Leibniz-Institut für
Wirtschaftsforschung
Halle

IWH TECHNICAL REPORTS

RLPC:
Record Linkage Pre-Cleaning

Technische Dokumentation der Routinen

Wilfried Ehrenfeld

02|2015

Autor:
Dr. Wilfried Ehrenfeld

Kontakt:
Dr. Cornelia Lang
Leiterin des IWH-Datenzentrums
Telefon: + 49 345 77 53 802
Fax: + 49 345 77 53 820
E-Mail: cornelia.lang@iwh-halle.de

Herausgeber: LEIBNIZ-INSTITUT FÜR WIRTSCHAFTSFORSCHUNG HALLE – IWH
Geschäftsführender Vorstand: Prof. Reint E. Gropp, Ph.D.
Prof. Dr. Oliver Holtemöller
Dr. Tankred Schuhmann

Hausanschrift: Kleine Märkerstraße 8, D-06108 Halle (Saale)
Postanschrift: Postfach 11 03 61, D-06017 Halle (Saale)
Telefon: +49 345 7753 60
Telefax: +49 345 7753 820
Internetadresse: www.iwh-halle.de

Zitierhinweis:
Ehrenfeld, Wilfried: RLPC: Record Linkage Pre-Cleaning – Technische Dokumentation der Routinen. IWH Technical Reports 02/2015. Halle (Saale) 2015.

ISSN 2365-9076

RLPC: Record Linkage Pre-Cleaning

Technische Dokumentation der Routinen

Zusammenfassung

Übergeordnetes Ziel des Record-Linkage ist die Zusammenführung verschiedener Datensätze anhand eines eindeutigen Identifizierungsmerkmals. In den uns vorliegenden Fällen handelt es sich primär um Unternehmensdatensätze aus Datenbanken mit Unternehmensmerkmalen (z. B. BvD Amadeus/Dafne), Patentdatensätze (z. B. Patstat oder DPMA) sowie Förderdatensätze (z. B. BMBF Förderkatalog). Diese Datensätze sollen über den Namen der Unternehmen verknüpft werden. Da in der Praxis Unternehmensnamen in verschiedenen Datenbasen uneinheitlich geschrieben werden – beispielsweise die Unternehmensform – ist eine Harmonisierung und Standardisierung notwendig.

Die hier beschriebenen Routinen vollziehen das Record-Linkage-Pre-Cleaning (RLPC). Sie dienen der Erzeugung eines Record-Linkage kompatiblen Namens (RLName) aus einem vorgegebenen (Akteurs-)Namen (Name). Dabei werden Sonderzeichen auf ASCII-Zeichen zurückgeführt, Unternehmensformen identifiziert, Klammerausdrücke isoliert bzw. abgespalten und schließlich so ein Ausdruck geschaffen, der einen Vergleich mit anderen Namen ermöglicht. Im Anschluss an dieses Pre-Cleaning kann mittels Record-Linkage Systemen die Zusammenführung mehrerer so vorbehandelter Datensätze erfolgen.

Inhaltsverzeichnis

Abbildungsverzeichnis

1. Ausgangsbasis und Umriss des Verfahrens

Übergeordnetes Ziel des Record-Linkage ist die Zusammenführung verschiedener Datensätze anhand eines eindeutigen Identifizierungsmerkmals. In den uns vorliegenden Fällen handelt es sich primär um Unternehmensdatensätze aus Datenbanken mit Unternehmensmerkmalen (z. B. BvD Amadeus/Dafne) bzw. Hochschulen und außeruniversitäre Forschungseinrichtungen (z. B. Research Explorer - siehe Ehrenfeld 2015b) sowie Patentdatensätze (z. B. Patstat/Regpat oder DPMA), Publikationsdaten (z. B. Weg of Knowledge) und Förderdatensätze (z. B. BMBF Förderkatalog). Diese Datensätze sollen über den Namen der Unternehmen verknüpft werden. Da in der Praxis Unternehmensnamen in verschiedenen Datenbasen uneinheitlich geschrieben werden - beispielsweise die Unternehmensform - ist eine Harmonisierung und Standardisierung notwendig.

Aus einer vorgegebenen Variable für den (Akteurs-)Namen `Name` wird mittels des hier beschriebenen Verfahrens der bereinigte, Record-Linkage kompatible Name `RLName` erzeugt. Das Verfahren wird deshalb als Record-Linkage-Pre-Cleaning (kurz: RLPC) bezeichnet. Dazu werden im Groben die im Folgenden beschriebenen Schritte durchlaufen. Zu Beginn des Verfahrens werden einige grundlegende Variablen und Routinen definiert. Diese sind für den weiteren Verlauf des Verfahrens notwendig. Das eigentliche Verfahren gliedert sich in drei Stufen.

Zuerst erfolgt eine Zeichenbereinigung. Dazu wird der Akteursname komplett in Großbuchstaben umgewandelt. Im Zuge dessen werden auch deutsche Umlaute, Zeichen mit Akzenten oder Zeichen mit Codierungsvermerken durch ihre ASCII-Äquivalente ersetzt. Doppelte Leerzeichen im Namen sowie Leerzeichen am Beginn oder Ende des Namens werden entfernt. Anschließend werden Klammersymbole und Schreibweisen für „und“ vereinheitlicht sowie vorhandene Klammerausdrücke extrahiert.

In einem zweiten Schritt werden alle Nicht-ASCII-Zeichen aus dem Namen entfernt. Anschließend werden die Gesellschaftsformen von Unternehmen identifiziert. Dies geschieht mittels einer Identifikationstabelle, die zum aktuellen Zeitpunkt über 600 Schreibweisen für verschiedene Gesellschaftsformen umfasst. Die Originalschreibweisen der Gesellschaftsform werden anschließend aus dem Unternehmensnamen entfernt. Schließlich werden einige Schreibweisen von häufig verwendeten, aber unterschiedlich geschriebenen Begriffen harmonisiert. Zuletzt werden alle Leerzeichen aus dem Ausdruck entfernt.

