehe Kap. 3.3) sowie die Klassifizierung (siehe Kap. 3.4). Um das gewünschte Datenset zu erhalten mussten die Einträge aus den Ergebnissen des Blocker 1 (OhneHNRGes_in) und des Blocker 2 (Block) zusammengeführt und mit den Gesamthausnummern ergänzt werden. Dadurch, dass eine Adresse im Referenzdatensatz mehrere Gesamthausnummern enthalten kann, wurde jeweils nur jene benutzt, welche die geringste Abweichung hatte.

FzStammnummer	FhPLZ1	FhOrt1	Fh_STR_HNR	POSTLEITZAHL	ORTBEZ	Ad_STR_HNR	Diff_HNR	Diff_HNRA
412991400	8049	zürich	rebbergstr 64	8049	zürich	rebbergstrasse 64	0	0.0
412991400	8049	zürich	rebbergstr 64	8037	zürich	rebbergstrasse 37	27	0.0
412991400	8049	zürich	rebbergstr 64	8037	zürich	rebbergstrasse 35	29	0.0
412991400	8049	zürich	rebbergstr 64	8037	zürich	rebbergstrasse 33	31	0.0
412991400	8049	zürich	rebbergstr 64	8037	zürich	rebbergstrasse 31	33	0.0

Abb. 16: Datensatz blocking mit FzStammnummer = 412991400.

Aus der neugebildeten Variable Fh_STR_HNR wurde die jeweilige Hausnummer extrahiert und mit jener aus dem Referenzdatensatz HNR, die Differenz Diff_HNR gebildet. Zusätzlich wurde mit der Levenshtein distance [7] der Unterschied zum Zusatz in Diff_HNRA berechnet. Mit diesen beiden Massen wurde die Adresse gewählt, welche den geringsten Unterschied zur ursprünglichen, im Testdatensatz, besass. In dem Beispiel in der Abb. 16 wurde das rot markierte Paar für das Kandidatenset gewählt.

$$\operatorname{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ \min \begin{cases} \operatorname{lev}_{a,b}(i-1,j) + 1 \\ \operatorname{lev}_{a,b}(i,j-1) + 1 \\ \operatorname{lev}_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} \end{cases} & \text{otherwise.} \end{cases}$$

Formel 2: Berechnung für die Levenshtein distance [7].

Durch diese Optimierung konnten die Anzahl Paare von ca. 12’729'000 (blocking) auf ca. 1’625’400 (Kandidatenset) reduziert werden und daher um 87.2%.

Die Adressen, welche noch bereinigt werden müssen, betragen 16'300 und machen gerade einmal 1% im Kandidatenset aus. Wenn wir nun Paare aus dem Kandidatenset ziehen würden, wäre nur gerade eine von hundert Ziehungen ein richtiger Match. Das nächste Kapitel zeigt eine Methode, um diesen Faktor zu verbessern und mehr korrekte Matches zu erhalten.

3.3 Pair Comparison

Um noch weitere Paare entfernen zu können, welche nicht zusammengehören, wurde die Distanz mit dem Soft-TF-IDF [8] aus dem Packag py_stringmatching [9] berechnet. Als Ähnlichkeitsfunktion wurde die Jaro distance mit einem threshold=0.5 gewählt. Dies bedeutet, dass ein Text als gleich gewertet wird, falls die berechnete Ähnlichkeit grösser ist als der threshold. Das entsprechende Gewicht wird dann in die Berechnung des Soft-TF-IDF übernommen. Die Grenze wurde zu Beginn so gewählt, da sie als Default-Wert vom Packag angegeben war. Während des gesamten Prozesses hat sich dies so bewährt und darum wurde sie nicht mehr angepasst. Ausserdem ist die Grenze nicht von zentraler Bedeutung, da der neue Datensatz nur am Anfang zum Ziehen der zu labelnden Paare (siehe Kap. 3.4) verwendet wird.

$$sim_j = \begin{cases} 0 & \text{if } m = 0 \\ \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right) & \text{otherwise} \end{cases}$$

Where:

- $|s_i|$ is the length of the string s_i
- m is the number of *matching characters*
- t is half the number of *transpositions*

Formel 3: Berechnung für die Jaro distance zwischen zwei Strings.

