**Bachelorarbeit**

# String Matching von historischen Toponymen

Martin Becker

| | |
|---|---|
| Abgabedatum: | 20. Mai 2016 |
| Betreuer: | Prof. Dr. Alexander Wolff |
| | Benedikt Budig |
| | Thomas van Dijk |


Julius-Maximilians-Universität Würzburg
Lehrstuhl für Informatik I
Algorithmen, Komplexität und wissensbasierte Systeme

# Zusammenfassung

In dieser Bachelorarbeit wird ein Modul eines schon vorhandenen Konzeptes zur Digitalisierung alter Landkarten entwickelt. Hierbei soll das Modul die Städtenamen alter Landkarten übergeben bekommen und diese den aktuellen Städtenamen zuordnen.

Es werden nach einer Einleitung ins Thema schon bekannte Methoden zur Abstandsbestimmung zweier Worte, hier Städtenamen, vorgestellt und bekannte Verknüpfungen dieser mit entsprechenden Datenbanken diskutiert. Außerdem werden komplexere Verfahren zum Matchen von (Städte-)Namen kurz vorgestellt. Hierauf aufbauend wird das Konzept des entwickelten Algorithmus erklärt und die dazu benötigten Daten aus einer Datenbank extrahiert. Danach wird durch Vergleiche dieses Algorithmus mit dem Levenshtein-Algorithmus entschieden, wie gut der gefundene Algorithmus ist. Zum Schluss werden noch anstehende Arbeiten am Konzept und Verbesserungsvorschläge für den Algorithmus vorgestellt.

# Inhaltsverzeichnis

# 1 Einleitung

Die Digitalisierung der Welt schreitet in großen Schritten voran. Dies hat zum Beispiel dazu geführt, dass, obwohl noch vor einem halben Jahrhundert Informationen fast ausschließlich in Büchern in der Bibliothek zu erhalten waren, in der heutigen Zeit die Informationsbeschaffung über das Internet und in digitaler Form kaum noch wegzudenken ist. Große Teile des Wissens der Menschheit sind dort zu finden.

Trotzdem ist die Digitalisierung noch nicht abgeschlossen und wird in vielen Bereichen weiter vorangetrieben. Beispielsweise wurden die Karten und Pläne des Stadtarchivs Würzburg digitalisiert, um den Bestand zu sichern und Interessierten und Forschern einen leichteren Zugriff auf die vorhandenen Daten zu ermöglichen [1]. Dennoch hilft eine Digitalisierung, die nur aus dem Einscannen und Speichern in einer Bilddatei besteht, oft nur wenig weiter. Nichts ändert es an der Tatsache, dass die Daten noch einzeln ausgelesen und unter Umständen verglichen werden müssen. Häufige Fragestellungen, wie „Wie hat sich die Städtelandschaft im Laufe der Jahre verändert, welche Städte sind verschwunden?“ erfordern Stunden und Tage an Arbeit, weil zu untersuchende Städte einzeln überprüft und auf Landkarten anderer Jahre gefunden werden müssen. Wären jedoch alle Metadaten, wie enthaltene Städte mit Koordinaten, Landschaftsformen, Flüsse, digital gespeichert (und nicht in Form einer Bilddatei), wäre es möglich, zeiteffizient die gesuchten Resultate zu erzielen, also mit vorherigen Beispiel diejenigen Städte zu finden, die im Laufe der Jahre verschwunden sind.

Abbildung 1.1 zeigt einen Ausschnitt aus einer alten Landkarte [2] aus der „Franconica“-Kartensammlung der Universitätsbibliothek Würzburg [3] eines kleinen Gebiets um Würzburg. Die Bezeichnungen Lengfeld, Gernbrun, Ransaker, Versbach und Würtzburg entsprechen den heutigen Stadtteilen/Gemeinden Lengfeld, Gerbrunn, Randersacker, Versbach und Würzburg (siehe aktuelle Landkarte 1.2). Es lässt sich leicht erkennen, dass sich die Städtenamen im Laufe der Jahre verändert haben, aber immer noch eine gewisse Ähnlichkeit aufweisen.

Ziel dieser Bachelorarbeit ist es, aufbauend auf diesen Ähnlichkeiten, eine Zuordnung zwischen den historischen und modernen Städtenamen durchzuführen. Zuerst werden schon bekannte Abstandsmaße für Zeichenketten analysiert (siehe Kapitel 2), woraus eine Grundidee für einen Algorithmus entwickelt wird (siehe Kapitel 3). Danach werden

[1] `http://www.wuerzburg.de/de/themen/kultur-bildung-kulturangebot/denkmalpflegeundstadtgeschichte/stadtarchiv\/bestaendeundbenutzung/28845.Digitalisierung-der-Karten--und-Planbestnde-des-Stadtarchivs-Wrzburg.html`

[2] Johann Heinrich Seyfried und Johann Jakob Schollenberger. Das Bisthum Wurtzburg In Francken / I. H. S. delineavit. I. Schollenb. sc./Nürnberg: Joh. Hoffman, 1676 - Kupferstich, koloriert, 52 x 38 cm - Maßstab: ca. 1:280000, 1676. 36/A 10.12.