Die hier vorgestellten Routinen basieren auf den von Magerman, Van Looy und Song (2006) vorgestellten Methoden zur Harmonisierung von Patentdaten. Diese wurden für die uns vorliegenden Datensätze jedoch umfangreichen Modifikationen und Erweiterungen unterzogen und in ein modulares System von Stata-Routinen übersetzt. Im Anschluss an dieses Pre-Cleaning kann die Zusammenführung mehrerer so vorbehandelter Datensätze erfolgen. Diese Prozeduren eignen sich sehr gut für eine spätere Zuordnung bei *leicht* abweichenden Schreibweisen für denselben Akteursnamen.

Von der Situation leicht abweichender Schreibweisen zu unterscheiden sind im technischen Sinne deutlich andere Bezeichnungen für dieselbe Institution. Als Beispiel hierfür seien die Technischen Universitäten genannt (mit ihre oft verwendeten Kurzform „TU“) sowie (als „Klassiker“) die Rheinisch-Westfälische Technische Hochschule Aachen (kurz: RWTH Aachen). Mit einer rein deterministischen Zuordnung der Originaldatensätze oder „unscharfen“ (probabilistischen) Methoden alleine ist es in diesen Fällen kaum möglich, eine sichere Zuordnung zu gewährleisten. Diese Fälle können zur Standardisierung mittels automatisierter Ersetzungsregeln bzw. mit einer zusätzlichen Tabelle für unterschiedliche Schreibweisen derselben Institution erfasst werden.

Als Record-Linkage Systeme haben in der Vergangenheit bereits die „Merge-Toolbox“ (Schnell, Bachteler und Reiher 2005 bzw. Schnell, Bachteler und Bender 2004) sowie das kommerzielle Programm „Fuzzy Dupes“ Verwendung gefunden. Eine konkrete Anwendung im Rahmen des Projektes „RegDemo“ ist in Titze et al. (2015) sowie Ehrenfeld (2015a) beschrieben.

Abbildung 1 zeigt den strukturellen Ablauf des Pre-Cleanings. Im Folgenden werden die einzelnen Stufen des Verfahrens genauer beschrieben.

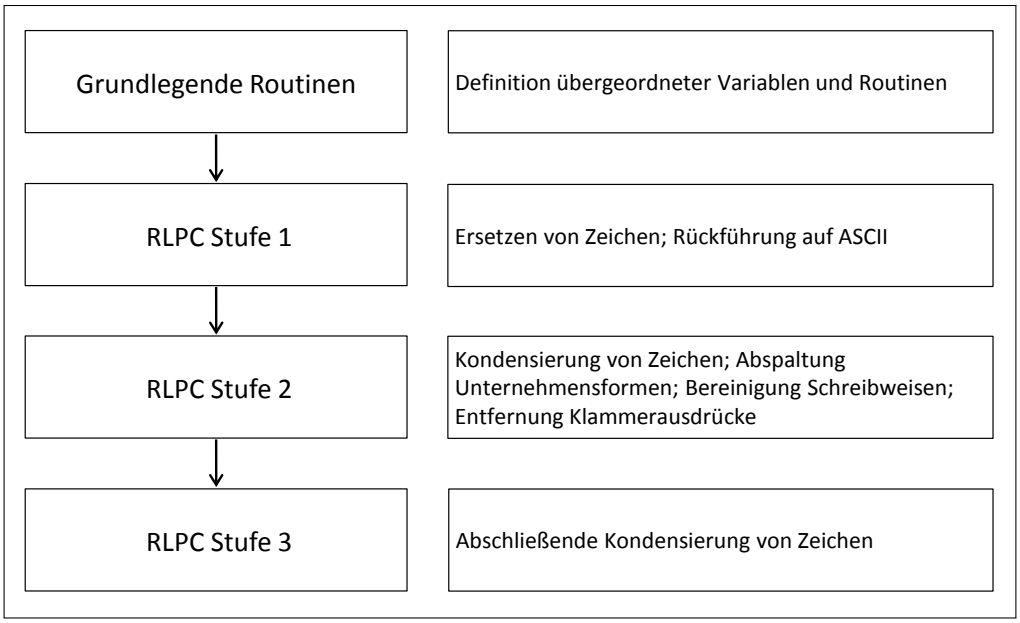

Abbildung 1: Ablaufplan: Struktureller Ablauf des Verfahrens in Stufen.

2. Grundlegende Routinen

In dieser Stufe werden grundlegende Variablen definiert und Routinen geladen. Die Routinen werden dem Verfahren so als Programme zur Verfügung gestellt. Abbildung 2 zeigt den Ablauf dieser grundlegenden Routinen.