Eine Erklärung zum Soft-TF-IDF würde den Rahmen dieser Arbeit sprengen und deshalb wird auf die Homepage von Wordpress[3] verwiesen. Auf dieser hat es ein schönes Beispiel, mit Verwendung der Jaro distance, um das Ähnlichkeitsmass besser zu verstehen. Der Vorteil dieser Methode ist es, dass Adressen, welche einen Schreibfehler hatten, trotzdem einen hohen Wert mit dem Soft-TF-IDF erhalten. Dies ist zu begründen durch die Jaro distance, welche zusätzlich in die Berechnung einfliesst und eine gewisse Abweichung zulässt. Beim üblichen TF-IDF würden nur exakte Matches berücksichtigt werden und daher ein tieferer Wert resultieren. Die Ähnlichkeit (score1) wurde auf Grundlage der Attribute mit der Postleitzahl, dem Ort und der Strasse (PLZ_ORT_STR) berechnet.

Für die Teilmenge vom Kandidatenset (Kandidatenset_sub) wurden die Paare gemäss FzStammnummer gruppiert und nach score1 sortiert. Anschliessend wurden die ersten 4 Einträge verwendet. Dieser Wert ist wie bereits bei der Grenze der Jaro distance nicht sonderlich wichtig. Als Kriterium wurde diese Anzahl Einträge gewählt, damit

[3] Berechnungsbeispiel zum Soft-TF-IDF mit der Jaro distance: https://sistemanalize.wordpress.com/2017/12/19/big-data-hybrid-similarity-measure-the-soft-tf-idf-measure/

beim Ziehen aus dem Kandidatenset_sub (ca. 33'300 Einträge) ca. jedes zweite Paar ein korrekter Match ist.

FzStammnummer	FhPLZ1	Fh_STR_ORT	FhHausNrGes1	POSTLEITZAHL	Ad_STR_ORT	HNRGes	score1
412991400	8049	rebbergstr zürich	64	8049	rebbergstrasse zürich	64	0.923810
412991400	8049	rebbergstr zürich	64	8037	rebbergstrasse zürich	37	0.923810
412991400	8049	rebbergstr zürich	64	8049	rebbergsteig zürich	7	0.906667
412991400	8049	rebbergstr zürich	64	8037	rebbergsteig zürich	6	0.906667

Abb. 17: Datensatz Kandidatenset_sub mit FzStammnummer = 412991400.

Da die 4 Paare mit der höchsten Ähnlichkeit (score1) gewählt wurden, ist sichergestellt, dass jede zu bereinigende Adresse, aus dem Testdatensatz vorhanden ist. Das heisst, jede Adresse hat mindestens 1 und höchstens 4 vorgeschlagene Adressen aus dem Referenzdatensatz. In der Abb. 17 sieht man, dass der rot markiert Eintrag der korrekte Match ist.

3.4 Klassifizierung

Ganz am Ende des End-to-End-Prozess befindet sich die Klassifizierung, welche durch überwachtes Lernen, die erzeugten Paare in Match und Kein-Match einteilen sollte. Damit dies umgesetzt werden konnte, musste zuerst ein Datensatz für das Training und Validieren des Klassifizierers erstellt werden. Ausserdem wurden verschiedene Methoden ausprobiert und deren Resultate dargestellt.

3.4.1 Trainings- und Testdaten

Damit die Maschine überhaupt etwas aus den vorbereiteten Daten (Kandidateset_sub) lernen konnte, wurde eine Stichprobe mit 500 Adresspaaren gezogen. Diese mussten anschliessend manuell verglichen und mit einem Label 0 (Kein Match) oder 1 (Match) gekennzeichnet werden. Um die richtigen Labels zu vergeben, musste man im Referenzdatensatz und Internet forschen. Das Labeln aller 500 Paare beanspruchte ca. 5 Stunden und war daher sehr zeitintensiv. Um diesem Problem entgegenzuwirken, könnte das Active Learning [10] (siehe Kap. 3.4.3) verwendet werden. Die gelabelten Daten wurden anschliessend in 400 Trainings- und 100 Testdaten eingeteilt. Nachdem die Features (siehe unten) erstellt wurden, konnten man die Daten verwenden, um den Klassifizierer zu trainieren und validieren.