[3] `http://www.franconica-online.de/`


**Abb. 1.1:** Ausschnitt aus einer historischen Landkarte (Das Bisthum Wurtzburg, 1676)


**Abb. 1.2:** Aktueller Kartenauschnitt mit Openstreetmap

Wortstrukturen aus einer Datenbank mit historischen und modernen Städtenamen extrahiert, um zu sehen, wie stark sich welche Wortstrukturen ähneln (siehe Kapitel 4). Als letztes wird noch mit verschiedenen Experimenten überprüft, inwieweit der entwickelte Algorithmus besser als schon bekannte Algorithmen ist (siehe Kapitel 5). Der entwickelte Algorithmus soll ein Modul in einem Workflow zur Digitalisierung von alten Landkarten realisieren [Bud15]. Hierbei bekommt das Modul die, mittels einer OCR-Software aus der Landkarte extrahierten, historischen Städtenamen und gibt die dazugehörigen heutigen Städtenamen aus. Dies wird durch einen Abgleich mit einer Datenbank und dem Finden des nächsten Nachbarn, also demjenigen Stadtnamen, der dem gegebenen Stadtnamen am ähnlichsten ist, realisiert. Abschließend wird noch ein Ausblick auf mögliche weitere Verbesserungen und Forschungen zu dem entwickelten Algorithmus im Speziellen, aber auch zu dem Konzept, dessen Teil dieser Algorithmus ist, gegeben (siehe Kapitel 6).

# 2 Stand der Technik

Im Folgenden wird der bisherige Stand der Technik dargelegt. Es werden zuerst drei Verfahren vorgestellt, die zwei Worte einlesen und deren Ähnlichkeit bestimmen. Anschließend wird ein Verfahren vorgestellt, das auf großen Datenbanken basiert, um alle möglichen Schreibalternativen für Worte abzudecken. Zuletzt werden schon vorhandene Forschungen in diesem Bereich und komplexere Verfahren vorgestellt.

## 2.1 Metrik und Levenshtein

In der Mathematik bezeichnet eine Metrik eine Abstandsfunktion. Diese Funktion $d : X^2 \to \mathbb{R}_+$ auf zwei Elementen einer Menge, deren Zielbereich die nicht-negativen reellen Zahlen sind, muss folgende Axiome erfüllen, um eine Metrik zu sein [Her86]:

| | | |
|---|---|---|
| (Positive Definitheit) | $d(x,y) = 0 \Leftrightarrow x = y$ | |
| (Symmetrie) | $d(x,y) = d(y,x)$ | für alle $x, y \in X$ |
| (Dreiecksungleichung) | $d(x,y) \leq d(x,z) + d(y,z)$ | für alle $x, y, z \in X$ |

Die hier betrachtete Distanz auf der Menge aller Worte, die später als Levenshtein-Distanz bekannt wurde, ist eine solche Metrik. Sie wurde zuerst in einem Paper von Vladimir I. Levenshtein [Lev65] erwähnt, bei die automatische Verbesserungen von Fehlern in Worten diskutiert wird. Bei den Fehlern hat man sich auf das fälschlicherweise Einfügen und Löschen eines Buchstaben und dem Ersetzen eines Buchstaben durch einen anderen beschränkt. Hierbei wurde sie als die minimale Anzahl der Einfüge-, Lösch- und/oder Ersetzungsoperationen, die benötigt werden, um von dem einen Wort zu dem anderen Wort zu gelangen, definiert. Im Laufe der Zeit gab es viele Abwandlungen dieser Distanz und sie wurde unter anderem von Wagner [Fis74] mit Hilfe des folgenden Algorithmus für beliebige Worte berechnet:

1. $D[0,0] := 0$;
2. for i :=1 to |A| do $D[i,0] := D[i-1,0] + \gamma(A < i > \to \Delta)$;
3. for j:=1 to |B| do $D[0,j] := D[0,j-1] + \gamma(\Delta \to B < j >)$;
4. for i:=1 to |A| do
5. for j:=1 to |B| do
6. $m_1 := D[i-1,j-1] + \gamma(A < i > \to B < j >)$;
7. $m_2 := D[i-1,j] + \gamma(A < i > \to \Delta)$;
8. $m_3 := D[i,j-1] + \gamma(\Delta \to B < j >)$;
9. $D[i,j] := min(m_1, m_2, m_3)$;

Dieser Algorithmus hat für das Einfügen und Löschen jedes einzelnen Buchstabens und auch für jede Austauschoperation von zwei Buchstaben eine andere Gewichtung, die durch die Funktion $\gamma$ gespeichert wird. Er arbeitet das Ausgangs- und das Zielwort Buchstabe für Buchstabe ab und speichert jeweils den bisher kürzesten Weg (beziehungsweise dessen Länge) vom Ausgangswort bis zum i. Buchstaben zum Zielwort bis zum j. Buchstaben in einer Matrix $D$ an der Stelle $D[i, j]$. In jedem Schritt überprüft der Algorithmus wieder, welchen von den bisherigen Teilwegen er mit einer der drei Operationen erweitern muss, um den kürzesten Weg zwischen den beiden betrachteten Teilworten zu finden. Die Länge des kürzesten Weges befindet sich nach Durchlauf des Algorithmus in der Matrix an der Stelle $D[|A|, |B|]$, da dort die Teilworte den ganzen Worten entsprechen.

| | | W | Ü | R | Z | B | U | R | G |
|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| W | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| I | 2 | 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| R | 3 | 2 | 2 | 1 | 2 | 3 | 4 | 5 | 6 |
| T | 4 | 3 | 3 | 2 | 2 | 3 | 4 | 5 | 6 |
| Z | 5 | 4 | 4 | 3 | 2 | 3 | 4 | 5 | 6 |
| B | 6 | 5 | 5 | 4 | 3 | 2 | 3 | 4 | 5 |
| U | 7 | 6 | 6 | 5 | 4 | 3 | 2 | 3 | 4 |
| R | 8 | 7 | 7 | 6 | 5 | 4 | 3 | 2 | 3 |
| G | 9 | 8 | 8 | 7 | 6 | 5 | 4 | 3 | 2 |