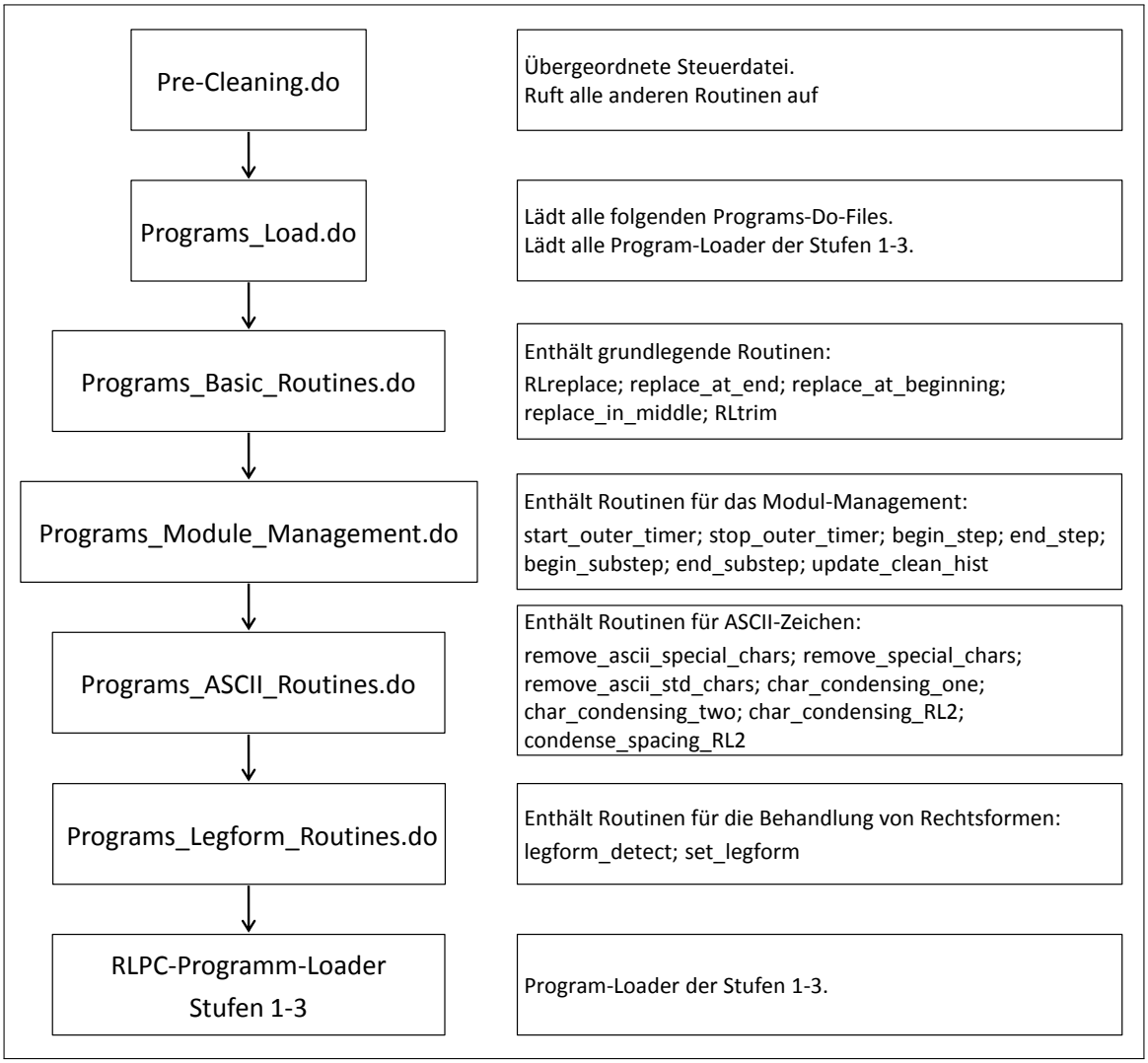

Abbildung 2: Ablaufplan: Laden der grundlegenden Routinen.

2.1. PreCleaning.do

Dies ist die übergeordnete Steuerdatei. Von dieser werden alle anderen Routinen aufgerufen. Hier werden die Pfade zu den Routinen und Datensätzen definiert, die übergeordnete Zeitmessung gesteuert und die für das RLPC benötigten zusätzlichen Variablen definiert.

Das Verfahren benutzt folgende globale Pfade und Dateinamen:

- `$workdir` ist das Arbeitsverzeichnis. Dort wird das generelle logfile `$logname` angelegt.
- `$progdir` ist der Pfad zu den Programmen. Diese liegen als do-files vor.
- `$datadir` ist der Pfad zu den Datensätzen `$load_dataset` und `$save_dataset`.
- `$load_dataset` ist der Dateiname des zu bearbeitenden Datensatzes.
- `$save_dataset` ist der Dateiname des bearbeiteten Datensatzes.

- `$logname` ist der Dateiname des logfiles.

Die neu angelegten Variablen sind im Einzelnen:

- `RLName` wird aus der Variable `Name` generiert ist später der bereinigte, auf ASCII-Zeichen zurückgeführte, und ggf. kondensierte Name. Diese Variable stellt das Hauptergebnis des Verfahrens dar.
- `temp_name` ist der `RLName` aus der letzten Stufe. Er dient später zum Vergleich mit dem `RLName` aus der aktuellen Stufe. So können Änderungen identifiziert werden (Notwendig für `clean_hist`).
- `clean_hist` gibt die wirklich angewendeten Schritte in den einzelnen Einträgen von `RLName` an.
- `legal_form` ist die in Schritt 2.2 abgespaltene und identifizierte Rechtsform eines Unternehmens (siehe Abschnitt 4.2 und 4.3).
- `brackets` nimmt einen in Schritt 1.5 eventuell abgespaltenen Klammerausdruck auf (siehe Abschnitt 3.5).

2.2. Programs_Load.do

`Programs_Load.do` lädt alle folgenden do-files und stellt die darin enthaltenen Programme zur Verfügung.

2.3. Programs_Basic_Routines.do

Die Datei `Programs_Basic_Routines.do` enthält einige grundlegende Programme zur Bearbeitung des Strings `RLName`. Vor allem sind dies Befehle zum Ersetzen von Zeichenketten.

RLreplace

RLreplace ersetzt Zeichenkette <Such-String> durch Zeichenkette <Ersatz-String> in `RLName` - unabhängig von der Position von <Such-String> in `RLName` (normales Ersetzen).

Verwendung: `RLreplace "<Such-String>" "<Ersatz-String>"`

replace_at_end

Dieser Befehl ersetzt Zeichenkette <Such-String> durch Zeichenkette <Ersatz-String> in `RLName`, wenn <Such-String> am *Ende* von `RLName` steht.

Verwendung: `replace_at_end "<Such-String>" "<Ersatz-String>"`

replace_at_beginning

Dieser Befehl ersetzt Zeichenkette <Such-String> durch Zeichenkette <Ersatz-String> in RLName, wenn <Such-String> am *Anfang* von RLName steht.

Verwendung: `replace_at_beginning "<Such-String>" "<Ersatz-String>"`

replace_in_middle

Dieser Befehl ersetzt Zeichenkette <Such-String> durch Zeichenkette <Ersatz-String> in RLName, wenn <Such-String> in der *Mitte* von RLName steht (vgl. RLreplace).

Verwendung: `replace_in_middle "<Such-String>" "<Ersatz-String>"`

RLtrim

RLtrim löscht alle Leereichen am Anfang und Ende von RLName sowie alle doppelten Leerzeichen in RLName.