Damit die Maschine die Daten auswerten konnte, mussten sogenannte Features erzeugt werden. Diese wurden benötigt, um die Adresspaare zu klassifizieren. Durch die Verwendung der automatisch erzeugten Features, aus dem Package py_entitymatching [5], wurde dieser Schritt sehr einfach gehalten und bietet noch Verbesserungspotential. Es wurden ausschliesslich Ähnlichkeitsfunktionen mit unterschiedlichen Tokenizer angewendet. In der Tab. 3 können die Bestandteile der Features betrachtet

werden. Um den End-to-End Prozess in diesem Bereich zu verbessern, wäre es möglich die Methode Word2Vec [11] zu verwenden. Diese erzeugt für jede Adresse einen Vektor, welche diese in einem n-dimensionalen Raum projiziert. Dies ergäbe n-Features, welche wieder vom Klassifizierer benutzt würde.

Attribut	Tokenizer	Ähnlichkeitsfunktion
Strasse	qgm_3	jaccard
Hausnummer	dlm_dc0	cosine
Postleitzahl	None	monge_elkan
Ort		lev_dist
BFS-Nummer		lev_sim
BFS_PLZ_ORT_STR_HNR (Alle einzelnen Attribute in einem Gesamten)		needleman_wunsch
		smith_waterman
		exact_match
		abs_norm

Tab. 3: Darstellung der verschiedenen Bestandteile der automatisch erzeugten Features.

Ein Feature bestand aus je einem Attribut aus dem Test- (Fh) und Referenzdatensatz (Ad) nach dem Normalisieren (siehe Kap. 2.3), dem entsprechendem Tokenizer und einer Ähnlichkeitsfunktion. Daraus ergab sich für jedes Adresspaar 36 numerische Features. Diese und das entsprechende Label konnte anschiessend für das Trainieren und Validieren des Klassifizierers verwendet werden.

3.4.2 Passive Learning

Für den Ansatz mit dem Passive Learning wurden alle 400 Trainings- und 100 Testdaten verwendet. Es wurden 6 verschiedene Klassifizierer (siehe Tab. 4) getestet, welche wiederum vom Package py_entitymatching [5] zu Verfügung steht. Das Problem dieses Ansatzes, sind die vielen Labels, welche benötigt wurden.

Klasssifizierer	Precision		Recall		F1-Score	
	Mean	Var	Mean	Var	Mean	Var
Random Forest	0.9593	0.0009	0.9464	0.0015	0.9521	0.0005
Logistic Regression	0.9559	0.0008	0.9424	0.0011	0.9486	0.0004
Linear Regression	0.9533	0.0009	0.9221	0.0017	0.9368	0.0007
Decision Tree	0.9257	0.0010	0.9432	0.0017	0.9334	0.0005
Support-Vector Machine	0.9551	0.0008	0.9089	0.0021	0.9306	0.0007
Naive Bayes	0.8978	0.0016	0.9207	0.0017	0.9080	0.0007

Tab. 4: Ergebnisse der verschiedenen Klassifizierer mit 50 Durchläufen.

Um die Ergebnisse zu erhalten, wurde für jeden Klassifizierer 50 Durchläufe definiert. Dies beinhaltet für jede Wiederholung die neue Aufteilung der gelabelten Daten in 400 Trainings- und 100 Testdaten. Danach folgte das Training und um die Resultate zu erhalten, die Validierung mit den Testdaten. Mit den Vorhersagen und den Labels konnten die Masse (Precision, Recall, F1-Score) berechnet werden. Dabei produzierte

die Ensemble Methode Random Forest die besten Testresultate und wurde für die weiteren Methoden verwendet. Für die Daten resultieren dabei die folgenden Ergebnisse in der Tab. 5, welche alle wichtigen Messwerte enthalten.

```
Precision : 100.0% (44/44)
Recall : 89.8% (44/49)
F1 : 94.62%
False positives : 0 (out of 44 positive predictions)
False negatives : 5 (out of 56 negative predictions)
```

Tab. 5: Ergebnisse des Random Forest nach dem Training mit 400 Daten.