**Tab. 2.1:** Berechnungstabelle beim Levenshtein-Algorithmus für die Worte Wirtzburg und Würzburg

| | | W | Ü | R | Z | B | U | R | G |
|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | | | | | | |
| W | 1 | 0 | 1 | | | | | | |
| I | 2 | 1 | | | | | | | |
| R | 3 | 2 | | | | | | | |
| T | 4 | 3 | | | | | | | |
| Z | 5 | 4 | | | | | | | |
| B | 6 | 5 | | | | | | | |
| U | 7 | 6 | | | | | | | |
| R | 8 | 7 | | | | | | | |
| G | 9 | 8 | | | | | | | |

**Tab. 2.2:** Berechnungstabelle beim Levenshtein-Algorithmus für die Worte Wirtzburg und Würzburg bis zu den Buchstaben Ü beziehungsweise I

Wir verwenden als Basis für unseren Algorithmus eine vereinfachte Version der Levenshtein-Distanz, die folgende Operationen zulässt:

Einfügen mit dem Gewicht 1 (unabhängig von den Buchstaben)

Löschen mit dem Gewicht 1 (unabhängig von den Buchstaben)

Übernahme mit dem Gewicht 0 (wenn beide Buchstaben gleich sind)

Ersetzen mit dem Gewicht 1 (wenn beide Buchstaben verschieden sind)

Der Algorithmus zur Berechnung des Abstandes geht hierbei entsprechend dem vorgestellten Algorithmus von Wagner vor. Ein Beispiel, wie die Matrix bei der Berechnung des Abstandes der Worte Wirtzburg und Würburg aussieht, befindet sich in der Tabelle 2.1. Hierbei wurden zur Verdeutlichung die Buchstaben der betrachteten Worte an den Rändern der Matrix eingefügt. Tabelle 2.2 zeigt, wie die Matrix bei der Berechnung der Länge des kürzesten Weges von „Wi“ nach „Wü“ aussieht. Das Ergebnis wird dann in die rot hinterlegte Zelle geschrieben. Da „Ü“ und „I“ nicht der gleiche Buchstabe sind, gibt es drei mögliche Operationen: Einfügen, Löschen und Ersetzen. Beim Einfügen wird der Zellenwert links davon genommen und mit dem Einfügegewicht 1 addiert. Beim Löschen wird die Zelle oberhalb betrachtet und der Zellenwert um 1 erhöht und beim Ersetzen wird die Zelle links oberhalb betrachtet und der Wert um 1 erhöht. Nun wird das Minimum dieser drei möglichen Weglängen genommen und in die rot hinterlegte Zelle geschrieben. Dieser Vorgang wird, wie der Algorithmus von Wagner angibt, mit jeder Zelle wiederholt, bis das Ergebnis in der Zelle ganz rechts unten steht.

## 2.2 n-gram

Eine Alternative beim Finden eines nächsten Nachbarn ist das n-gram-Verfahren, welches im Buch Linguistisches Identity Matching beschrieben wird [Lis11]. Hierbei werden (wie bei der Levenshtein-Distanz) immer zwei Worte miteinander verglichen. Beide Worte werden in Teilzeichenketten (Substrings) der Länge n aufgeteilt. Je mehr identische Substrings in beiden Worten vorkommen, desto ähnlicher sind sich die beiden Worte. Dieses Verfahren wird für alle Kandidaten des nächsten Nachbarn durchgeführt und derjenige Kandidat mit der höchsten Ähnlichkeit wird als nächster Nachbar festgesetzt.

Vergleicht man Wirtzburg und Würzburg mit dem 2-gram-Verfahren (siehe Tabelle 2.3), so merkt man, dass, obwohl beide Worte sich sehr ähnlich sind, das Verfahren nur vier gleiche Bigramme liefert. Untersucht man jedoch Wirtzburg und Moritzburg, zwei komplett unterschiedliche Städte, so liefert das 2-gram-Verfahren fünf gleiche Bigramme (siehe Tabelle 2.4). Ähnliche Ergebnisse würde das n-gram-Verfahren für $n \geq 3$ liefern. Es gibt also Fälle, bei denen das n-gram-Verfahren alleine nur sehr unzuverlässige Ergebnisse erzielt.

| | Bigramme | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Wirtzburg | wi | ir | rt | tz | zb | bu | ur | rg |
| Würzburg | wü | ür | rz | | | | | |

**Tab. 2.3:** Bigramme für Wirtzburg und Würzburg

| | Bigramme | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Wirtzburg | wi | ir | rt | | tz | zb | bu | ur | rg |
| Moritzburg | mo | or | ri | it | | | | | |

**Tab. 2.4:** Bigramme für Wirtzburg und Moritzburg

## 2.3 Phonetik

Ein weiterer Ansatz ist das Problem des nächsten Nachbarn aus phonetischer Sicht anzugehen. Hierzu wird festgelegt, dass dasjenige Wort dem untersuchten Wort am nächsten ist, dessen Aussprache beziehungsweise dessen Lautbild ihm am ähnlichsten ist. Eine Umsetzung hiervon für die deutsche Sprache liefert die Kölner Phonetik [Wil05]. Dazu wird das untersuchte Wort in mehreren Schritten in eine Lautsprache umgewandelt, in der ähnliche Laute das gleiche Symbol haben:

1. Die Buchstaben des untersuchten Wortes werden mit Hilfe der Tabelle 2.5 ersetzt.
2. Alle mehrfach hintereinander vorkommenden Ziffern werden auf eine reduziert.
3. Alle Ziffern 0 werden gelöscht.

Die Kodierung von Wirtzburg und Würzburg nach den einzelnen Schritten sieht man in Tabelle 2.6. Sie haben beide die gleiche Kodierung sind sich also sehr ähnlich.

Ein weiteres phonetisches Verfahren ist Soundex. Dieses ist aber für alle Sprachen konzipiert, deshalb sehr allgemein und nicht auf die deutsche Sprache spezialisiert, weshalb wir hier nicht genauer darauf eingehen.