Verwendung: `RLtrim`

2.4. Programs_Module_Management.do

start_outer_timer

Startet die Zeitnahme durch den äußeren/übergeordneten Timer (`timer 1`).
Öffnet das Timelog zum Schreiben in $workdir/$timelog.

`$timelog = "$logname" + "_TimeStats.log"`

Verwendung: `start_outer_timer`

stop_outer_timer

Stoppt den äußeren Timer (`timer 1`) und schließt das Timelog $timelog.

Verwendung: `stop_outer_timer`

begin_step

Steht zu Beginn einer neuen <Stufe> des Verfahrens.
Startet den inneren Timer (`timer 2`) und ersetzt den temporären Namen der letzten Stufe (temp_name) durch den aktuellen RLName.

Verwendung: `begin_step "<Stufe>"`

end_step

Steht am Ende einer <Stufe> des Verfahrens.
Löscht überflüssige Leerzeichen in `RLName` (mittels RLtrim), stoppt den inneren Timer (`timer 2`), schreibt die innere Zeitnahme in das Timelog und aktualisiert die Einträge der tatsächlich angewendeten Schritte (update_clean_hist).

Verwendung: `end_step "<Stufe>"`

begin_substep

Steht zu Beginn einer neuen <Unter-Stufe> des Verfahrens.
Startet den Timer auf Ebene 3 (`timer 3`).

Verwendung: `begin_substep "<Unter-Stufe>"`

end_substep

Steht am Ende einer <Unter-Stufe> des Verfahrens.
Stoppt den Timer auf Ebene 3 (`timer 3`) und schreibt die Zeitnahme in das Timelog.

Verwendung: `end_substep "<Unter-Stufe>"`

update_clean_hist

Schreibt die innere Zeitnahme (`timer 2`) in das Timelog und aktualisiert die Einträge der tatsächlich angewendeten Schritte in `clean_hist`. Es stehen hier also nur die Nummern der Schritte, die in `RLName` tatsächlich eine Veränderung bewirkt haben. Die Identifikation erfolgt durch Vergleich von `temp_name` und `RLName`.

Verwendung: `update_clean_hist`

2.5. Programs_ASCII_Routines.do

Diese Routinen eliminieren jeweils unterschiedliche Sätze von ASCII-Zeichen in `RLName`. Sie werden in Schritt 2.1 bzw. 3.1 der RLPC-Routinen für das String-Condensing verwendet (siehe Abschnitt 4.1 bzw. Abschnitt 5.1).

remove_ascii_special_chars

Dieser Befehl ersetzt einige „spezielle“ ASCII-Zeichen in RLName durch Leerzeichen. Dies sind die Folgenden:

Nummer	33-39	42	44-47	58-59	61	63-64	92	94-96	124	126
Zeichen	! " # $ % & ´	*	, - . /	: ;	=	? @	\	^ _ ‘	|	~

Verwendung: `remove_ascii_special_chars`

remove_special_chars

Dieser Befehl ersetzt die folgenden spezielle Zeichen in RLName durch Leerzeichen:

$$^2 \cdot {}^\circ \mu \acute{} \S$$

Verwendung: `remove_special_chars`

remove_ascii_std_chars

Dieser Befehl ersetzt alle „normalen“ ASCII-Zeichen in RLName durch Leerzeichen. Er wird nur zu Testzwecken verwendet um die Differenz aller in RLName vorhandenen Zeichen zu den „speziellen“ Zeichen zu bilden.

Diese Zeichen sind:

Nummer	32	40-41	48-57	65-90	97-122
Zeichen	<Space>	( )	0-9	A-Z	a-z

Verwendung: `remove_ascii_std_chars`

char_condensing_one

Diese Routine ersetzt alle Zeichen in RLName, die nicht A-Z, 0-9, (,) oder <Space> sind durch Leerzeichen. Hierzu wird ein geeigneter regulärer Ausdruck verwendet.

Verwendung: `char_condensing_one`

char_condensing_two

Diese Routine *löscht* alle Zeichen in RLName, die nicht A-Z oder 0-9 sind. Hierzu wird ein geeigneter regulärer Ausdruck verwendet.

Im Gegensatz zu char_condensing_one *löscht* die Routine die betroffenen Zeichen.
Weiter wird hier auch das Leerzeichen (<Space>) gelöscht.

Verwendung: `char_condensing_two`

char_condensing_RL2

Diese Routine entspricht char_condensing_two für (den experimentellen) RLName2.

Verwendung: `char_condensing_RL2`

condense_spacing_RL2

Diese Routine kondensiert (löscht Leerzeichen) gesperrte Namen in RLName2. Die gesperrten Namen werden anhand ihres Aufbaus der Form „Mindestens drei einzelne Buchstaben mit Leerzeichen dazwischen“ identifiziert. Hierzu wird ein geeigneter (nichttrivialer) regulärer Ausdruck verwendet.
Anschließend löscht die Routine alle überflüssigen (doppelten) Leerzeichen in RLName2.

Verwendung: `condense_spacing_RL2`

2.6. Programs_Legform_Routines.do

Diese Routinen werden in Schritt 2.2 der RLPC-Routinen zur Identifizierung und Abspaltung von Rechtsformen benötigt (siehe Abschnitt 4.2 und 4.3).

legform_detect

Diese Routine erkennt Rechtsformen in RLName und ersetzt diese in RLName durch ihre Kurzform. Die Kurzform wird dabei durch Einfügen von « und » (z. B. «KG») für folgende Schritte erkennbar gemacht.

Dabei berücksichtigt die Routine besonders kurze Langformen (kürzer als 5 Zeichen) am Anfang und am Ende von RLName. Diese müssen durch ein zusätzliches Leerzeichen in RLName abgetrennt vorhanden sein. Bei Erkennung in der Mitte muss die Langform grundsätzlich duch ein Leerzeichn vorne und hinten abgetrennt vorliegen. Diese Maßnahmen dienen der Reduzierung von falsch (positiv) erkannten Gesellschaftsformen.