Es resultierten nur 5 falsch klassifizierte Adressen (siehe Abb. 18). Die Variable BFS_PLZ_ORT_STR_HNR enthält die BFS-Gemeindenummer, die Postleitzahl, den Ort, die Strasse und die Gesamthausnummer. Ausserdem stammen die Daten aus dem Test- (Fh) und dem Referenzdatensatz (Ad) nach der Normalisierung.

FzStammnummer	Ad_ID	label	predicted	probability	Fh_BFS_PLZ_ORT_STR_HNR	Ad_BFS_PLZ_ORT_STR_HNR
647780404	5875865	1	0	0.50	181 8492 wila schalchen	181 8492 wila schalchenstrasse
683089254	2689413	1	0	0.00	62 8058 zürich frachtgebäude ost	62 8058 zürich
214715259	2689413	1	0	0.00	62 8058 zürich grenzwachtposten zürich flughafen	62 8058 zürich
681768361	2689413	1	0	0.00	62 8058 zürich flughafen zürich-kloten	62 8058 zürich
649545522	2689413	1	0	0.34	62 8058 zürich fracht ost	62 8058 zürich

Abb. 18: Die 5 falsch klassifizierten Paare des Random Forest.

In der Spalte probability kann die Wahrscheinlichkeit, welche der Klassifizierer berechnet hatte, ob Match oder nicht, betrachtet werden. Dabei ist es doch sehr erstaunlich, dass die drei Records in der Mitte (rot markiert) eine Wahrscheinlichkeit von 0% haben, obwohl sie ein Match wären. Dies bedeutet, dass sich der Random Forest eindeutig für das falsche Label entschieden hat. Ausserdem sieht man, dass 4 von 5 Fällen den Flughafen mit der Postleitzahl 8058 betreffen. Mögliche Erklärung wäre, dass sich in den Trainingsdaten keine Beispiele befanden, um dies zu lernen.

3.4.3 Active Learning

Wie bereits in Kap. 3.4.1 angesprochen ist das Erstellen von gelabelten Daten sehr zeitintensiv. Bei den 500 Datenpunkten wurden insgesamt 5 Stunden benötigt. Das heisst für jedes einzelne Label im Schnitt ca. 36 Sekunden. Darum sollte ein Klassifizierer ermittelt werden, welcher mit möglichst wenigen Trainingsdaten auskommt.

<u>Algorithmus (Zufällige Trainingsdaten)</u>

1. Start mit 10 zufälligen Trainingsdaten
2. Training und Validierung
3. Ziehen von 10 zufälligen Trainingsdaten
4. Zurück zu Punkt 2.

Um ein aussagekräftiges Resultat zu erhalten, wurde jeder Schritt 50-mal wiederholt. Daraus konnte der Durchschnitt und das 95%-Konfidenzintervall berechnet und anschliessend in der Abb. 19 visuell dargestellt werden.