## 2.4 Thesauri

Ein Thesaurius ist, vereinfacht ausgedrückt, eine Datenbank, die nicht nur die gesuchten Worte, sondern auch verschiedene Variationen dessen enthält. Diese Variationen werden häufig per Hand hinzugefügt und sind deswegen sehr lückenhaft [Lis11]. Es gibt jedoch auch generative Algorithmen, die nach vorgegebenen Regeln diese Variationen generieren. Ein betrachtetes Wort wird also einem Wort zugeordnet, wenn es ihm oder einer seiner Variationen entspricht. Da, wie schon erwähnt, diese Datenbanken oft lückenhaft sind, ist es auch möglich Thesauri in Kombination mit einem Abstandsmaß zu kombinieren, indem man zu jedem Wort und jeder Variation den nächsten Nachbarn findet. Im Kontext von Städtenamen entspricht ein Thesaurus also nicht nur einer Datenbank von modernen Städtenamen, sondern auch ihrer historischen Namen.

| Zeichen | Kontext | Symbol |
|---|---|---|
| A,E,I,J,Y,O,U | im Anlaut | 0 |
| H | | - |
| B,P | | 1 |
| D,T | nicht vor C,S,Z | 2 |
| F,P,H,V,W | | 3 |
| G,K,Q | | 4 |
| C | im Anlaut, vor A,H,K,L,O,Q,R,U,X | 4 |
| | ansonsten, vor A,O,U,H,K,X,Q | |
| X | wenn nicht nach C,K,Q | 48 |
| L | | 5 |
| M,N | | 6 |
| R | | 7 |
| S,Z | | 8 |
| C | im Anlaut, nicht vor | |
| | A,H,K,L,O,Q,R,U,X folgt | |
| | ansonsten, nicht vor A,O,U,H,K,X,Q | |
| | nach S,Z | |
| D,T | vor S,C,Z | |
| X | nach C,K,Q | |

**Tab. 2.5:** Ersetzungsregeln der Kölner Phonetik nach Postel (1969)

| Schritte | Wirtzburg | Würzburg |
|---|---|---|
| 1. | 307881074 | 20781974 |
| 2. | 30781074 | 30781074 |
| 3. | 37817 | 378174 |

**Tab. 2.6:** Kodierung von Wirtzburg und Würzburg nach den einzelnen Schritten der Kölner Phonetik

## 2.5 praktische Anwendungen

Da Personennamen das größte Feld für die Verwendung von Name-Matching-Verfahren sind, wurden die meisten größeren Anwendungsverfahren auf dieses Gebiet spezialisiert. Im Buch Linguistisches Identity Matching [Lis11] wurde ein Ansatz hierzu genauer beschrieben. Ähnlich wie bei den phonetischen Ansätzen werden hier mit „linguistischen Regelsets“ aus den zu vergleichenden Namen „Similarity Keys“ generiert. Diese Regelsets gehen aber in ihrem Umfang weit über das hinaus, was die Kölner Phonetik oder auch Soundex aufweist. Kurz zusammengefasst ist dieses Verfahren eine Kombination aus einem erweiterten phonetischen Ansatz kombiniert mit einem Verfahren zur Bestimmung eines Abstandes.

Eine umfassende Übersicht über Verfahren zur Überprüfung von phonetischer Ähnlichkeit findet sich in der Magisterarbeit von Wilz [Wil05]. Obwohl auch auf Verfahren

zum Vergleich von Zeichenketten eingegangen wurde, lag das Hauptaugenmerk eindeutig auf den phonetischen Verfahren und deren Vergleich untereinander.

Der klassische Algorithmus zur Berechnung der Levenshtein-Distanz ist vergleichsweise langsam. Deshalb wurde von Ukkonen [Ukk85] ein Algorithmus entwickelt, der die Laufzeit von vorher $O(mn)$ auf $O(s \cdot min(m, n))$ drücken konnte. Hierbei steht $m$ für die Länge der Eingangszeichenkette, $n$ für die Länge der Zielzeichenkette und $s$ für die Levenshtein-Distanz zwischen beiden.

# 3 Ein Algorithmus für String-Matching auf historischen Toponymen

Wie unter anderem im Buch Linguistitsches Identity Matching [Lis11] festgestellt wurde, wird ein Verfahren für sich alleine nicht dem gestellten Problem des Matchen der Städtenamen gerecht. Dies wird auch im experimentellen Teil dieser Bachelorarbeit belegt (siehe Kapitel 5). Die Idee ist es nun ähnlich wie bei den anderen vorgestellten kombinierten Verfahren, die besten Eigenschaften der bekannten Ansätze zu kombinieren.

Dies soll in folgender Form geschehen:

Der Levenshtein-Algorithmus wird als Ausgangspunkt genommen und dahingehend erweitert, dass er nicht nur die Möglichkeit zulässt, einzufügen, zu löschen oder zu ersetzen, sondern auch Übergänge bis zu der Länge 3 zuzulassen. Darunter ist zu verstehen, dass eine Zeichenkette der Länge $x$ in eine Zeichenkette der Länge $y$ umgewandelt wird, wobei gilt: $1 \leq x \leq 3$ und $1 \leq y \leq 3$. Dies wird damit begründet, dass die Kölner Phonetik (siehe Abschnitt 2.3) keine Laute einer längeren Länge als 3 berücksichtigt. Es wäre möglich gewesen, die Übergänge auf Basis der Kölner Phonetik zu gewichten, jedoch ist diese allgemein auf der Basis der deutschen Sprache aufgebaut und nicht auf das Problem mit Städtenamen spezialisiert. Deshalb ist es wichtig, selbst Informationen zur Häufigkeit solcher Übergänge aus einer Datenbank mit modernen und dazugehörigen historischen Städtenamen zu extrahieren, was in Kapitel 4 näher erläutert wird. Hierbei wird jedoch jeder historischer Stadtname nur mit dem dazugehörigen modernen Stadtnamen verglichen und nicht die historischen Stadtnamen unter sich. Dies wird so durchgeführt, da Lautverschiebungen, also Übergänge, gerichtet sind, das heißt durch die Lautverschiebungen die historischen Stadtnamen zu den modernen Stadtnamen wurden und nicht umgekehrt. Da bei den historischen Stadtnamen in der Datenbank aber nur schwer zu entscheiden ist, welcher Stadtname nun älter ist, werden sie nur mit dem modernen Stadtnamen verglichen. Dabei ist darauf zu achten, dass der betrachtete Algorithmus keine Metrik mehr umsetzt, da er die Eigenschaft der Symmetrie verletzt (siehe Abschnitt 2.1).