Die Liste der Gesellschafts-Langformen sowie ihrer Kurzformen ist:

step_2_2_legal_form_replacement.tsv.

Diese Datei muss im Programmverzeichnis $progdir liegen und sollte der Länge der Langform nach absteigend (!) sortiert sein. So können Spezialformen besser erkannt werden (z. B. GGMBH vs. GMBH)

Die Liste umfasst aktuell mehr als 620 Einträge. Sie enthält zwei Spalten, die durch das <TAB>-Zeichen getrennt sind (Dateikennung tsv). Sie hat das folgende Format:

<Suchtext/Langform><TAB><Kurzform><CR LF> .

Zur Suche der Rechtsform kann der Routine angegeben werden, ob diese am Anfang, am Ende oder in der Mitte von RLName suchen soll. Als Parameter für <replacetype> können deshalb "replace_at_end", "replace_at_beginning" oder "replace_in_middle" angegeben werden.

Verwendung: `legform_detect "<replacetype>"`

set_legform

Dieser Befehl trägt die Rechtsform <set_string> in die (leere) Variable legal_form ein, wenn <search_string> in RLName enthalten ist. Es wird also der erste gefundene Eintrag beibehalten. Der <search_string> ist hier typischerweise die durch « und » gekennzeichnete Kurzform der Rechtsform (z. B. «KG») aus Schritt 2.2 bzw. der Routine legform_detect. Der Ausdruck <set_string> ist dann typischerweise die „normale“ Kurzform.

Beispiel: `set_legform "«GMBH»" "GMBH".`

Verwendung: `set_legform "<search_string>" "<set_string>"`

3. Routinen der RLPC-Schritte - Stufe 1

Hier beginnt das eigentliche Verfahren. Die erste Stufe des RLPC-Verfahrens ersetzt Symbole in RLName oder führt diese auf Standard-ASCII-Zeichen zurück. Abbildung 3 zeigt den Ablauf dieser ersten Stufe.

Jedes Programm des RLPC-Verfahrens in Stufe 1 bis 3 wird durch begin_step und end_step „eingerahmt“. Oftmals wird als letzte Routine in einem Unterschritt RLtrim aufgerufen.

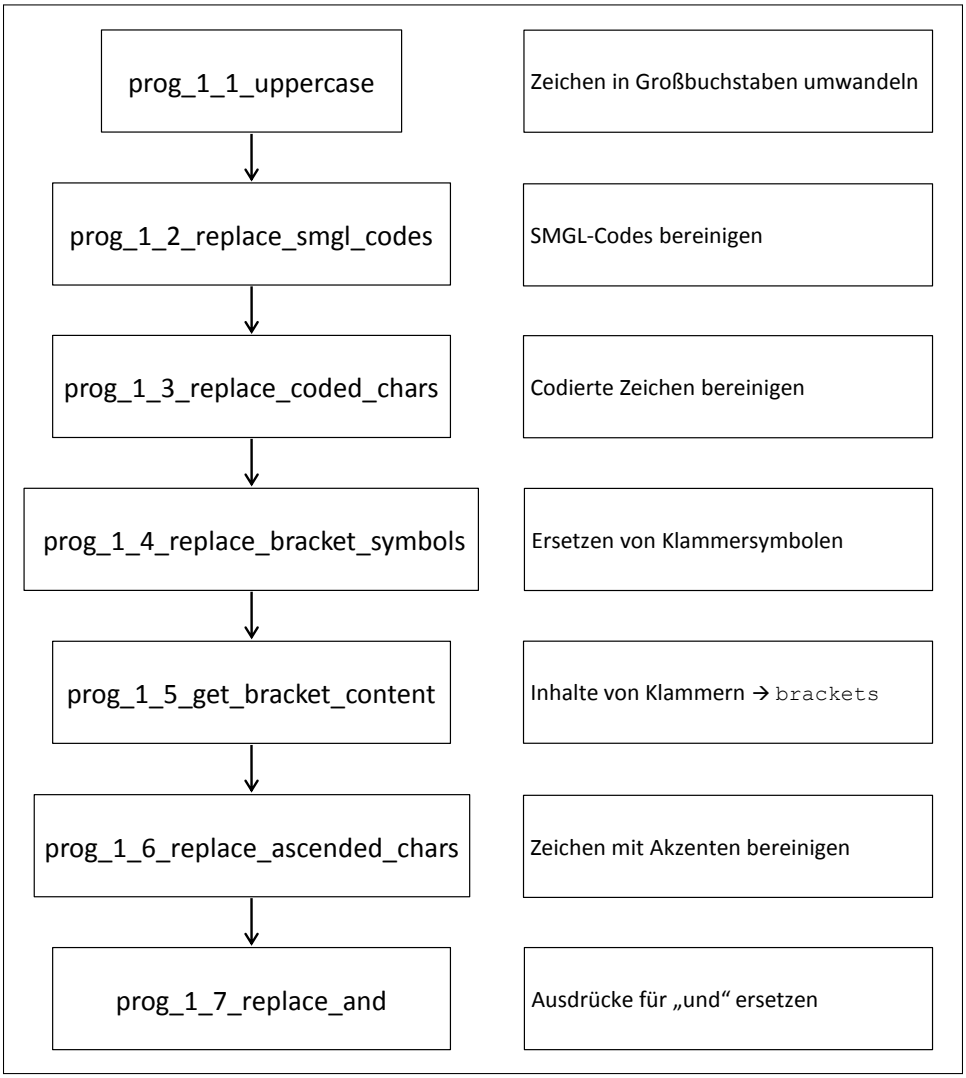

Abbildung 3: Ablaufplan: Funktionen Routinen der Stufe 1.

3.1. prog_1_1_uppercase.do

Diese Routine wandelt alle Buchstaben von `RLName` in Großbuchstaben um. Da Stata bei der `upper`-Funktion einige Zeichen ignoriert, werden diese hier ebenfalls durch ihre großgeschriebenen Versionen ersetzt. Dies betrifft auch die deutschen Umlaute. Im Zuge dessen wird auch „ß“ in „SS“ umgewandelt.