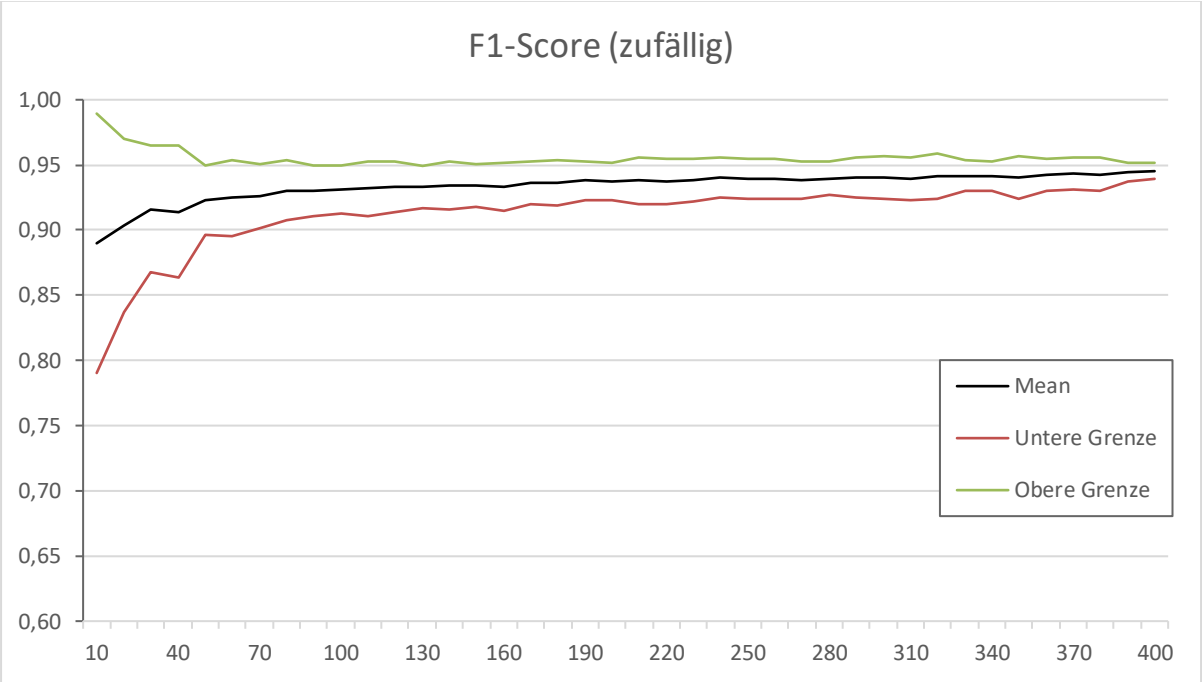

Abb. 19: F1-Score des Random Forest mit 50 Durchläufen und der Anzahl zufälliger Trainingsdaten.

Man kann beobachten, dass bereits mit wenigen Trainingsdaten ein sehr guter F1-Score erreicht wurde. Ausserdem streuen die Ergebnisse weniger, je mehr Daten verfügbar sind. Der Random Forest verbessert sich stetig und näher sich dem Passive Learning (bei 400 Trainingsdaten) an.

Das Ziel war es ein möglichst schneller Anstieg des F1-Scores zu gewährleisten. Dies würde bedeuten, dass weniger gelabelte Daten nötig wären. Um dies ermöglichen zu können, wurde ein Algorithmus getestet, welcher die absolute Differenz zwischen Wahrscheinlichkeit (probability) und dem Label vergleicht. Mit diesem wurde entschieden, welche Beispiele für das Training hinzugefügt werden sollen. Als Klassifizierer wurde wieder der Random Forest gewählt.

<u>Algorithmus (Ausgewählte Trainingsdaten)</u>

1. Start mit 10 zufälligen Trainingsdaten
2. Training und Validierung
3. Eruieren der absoluten Differenzen
4. Ziehen von 10 Trainingsdaten mit den grössten Differenzen
5. Zurück zu Punkt 2.

Wie bereits beim Algorithmus mit den zufälligen Trainingsdaten, erfolgten 50 Wiederholungen. Der Mittelwert und das 95%-Konfidenzintervall wurden in der Abb. 20 dargestellt.

Abb. 20: F1-Score des Random Forest mit 50 Durchläufen und der Anzahl ausgewählter Trainingsdaten.