Der hierbei gewonnene Edit-Distance-Algorithmus wird im Weiteren verwendet, um die jeweiligen Abstände zwischen dem zu untersuchenden alten Stadtnamen, der durch die vorherigen Verfahren [Bud15] aus einer alten Landkarte extrahiert wurde, und möglichen neuen Städtenamen zu untersuchen. Bei Experimenten in Abschnitt 5.2 werden auch andere bekannte, alte Schreibweisen als Vergleich herangezogen, was eine Ähnlichkeit zu Thesauri aufweist (siehe Abschnitt 2.4). Hierbei wird dieselbe Datenbank verwendet, aus der auch die Häufigkeiten der Übergänge extrahiert wurden. Derjenige neue Stadtname mit dem geringsten Abstand wird dem alten Stadtnamen zugeordnet. Bei mehreren ähnlich guten Treffern wird, entsprechend dem schon erwähnten Konzept [Bud15], der Benutzer aufgefordert aus einer Auswahl den wahrscheinlichsten Treffer auszuwählen.

# 4 Extrahieren der Gewichte

Im Folgenden wird erläutert, wie die Gewichtung der Übergänge aus der gewählten Datenbank [Zim15] (Lorscher Kodex) extrahiert werden und eine kurze Bewertung der Ergebnisse gegeben.

Zuerst wird an einem Beispiel beschrieben, was diese Übergänge überhaupt sind: Eine alte Schreibweise für die Stadt Würzburg ist Wirzburg. Bei Levenshtein entspricht dies (beim Weg von Wirzburg nach Würzburg) nur Übernahmen gleicher Buchstaben außer einem Löschen von „i“ und einem Einfügen von „ü“. Was aber eigentlich geschieht ist eine Lautverschiebung von „i“ nach „ü“, also ein Übergang. Werden die Schreibweisen Würzburg und Würtzburg betrachtet, so könnte man dies als Einfügen des Buchstaben „t“ betrachten. Phonetisch gesehen liegt es jedoch am darauf folgenden Buchstaben „z“, dass die beiden Wörter immer noch sehr ähnlich sind (beide Beispiele siehe auch Kölner Phonetik 2.3). Wir haben hier also einen phonetischen Übergang von „z“ nach „tz“. Wir erlauben nun, wie im vorigen Kapitel 3, schon beschrieben auch Übergänge bis zur Länge drei mit entsprechenden Gewichten, die nun aus der Analyse der Datenbank extrahiert werden sollen. Übergänge, die von einem leeren Wort ausgehen oder zu einem leeren Wort führen, werden auch weiterhin als Einfügen beziehungsweise Löschen mit dem Gewicht 1 betrachtet. Hierbei gehen wir davon aus, dass je häufiger ein Übergang auftritt, desto phonetisch ähnlicher sich die beiden Zeichenketten sind. Dementsprechend geringer sollte deren Gewichtung sein.

Unser Algorithmus betrachtet nun alle möglichen Kombinationen aus historischen Stadtnamen und dazugehörigen modernen Stadtnamen. Für jede Kombination werden nun alle verschiedenen Übergänge ausgelesen und gezählt. Das geschieht folgendermaßen: Zuerst werden bei beiden Stadtnamen die Bereiche markiert, die übereinstimmen, also bei Levenshtein (siehe Abschnitt 2.1) übernommen werden würden. Betrachtet werden nun die Abschnitte dazwischen, die sich also unterscheiden und den Übergängen entsprechen. Jeder betrachtete Abschnitt entspricht einem Übergang von einer Zeichenkette a der Länge $x \geq 0$ in eine Zeichenkette b der Länge $y \geq 0$. Das Auslesen der Gewichte wird nun in mehrere Fälle geteilt:

**Fall 1:** $x > 3 \vee y > 3$**:**

Da wir nur Übergänge maximal der Länge 3 betrachten, wird der Abschnitt in diesem Fall ignoriert. Es wird angenommen, dass dieser Abschnitt dadurch entstanden ist, dass die beiden verglichenen Worte sich phonetisch zu unterschiedlich sind und deshalb für das Extrahieren von Gewichten nicht hilfreich sind.

**Fall 2:** $(x = 3 \wedge y \leq 3) \vee (x \leq 3 \wedge y = 3)$**:**

Wenn $x \neq 0$ und $y \neq 0$ gilt, wird **der Zähler für den Übergang von a nach**

**b um 1 erhöht.** Wenn $x = 0$ oder $y = 0$ gilt, wird dies als eine Aneinanderreihung von Einfüge- beziehungsweise Löschoperationen angesehen und hat eine dementsprechende Gewichtung.

**Fall 3:** $(x = 2 \wedge y \leq 2) \vee (x \leq 2 \wedge y = 2)$**:**
Wenn $x \neq 0$ und $y \neq 0$ gilt, wird **der Zähler für den Übergang von a nach b um 1 erhöht.** Wenn $x = 0$ oder $y = 0$ gilt, wird dies als eine Aneinanderreihung von Einfüge- beziehungsweise Löschoperationen angesehen und hat eine dementsprechende Gewichtung.