Verwendung: `prog_1_1_uppercase`

Verwendete Routinen: `RLreplace`

3.2. prog_1_2_replace_smgl_codes.do

Hier werden nicht aufgelöste Sonderzeichen der Standard Generalized Markup Language (SGML) übersetzt. Hierzu gehören etwa Zeichen wie "&AMPL;".

Verwendung: `prog_1_2_replace_smgl_codes`

Verwendete Routinen: `RLreplace`

3.3. prog_1_3_replace_coded_chars.do

Dieses Programm ersetzt nicht aufgelöste codierte Sonderzeichen durch ihre vereinfachten ASCII-Varianten. So ersetzt die Routine etwa "{OVERSCORE (A)}" durch "A".

Verwendung: `prog_1_3_replace_coded_chars`

Verwendete Routinen: `RLreplace`

3.4. prog_1_4_replace_bracket_symbols.do

Hier werden die Klammersymbole [, { und < durch (ersetzt. Die Symbole], } und > werden durch) ersetzt. Doppelte spitze Klammern (« und ») werden gelöscht.

Verwendung: `prog_1_4_replace_bracket_symbols`

Verwendete Routinen: `RLreplace`

3.5. prog_1_5_get_bracket_content.do

Die Routine identifiziert Klammerausdrücke in `RLName`. Wird ein Klammerausdruck identifiziert, wird dieser aus `RLName` entnommen und in der Variable `brackets` gespeichert. Steht `RLName` komplett in (runden) Klammern, werden nur die Klammersymbole aus `RLName` entfernt. Die Variable `brackets` enthält dann den Text „Steht komplett in Klammern“.

Der Inhalt in runden Klammern von `RLName` wird später in Schritt 2.6 (Abschnitt 4.7) gelöscht.

Verwendung: `prog_1_5_get_bracket_content`

Verwendete Routinen: `RLtrim`

3.6. prog_1_6_replace_ascended_chars.do

Hier werden akzentuierte Zeichen (accented characters) auf ihre vereinfachten ASCII-Varianten zurückgeführt. Beispiel: „À“ wird durch „A“ ersetzt. Die deutschen Umlaute Ä, Ö, Ü werden hier durch AE, OE, UE ersetzt.

Verwendung: `prog_1_6_replace_ascended_chars`

Verwendete Routinen: `RLreplace`

3.7. prog_1_7_replace_and.do

Diese Routine ersetzt Ausdrücke für „UND“ wie „+“, „AND“, „UND“, „U.“ und „ET“ durch das Et-Zeichen (&).

Verwendung: `prog_1_7_replace_and`

Verwendete Routinen: `RLreplace`

4. Routinen der RLPC-Schritte - Stufe 2

Die zweite Stufe des RLPC-Verfahrens kondensiert Zeichen, isoliert Unternehmensformen, bereinigt Schreibweisen und entfernt Klammerausdrücke. Abbildung 4 zeigt den Ablauf dieser zweiten Stufe.