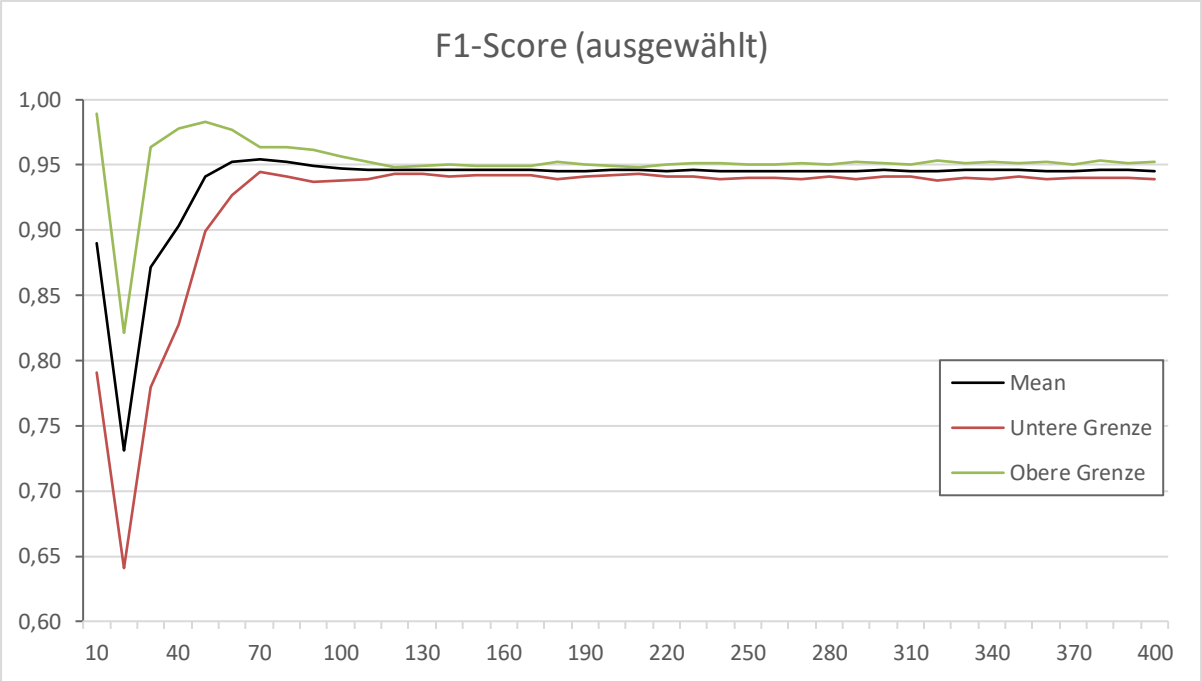

Abb. 20: F1-Score des Random Forest mit 50 Durchläufen und der Anzahl ausgewählter Trainingsdaten.

In den Ergebnissen kann erstaunlicherweise eine Verschlechterung beobachtet werden. Dies geschah nach der ersten Ziehung von 10 weiteren Trainingsdaten. Jedes Mal entschied sich der Algorithmus für solche mit denen er im nächsten Schritt einen schlechteren F1-Score erzielte. Eine Erklärung könnte eine Verzerrung der Trainingsdaten sein. Der Algorithmus analysiert, nach dem Training, die gesamten Trainingsdaten und aufgrund dieser analysiert er die höchste Abweichung zwischen Label und Wahrscheinlichkeit. Ab ca. 120 Trainingsdaten ist das 95%-Konfidenzintervall sehr schmal. In diesem Beispiel kann der Algorithmus den F1-Score nicht mehr verbessern.

Um die beiden Methoden (zufällig/ ausgewählte Trainingsdaten) besser interpretieren zu können, wurden sie einander in der Tab. 6 und Abb. 21 gegenübergestellt.

Labels	Zufällig	Ausgewählt	Labels	Zufällig	Ausgewählt
10	0,8899	0,8899	60	0,9246	0,9522
20	0,9033	0,7310	70	0,9263	0,9544
30	0,9162	0,8719	80	0,9305	0,9524
40	0,9142	0,9030	90	0,9301	0,9497
50	0,9231	0,9412	100	0,9315	0,9477

Tab. 6: Gegenüberstellung der Algorithmen zufällige/ ausgewählte Trainingsdaten nach Anzahl Labels.