An obigem Beispiel (Würtzburg und Würzburg) erkennt man, dass der Übergang von einer Zeichenkette in eine andere oft mit der direkten Umgebung des Übergangs zusammenhängt. Da es jedoch schwierig ist, festzustellen, welcher Übergang im phonetischen Sinne, also welche Zeichenketten, deren Lautbild ähnlich ist, ineinander übergehen, auftritt, werden alle möglichen Übergänge betrachtet. Dabei darf natürlich die Beschränkung von 3 für die Länge des Überganges nicht überschritten werden. Hier hat die Eingangs- oder die Zielzeichenkette eine Länge von 2, wobei die jeweils andere maximal eine Länge von 2 hat. Es ist also möglich, entweder vorne oder hinten (sowohl an die Eingangs-, als auch an die Zielzeichenkette) ein Zeichen anzuhängen, ohne die Beschränkung zu überschreiten. Genau das wird jetzt durchführen.

Wenn Eingangszeichenkette a und somit auch die Zielzeichenkette b nicht am Anfang ihrer jeweiligen Stadtnamen liegen, existiert ein Zeichen c, sodass die Zeichenketten ca und cb in den jeweiligen Stadtnamen liegen. **Es wird der Zähler für den Übergang von ca nach cb um 1 erhöht.**

Wenn Eingangszeichenkette a und somit auch die Zielzeichenkette b nicht am Ende ihrer jeweiligen Stadtnamen liegen, existiert ein Zeichen c, sodass die Zeichenketten ac und bc in den jeweiligen Stadtnamen liegen. **Es wird der Zähler für den Übergang von ac nach bc um 1 erhöht.**

**Fall 4:** $x \leq 1 \vee y \leq 1$**:**
Wenn $x \neq 0$ und $y \neq 0$ gilt, wird **der Zähler für den Übergang von a nach b um 1 erhöht.** Wenn $x = 0$ oder $y = 0$ gilt, wird dies als Einfüge- beziehungsweise Löschoperation angesehen und hat eine dementsprechende Gewichtung.

Es wird ähnlich wie im Fall 3 vorgegangen, nur dass hier 2 zusätzliche Zeichen gefügt werden können, bis die Beschränkung der Länge 3 erreicht ist. Werden jedoch vorne oder hinten 2 Zeichen angefügt, ist es möglich, dass dieses zweite Zeichen Teil eines anderen Übergangs ist (da die Übergänge minimal von nur einem, in beiden Stadtnamen übereinstimmenden Zeichen, getrennt sind). Dies muss extra überprüft werden.

Wenn Eingangszeichenkette a und somit auch die Zielzeichenkette b nicht am Anfang ihrer jeweiligen Stadtnamen liegen, existiert ein Zeichen c, sodass die Zeichenketten ca und cb in den jeweiligen Stadtnamen liegen. **Es wird der Zähler für den Übergang von ca nach cb um 1 erhöht.** Liegen zusätzlich noch ca und

cb nicht am Anfang ihrer jeweiligen Stadtnamen und gilt für die um ein Zeichen erweiterten Teilzeichenketten dca und eca, dass $d = e$, dann wird **der Zähler für den Übergang von dca nach dcb um 1 erhöht.**

Wenn Eingangszeichenkette a und somit auch die Zielzeichenkette b nicht am Ende ihrer jeweiligen Stadtnamen liegen, existiert ein Zeichen c, sodass die Zeichenketten ac und bc in den jeweiligen Stadtnamen liegen. **Es wird der Zähler für den Übergang von ac nach bc um 1 erhöht.** Liegen zusätzlich noch ac und bc nicht am Ende ihrer jeweiligen Stadtnamen und gilt für die um ein Zeichen erweiterten Teilzeichenketten acd und bce, dass $d = e$, dann wird **der Zähler für den Übergang von acd nach bce um 1 erhöht.**

Wenn Eingangszeichenkette a und somit auch die Zielzeichenkette b nicht am Anfang und auch nicht am Ende ihrer jeweiligen Stadtnamen liegen, existieren Zeichen c und d, sodass die Zeichenketten cad und cbd in den jeweiligen Stadtnamen liegen. **Es wird der Zähler für den Übergang von cad nach cbd um 1 erhöht.**

Am Beispiel der Stadtnamen Wirtzburg und Würzburg wurden in Tabelle 4.1 beispielhaft die Übergänge bestimmt.

Wenn Zeichenkette a viel häufiger als Zeichenkette b in der deutschen Sprache vorkommt, so kann es sein, dass a häufiger einen Übergang in eine Zeichenkette c hat als b, obwohl b vielleicht phonetisch gesehen c ähnlicher ist. Deshalb wird nicht nur die absoluten Anzahl an Übergängen von einer Zeichenkette in einer andere betrachtet, sondern zusätzlich noch gezählt, wie oft jede Zeichenkette in eine beliebige andere übergeht. Dadurch ist es möglich, Übergänge einer Zeichenkette d in eine Zeichenkette e zu Übergängen von d in eine beliebige andere Zeichenkette in ein Verhältnis zu setzen und somit eine relative Häufigkeit zu berechnen.