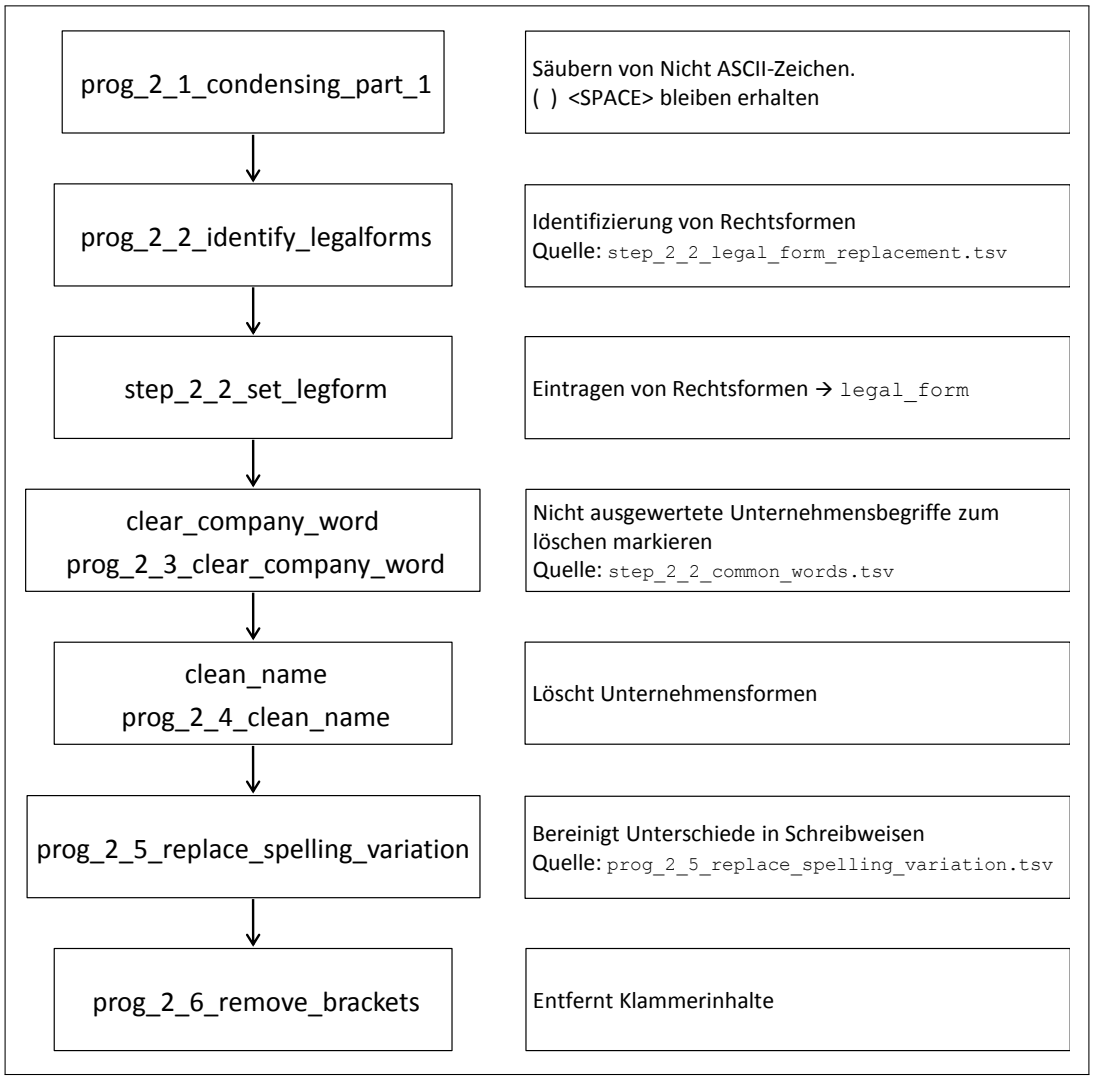

Abbildung 4: Ablaufplan: Funktionen Routinen der Stufe 2.

4.1. prog_2_1_condensing_part_1.do

Dieses Programm säubert `RLName` von ASCII-Sonderzeichen und anderen speziellen Zeichen mittels der Funktionen remove_ascii_special_chars und remove_special_chars. Diese Vorab-Säuberung beschleunigt die darauf folgende Regex-basierte Routine deutlich.

Anschließend werden alle Zeichen in `RLName`, die nicht A-Z, 0-9, (,) oder <Space> sind, unter Verwendung von char_condensing_one durch Leerzeichen ersetzt.

Schließlich wird bei öffnenden runden Klammern das Leerzeichen, welches unmittelbar auf das Klammersymbol folgt, entfernt. Weiter wird sichergestellt, dass vor jeder öffnenden runden Klammer ein Leerzeichen steht. Analog hierzu wird für schließende runde Klammern verfahren.

Verwendung: `prog_2_1_condensing_part_1`

Verwendete Routinen: `remove_ascii_special_chars`; `remove_special_chars`; `RLreplace`; `RLtrim`

4.2. prog_2_2_identify_legalforms.do

Hier werden die Rechtsformen von Unternehmen identifiziert und mit Hilfe der Routine legform_detect durch identifizierbare Kurzformen ersetzt. Dazu werden nacheinander die Rechtsformen am Anfang, am Ende und in der Mitte gesucht.

Da dieser Prozess recht zeitaufwendig ist, werden für diese Unter-Schritte die Zeiten gesondert ermittelt und in das Timelog eingetragen (begin_substep; end_substep). Nach jedem dieser Teilschritte wird mittels step_2_2_set_legform die Rechtsform in die Variable `legal_form` eingetragen.

Verwendung: `prog_2_2_identify_legalforms`

Verwendete Routinen: `legform_detect`; `step_2_2_set_legform`; `RLtrim`

4.3. prog_2_2_set_legform.do

Dieses do-file definiert das Programm `step_2_2_set_legform`, welches mittels der Funktion set_legform sämtliche identifizierten Kurzformen von Rechtsformen aus `RLName` in die (leere) Variable `legal_form` einträgt.

Verwendung: `step_2_2_set_legform`

Verwendete Routinen: `set_legform`

4.4. prog_2_3_clear_company_word.do

Hier wird die Routine `clear_company_word` definiert, die häufige, aber hier nicht ausgewertete Unternehmensbegriffe aus `RLName` durch den Platzhalter „«»“ ersetzt. Zu den so behandelten Unternehmensbegriffen gehören beispielsweise „INTERNATIONAL CORPORATION“ oder „GESELLSCHAFT“. Die doppelten spitzen Klammern „«»“ werden in Schritt 2.4 (Abschnitt 4.5) weiter behandelt.

Bei den Unternehmensbegriffen kann dabei angegeben werden, ob bei einer Länge des Unternehmensbegriffe unter 5 Zeichen dieser bei der Suche mit Leerzeichen abgetrennt werden soll (`type = 0`) oder nicht (`type = 1`). Anzahl und Position der Leerzeichen ist dabei abhängig von der Suchposition des Unternehmensbegriffes (Anfang, Mitte, Hinten) und dienen dazu, der unterschiedlichen Verwendung verschiedener Unternehmensbegriffe gerecht zu werden.

Die Liste der Unternehmensbegriffe ist:

`step_2_2_common_words.tsv`.

Diese Datei muss im Programmverzeichnis `$progdir` liegen und sollte der Länge der Unternehmensbegriffe nach absteigend (!) sortiert sein. So können Spezialformen besser erkannt werden, da sie vor den allgemeinen Formen abgefragt werden.

Die Liste enthält zwei Spalten, die durch das <TAB>-Zeichen getrennt sind (Dateikennung tsv). Sie hat das folgende Format:

<Unternehmensbegriff><TAB><type><CR LF> .

Zur Suche der Unternehmensbegriffe kann der Routine angegeben werden, ob diese am Anfang, am Ende oder in der Mitte von `RLName` suchen soll. Als Parameter für <replacetype> können deshalb "replace_at_end", "replace_at_beginning" oder "replace_in_middle" angegeben werden.

Verwendung: `clear_company_word "<replacetype>"`

Verwendete Routinen: –

Anschließend wird in diesem Schritt das eben beschriebene Programm `clear_company_word` nacheinander mit den Parametern "replace_at_end", "replace_at_beginning" oder "replace_in_middle" ausgeführt.

Verwendung: `prog_2_3_clear_company_word`

Verwendete Routinen: `clear_company_word`

4.5. prog_2_4_clean_name.do

Diese Routine definiert zum einen das Programm `clean_name`, welches den vorgegebenen String "«Unternehmensform»" in `RLName` löscht.

Verwendung: `clean_name "«Unternehmensform»"`

Verwendete Routinen: –

Anschließend wird clean_name für alle bekannten Unternehmensformen und „«»“ ausgeführt, um diese aus `RLName` zu entfernen.