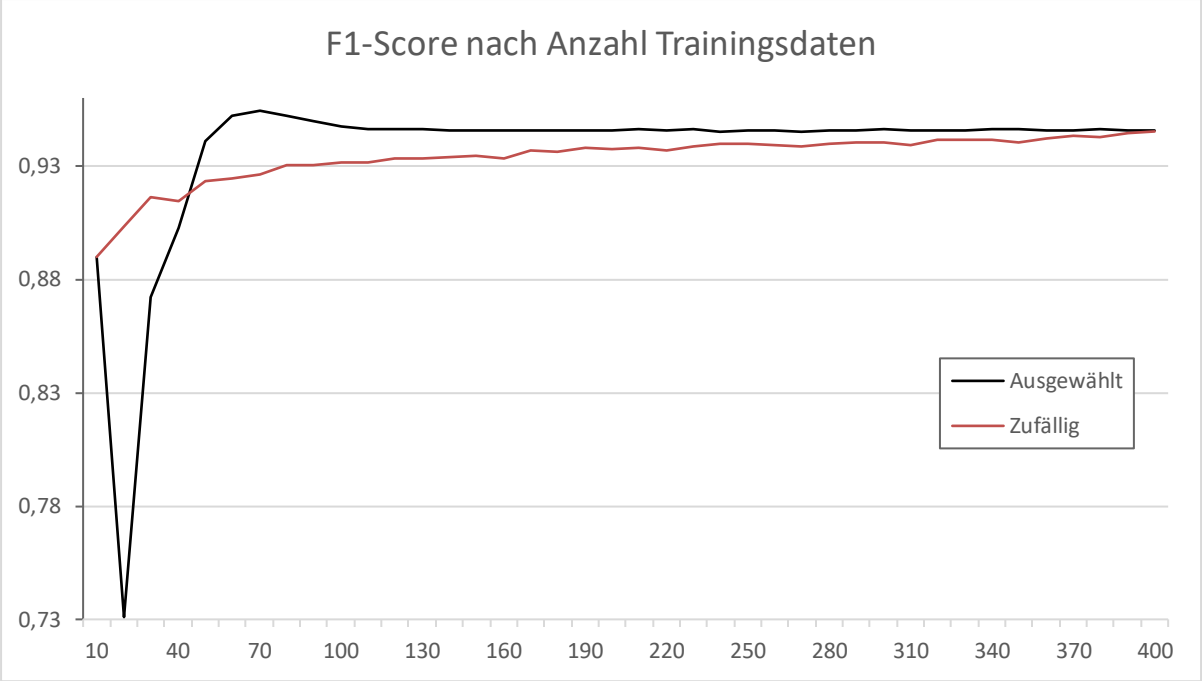

Abb. 21: Vergleich des Algorithmus mit zufälligen und ausgewählten Trainingsdaten.

Beim Vergleich der beiden Algorithmen kann festgestellt werden, dass jener welcher die Trainingsdaten spezifisch ausgewählt hat, eine bessere Lernkurve (ausgenommen Knick nach unten) besitzt und auch eine kleinere Streuung (siehe Abb. 20). Zu Beginn bis ca. 40 Trainingsdaten besass der Algorithmus mit den zufälligen Stichproben einen besseren F1-Score. Ab 50 Trainingsdaten konnte sich der mit den ausgewählten Daten behaupten. Spannend ist, dass er sich bis zu 50 Labels verbessern konnte, sich danach aber wieder verschlechterte. Mit allen Daten erreichen die beiden Algorithmen wieder das gleiche Resultat.

Um sich einen Grossteil der verschwendeten Zeit während dem Labeln zu ersparen, eignet sich der Algorithmus, welcher sich die spezifischen Trainingsdaten aussucht. Eine Umsetzung wäre sehr einfach. Nach jedem Training müsste die Maschine den User fragen, welche Labels für seine vorgeschlagenen Adresspaare richtig wären. Anschliessend würden diese zusammen mit den vorhergehenden in den Trainingsprozess einfliessen. Bei diesem Beispiel hätten 70 gelabelte Daten gereicht, um das beste Ergebnis zu erzielen. Wenn nun mit 36 Sekunden pro Label gerechnet wird, hätten dies einen Zeitaufwand von gerade einmal 42 Minuten betragen.

4 Schlusswort

Es sollte ein End-to-End Prozess aufgebaut werden, welcher für einen spezifischen Datensatz eine Adressbereinigung erstellt. Ausserdem sollten die Resultate in den einzelnen Schritten dargestellt und Methoden getestet werden, damit die Anzahl benötigter Labels minimiert werden könnten.