In Tabelle 4.2 sind alle Übergänge, die mindestens 10 mal auftreten, abgebildet. Abgesehen von ein paar Ausnahmen entsprechen diese in der Kölner Phonetik (siehe Abschnitt 2.3) der gleichen Kodierung. Oft verschwinden auch Vokale, was aber kein Unterschied zur Kölner Phonetik darstellt, da dort diese in Schritt 3 gelöscht werden. Die einzigen Übergänge, die mindestens 10 mal auftreten und sich nicht mit der Kölner Phonetik erklären lassen sind: „a->ad“, „e->er“, „s->rs“ und „es->ers“. Auffällig hierbei ist, dass sie alle einem Einfügen von einem Buchstaben entsprechen und in Kombination mit in der deutschen Sprache häufig auftretenden Buchstaben vorkommen (Vokalen und dem Buchstaben „s“).

| Übergang: i->ü | | | | |
|---|---|---|---|---|
| Hinzunahme von: | | 1 vorne | 1 hinten | 1 vorne und 1 hinten |
| eingetragende Übergänge: | i->ü | wi->wü | ir->ür | wir->wür |
| Übergang: t->{} | | | | |
| Hinzunahme von: | 1 vorne | 1 hinten | 2 hinten | 1 vorne und 1 hinten |
| eingetragende Übergänge: | rt->r | tz->z | tzb->rzb | rtz->rz etc |

**Tab. 4.1:** Übergänge am Beispiel von Wirtzburg und Würzburg

| Übergänge | t->d | u->e | u->o | i->e | a->e | a->o |
|---|---|---|---|---|---|---|
| absolute Häufigkeiten | 12 | 10 | 11 | 27 | 22 | 11 |
| Übergänge | i->ei | u->au | h->ha | t->dt | i->ie | a->ad |
| absolute Häufigkeiten | 18 | 29 | 19 | 28 | 13 | 27 |
| Übergänge | e->er | s->rs | s->sch | es->s | ne->n | re->r |
| absolute Häufigkeiten | 21 | 15 | 12 | 33 | 10 | 17 |
| Übergänge | ri->r | ma->me | in->en | es->ers | sh->rsh | ta->tad |
| absolute Häufigkeiten | 10 | 10 | 10 | 15 | 15 | 27 |
| Übergänge | at->adt | hu->hau | est->st | us->aus | esh->sh | ere->er |
| absolute Häufigkeiten | 27 | 19 | 11 | 20 | 21 | 12 |
| Übergänge | res->rs | | | | | |
| absolute Häufigkeiten | 13 | | | | | |

**Tab. 4.2:** Absolute Häufigkeiten der Übergänge mit einer Anzahl von mindestens 10

Nachdem nun die absoluten Anzahlen aller Übergänge extrahiert wurden, stellt sich die Frage, wie man daraus eine Gewichtung generiert. Sei im Folgenden $a$ die Ausgangszeichenkette, $b$ die Zielzeichenkette, $|a- > b|$ die Anzahl der Übergänge von $a$ nach $b$, $|a|$ die Anzahl, wie oft die Zeichenkette $a$ in eine andere Zeichenkette übergeht und $g(a- > b)$ das Gewicht des Übergangs $a- > b$. Die Gewichte werden durch folgende Formel berechnet:

$$g(a- > b) = 1 - \frac{|a- > b|}{|a|}$$

Es gibt mehrere Gründe, warum diese Formel sinnvoll ist:

1. Das Gewicht liegt zwischen 0 und 1. Wenn der Übergang mindestens einmal aufgetreten ist, so wird er als nicht mehr rein zufällig angesehen und sollte somit ein geringeres Gewicht als das Einfügen, Löschen und Austauschen von Zeichen haben, was unabhängig von den Zeichen ist.

2. Je häufiger ein bestimmter Übergang auftritt, desto geringer ist sein Gewicht.

3. Sei a->b der betrachtete Übergang. Dann kann die Gewichtung dieses Übergangs gering sein, obwohl der Übergang a->b häufig vorkommt. Nämlich dann, wenn a noch viel häufiger in eine andere Zeichenkette übergeht. Man kann also, wenn man die Zeichenkette a vor sich hat, mit einer nicht sehr hohen Wahrscheinlichkeit sagen, dass sie in b übergeht. Dementsprechend sollte sie ein hohes Gewicht haben.

4. Wenn a in allen ihren Fällen in b übergeht, so ist das Gewicht 0. Mann kann aus bisheriger Erfahrung also zu 100% sagen, dass wenn man ein a vor sich hat, es in ein b übergeht. Dementsprechend ist eine Gewichtung von 0 vertretbar.

# 5 Experimente

Nachdem nun ein Algorithmus entworfen und die dafür benötigten Gewichte extrahiert wurden, stellt sich natürlich die Frage, inwieweit dieser Algorithmus besser ist als die bereits bekannten. Aus diesem Grund werden im Folgenden einige Experimente durchgeführt, die den entworfenen Algorithmus (im Folgenden gewichteter Algorithmus genannt) mit dem Levenshtein-Algorithmus vergleichen. Dazu wird jeweils die gleiche Datenabank [Zim15], aus der auch die Gewichte extrahiert wurden, genutzt.

## 5.1 Evaluierung mit modernen Ortsnamen

Es wird nun einmal der entworfene Algorithmus zur Abstandbestimmung und einmal die Levenshtein-Distanz benutzt, wobei, wie in Kapitel 3 schon beschrieben, nur der moderne Stadtnamen als Vergleich herangezogen wird. Nun wird für jeden untersuchten historischen Stadtnamen der nächste Nachbar, oder die nächsten Nachbarn, falls mehrere den gleichen Abstand haben, gespeichert. Falls einer der nächsten Nachbarn, der, laut Datenbank, tatsächliche moderne Stadtname ist, so wird dies als Treffer, andernfalls als Fehlschlag, gewertet.

Die Durchführung des Experimentes liefert folgende Ergebnisse:

| | Treffer | Fehlschläge | Trefferquote |
|---|---|---|---|
| Levenshtein | 755 | 1044 | 41,97% |
| gewichtet | 816 | 983 | 45,36% |

Der gewichtete Algorithmus ist also etwas besser als die Levenshein-Distanz, jedoch sind beide Verfahren mit einer Trefferquote von unter 50% noch sehr unzufriedenstellend.