Verwendung: `prog_2_4_clean_name`

Verwendete Routinen: `clean_name`; `RLtrim`

4.6. prog_2_5_replace_spelling_variation.do

Diese Routine erkennt einen Suchtext in `RLName` und ersetzt diesen in `RLName` durch einen vorgegebenen Ersatztext. Hier wird diese Routine verwendet um Schreibweisen von Unternehmenszusätzen zu vereinheitlichen (z. B.INTERNATIONALE → INTERNATIONAL).

Die Liste der Unternehmenszusätze sowie ihrer Ersatztexte ist:

`prog_2_5_replace_spelling_variation.tsv`.

Diese Datei muss im Programmverzeichnis `$progdir` liegen. Die Liste enthält Einträge in zwei Spalten, die durch das <TAB>-Zeichen getrennt sind (Dateikennung tsv). Sie hat das folgende Format:

<Suchtext><TAB><Ersatztext><CR LF> .

Verwendung: `prog_2_5_replace_spelling_variation`

Verwendete Routinen: –

4.7. prog_2_6_remove_brackets.do

Hier werden alle Inhalte von `RLName` gelöscht, die in runden Klammern stehen. Die runden Klammern werden dabei ebenfalls gelöscht. Der Inhalt dieser runden Klammern wurde bereits in Schritt 1.5 (Abschnitt 3.5) in die Variable `brackets` übertragen.

Verwendung: `prog_2_6_remove_brackets`

Verwendete Routinen: –

5. Routinen der RLPC-Schritte - Stufe 3

Die dritte und letzte Stufe des RLPC-Verfahrens kondensiert alle Zeichen in `RLName` und finalisiert so die nun RL-kompatible Variable. Abbildung 5 zeigt den Ablauf dieser dritten und letzten Stufe.

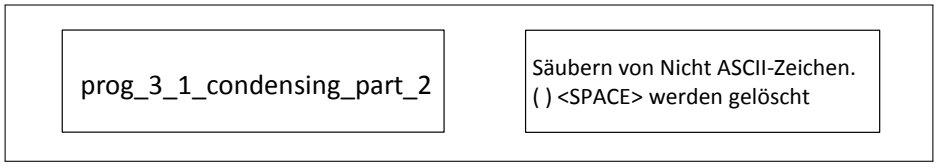

Abbildung 5: Ablaufplan: Funktionen Routinen der Stufe 3.

5.1. prog_3_1_condensing_part_2.do

Dieses Programm löscht alle Leerzeichen in `RLName`, sowie runde Klammersymbole („(“ und „)“). Anschließend werden sicherheitshalber nochmals alle Zeichen in `RLName`, die nicht A-Z oder 0-9 sind, unter Verwendung von char_condensing_two gelöscht.

Verwendung: `prog_3_1_condensing_part_2`

Verwendete Routinen: `RLreplace`; `char_condensing_two`

Literatur

Ehrenfeld, Wilfried (2015a): RegDemo: Aufbereitung und Zusammenführung der Akteursdaten - Technische Dokumentation der Routinen und Datensätze. IWH Technical Reports 1/2015.

Ehrenfeld, Wilfried (2015b): Research Explorer - Technische Dokumentation der Routinen. IWH Technical Reports 3/2015.

Ehrenfeld, Wilfried (2015c): RLPC: Record Linkage Pre-Cleaning - Technische Dokumentation der Routinen. IWH Technical Reports 2/2015.

Magerman, Tom, Bart Van Looy und Xiaoyan Song (2006): Data production methods for harmonized patent statistics: Patentee name harmonization. Katholieke Universiteit Leuven MSI 0605.

Schnell, Rainer, Tobias Bachteler und Stefan Bender (2004): A Toolbox for Record Linkage. In: *Austrian Journal of Statistics* 33.1–2, S. 125–133.

Schnell, Rainer, Tobias Bachteler und Jörg Reiher (2005): MTB: Ein Record-Linkage-Programm für die empirische Sozialforschung. ZA-Information 2005, No. 56.

Titze, Mirko, Wilfried Ehrenfeld, Matthias Piontek und Gunnar Pippel (2015): „Netzwerke zwischen Hochschulen und Wirtschaft: Ein Mehrebenenansatz“. In: Schrumpfende Regionen - dynamische Hochschulen: Hochschulstrategien im demografischen Wandel. Hrsg. von Michael Fritsch, Peer Pasternack und Mirko Titze. Wiesbaden: Springer Fachmedien. Kap. 11, S. 213–234.

A. Anhang

A.1. Code Statistics

Stand: Juli 2015

Modul	Anzahl Zeilen
Grundlegende Routinen	
PreCleaning.do	150
Programs_Load.do	34
Programs_Basic_Routines.do	71
Programs_Module_Management.do	126
Programs_ASCII_Routines.do	125
Programs_Legform_Routines.do	91
Routinen RLPC Stufe 1	
prog_1_1_uppercase.do	98
prog_1_2_replace_smgl_codes.do	52
prog_1_3_replace_coded_chars.do	50
prog_1_4_replace_bracket_symbols.do	28
prog_1_5_get_bracket_content.do	38
prog_1_6_replace_ascended_chars.do	61
prog_1_7_replace_and.do	23
Routinen RLPC Stufe 2	
prog_2_1_condensing_part_1.do	34
prog_2_2_identify_legalforms.do	37
prog_2_2_set_legform.do	57
prog_2_3_clear_company_word.do	82
prog_2_4_clean_name.do	70
prog_2_5_replace_spelling_variation.do	44
prog_2_6_remove_brackets.do	25
Routinen RLPC Stufe 3	
prog_3_1_condensing_part_2.do	27
Anzahl Module	27
Codezeilen Gesamt	**1323**

Leibniz-Institut für Wirtschaftsforschung Halle – IWH

HAUSANSCHRIFT: Kleine Märkerstraße 8, D-06108 Halle (Saale)

POSTANSCHRIFT: Postfach 11 03 61, D-06017 Halle (Saale)

TELEFON: +49 345 7753 60 TELEFAX +49 345 7753 820

INTERNET: www.iwh-halle.de
I S S N : 2 3 6 5 - 9 0 7 6