Vorgängig musste ein Referenzdatensatz erstellt werden, um alle Adressen der Schweiz zu erhalten. Dazu wurden die Daten vom OGD-Portal der schweizerischen Post geladen. Anschliessend folgte das sogenannte Blocking und Pair Comparison mit welchem die Berechnungszeit massiv verkürzt wurden. Zum Abschluss wurde, durch das Labeln einer Stichprobe, ein Trainings- und Testdatensatz erstellt. Mit diesen wurden verschiedene Klassifizierer und Methoden getestet. Dabei beanspruchte die Vorbereitung der Test- und Referenzdaten (siehe Kap. 2) mit ca. 70% den Hauptteil der Zeit. Demzufolge wurden für das Matching (siehe Kap. 3) noch 30% aufgewendet.

Die Arbeit zeigt auf, dass durch Anwenden eines aktiven Lernens (Active Learning), mit der Methode von ausgewählten Trainingsdaten, die Zeit für das Labeln von Daten stark verkürzt werden kann. Ausserdem wurde ersichtlich, dass eine saubere Datengrundlage die Ergebnisse verbessern kann und daher ein Hauptaugenmerk daraufgelegt werden sollte.

Für die Bildung von Features, welche essenziell für den Klassifizierer sind, wurde keine Optimierung behandelt. Diese wurden ausschliesslich über Ähnlichkeitsmasse definiert. Eine mögliche Erweiterung wäre das Anwenden von Word2Vec. Ausserdem wäre es spannend einen Prozess aufzubauen, welcher möglichst wenige manuelle Schritte benötigt. Das heisst das die Features automatisch auf Grundlage der Labels gebildet werden.

In Zukunft wird das Automatisieren eines End-to-End Prozesses eine immer wichtigere Rolle spielen. Während Maschinen immer mehr Aufgaben übernehmen und selbstständig ausführen können, werden sich menschliche Arbeitskräfte auf das Kontrollieren der Maschinen beschränken. Dies wäre das schrittweise Labeln der durch den Prozess (Active Learning) vorgeschlagenen Beispiele.

5 Verzeichnisse

5.1 Literaturverzeichnis

[1] Schweizer Post. Open Data Portal. (10.04.2020) https://swisspost.opendatasoft.com/pages/home/

[2] Schweizer Post. Beschreibung Strassenverzeichnis. (10.04.2020) https://swisspost.opendatasoft.com/api/datasets/1.0/strassenbezeichnungen_v2/attachments/strassenverzeichnis_mit_sortierdaten_de_pdf/

[3] Y. Govind et al. Entity Matching Meets Data Science:A Progress Report from the Magellan Project. SIGMOD, 2019

[4] G. Navarro. A Guided Tour to Approximate String Matching, 2001

[5] Anhaidgroup. Package py_entitymatching 0.3.0 documentation (29.04.2020) http://anhaidgroup.github.io/py_entitymatching/v0.3.x/index.html

[6] M. Hardy et al. Cosine similarity http://icybcluster.org.ua:34145/technology-documents/cosine_similarity_eng.pdf

[7] N. Babar. The Levenshtein Distance Algorithm (29.04.2020) https://dzone.com/articles/the-levenshtein-algorithm-1

[8] William W. Cohen. A Comparison of String Distance Metrics for Name-Matching Tasks https://www.cs.cmu.edu/~wcohen/postscript/ijcai-ws-2003.pdf

[9] Anhaidgroup. Package py_stringmatching 0.2 (29.04.2020) http://anhaidgroup.github.io/py_stringmatching/v0.4.x/index.html

[10] A. Arasu, M Götz and R. Kaushik. On active learning of record matching packages. SIGMOD, 2010 https://dl.acm.org/doi/10.1145/1807167.1807252

[11] T. Mikolov et al. Distributed Representations of Words and Phrasesand their Compositionality. NIPS 2013 https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf

5.2 Abbildungsverzeichnis

5.3 Tabellenverzeichnis

5.4 Formelverzeichnis