## 5.2 Evaluierung mit modernen und historischen Ortsnamen

Die Trefferquote des letzten Versuchs war eher gering und es wurde nach Analyse der Datenbank und auch anderer historischer Stadtnamen herausgefunden, dass viele Städte historische Stadtnamen haben, deren Lautbild sich sehr von dem des modernen Stadtnamens unterscheidet. So hat beispielsweise die Stadt Würzburg den historischen Stadtnamen Herbipolis, welcher aus dem lateinischen stammt. Hierbei lässt sich nur aufgrund vom Lautbild keine Zuordnung zu Würzburg finden. Deshalb wird nun von der vorherigen Einschränkung, den untersuchten Stadtnamen nur mit den modernen Städtenamen der Datenbank zu vergleichen, abgewichen. Stattdessen wird er, um einen nächsten Nachbarn zu finden, zusätzlich noch mit allen historischen Stadtnamen verglichen. Auch für

die Bestimmung der Gewichte wird so vorgegangen. Man könnte meinen, dass das durch den dadurch entstandenen Algorithmus definierte Abstandsmaß wieder der Definition einer Metrik entspricht. Jedoch ist es aufgrund der Formel für die Umwandlung der absoluten Anzahlen der Übergänge in die Gewichte (siehe Kapitel 4) möglich, dass zwei unterschiedliche Worte den Abstand 0 haben, was der positiven Definitheit widerspricht.

Es wird also unsere Algorithmus zusätzlich noch mit einem Thesaurus (siehe Abschnitt 2.4) verknüpft. Natürlich wird hierbei der untersuchte historische Stadtname ausgeschlossen, da dieser ansonsten immer der nächste Nachbar mit einem Abstand von 0 wäre. Die Durchführung wird als Treffer gewertet, wenn einer der nächsten Nachbarn (wie im vorherigen Versuch definiert) entweder der (laut Datenbank) moderne Stadtname des untersuchten historischen Stadtnamens ist, oder wenn er ein anderer historischer Stadtname des modernen Stadtnamens ist. Im zweiten Fall ist es möglich, daraus auf den modernen Stadtnamen zu schließen und dadurch die gesuchte Zuordnung durchführen.

Verglichen wird der gewichtete Algorithmus wiederum mit der Levenshtein-Distanz, wobei auch hier alle historischen Städtenamen als Vergleich herangezogen werden. Die Durchführung dieses Experimentes liefert folgende Ergebnisse (die Ergebnisse des ersten Experiments wurden der Übersicht halber mit dem Zusatz „(vorher)“ ergänzt):

| | Treffer | Fehlschläge | Trefferquote |
|---|---|---|---|
| Levenshtein | 1239 | 560 | 68,87% |
| gewichtet | 1366 | 433 | 75,93% |
| Levenshtein (vorher) | 755 | 1044 | 41,97% |
| gewichtet (vorher) | 816 | 983 | 45,36% |

Auch in diesem Experiment sieht man, dass der gewichtete Algorithmus etwas besser als die Levenshtein-Distanz ist. Jedoch hat sich die Trefferquote durch den zusätzlichen Vergleich mit anderen historischen Städtenamen um ungefähr 30% verbessert, weshalb nun mit dieser Methode weitergearbeitet wird.

## 5.3 Evaluierung mit Einschränkung der Treffermenge

Da die Levenshtein-Distanz nur die Gewichte 1 oder 0 hat, kann für den Abstand zweier Worte auch nur ein ganzzahliges Ergebnis erreicht werden. Beim gewichteten Algorithmus jedoch haben die Gewichte Werte, die im positiven rationalen Zahlenbereich liegen. Aus diesem Grund kamen wir zu der Annahme, dass die Berechnung des nächsten Nachbarn auf Basis der Levenshtein-Distanz sehr häufig mehrere Ergebnisse liefert, während der gewichtete Algorithmus wohl meistens nur eines liefert. Diese Hypothese wurde getestet und lieferte folgende Ergebnisse (gleiche Bezeichnungen wie im vorherigen Versuch):

| | Lev. | gewichtet | Lev. (vorher) | gewichtet (vorher) |
|---|---|---|---|---|
| durchschnittliche Anzahl nächster Nachbarn | 2,62 | 1,18 | 3,01 | 1,09 |

Dass der gewichtete Algorithmus trotz seiner weiten Spanne an Gewichten oftmals mehrere Ergebnisse liefert hat zwei Gründe. Zum einen gibt es manche Stadtnamen für unterschiedliche Städte (in der Datenbank) mehrfach, wodurch auch mehrere Treffer zustande kommen. Dies liegt nicht nur an der Datenbank an sich: Manche Stadtnamen sind mehrfach vergeben. So gibt es beispielsweise ein Erlenbach am Main und ein Erlenbach bei Marktheidenfeld. Beide werden jedoch häufig auch ohne den Zusatz als Erlenbach bezeichnet und würden somit auch so in einer Datenbank auftreten. Zum anderen umfasst die untersuchte Datenbank nicht alle deutschen Städte und ist demnach zwar einigermaßen groß, aber nicht sehr groß. Dadurch ist es gut möglich, dass manche Gewichte häufiger auftreten, was zu gleichen Abständen führen kann.

Die Suche nach dem nächsten Nachbarn mit der Levenshtein-Distanz liefert also zwei- bis dreimal so viele Ergebnisse wie mit dem gewichteten Algorithmus. Dementsprechend hat die Levenshtein-Distanz den Vorteil, mehr Versuche zum Treffen des gesuchten modernen Stadtnamens zu haben. Nun wird dieser Vorteil ausgeglichen, indem eine Durchführung nur als Treffer gewertet wird, wenn nur ein nächster Nachbar gefunden wird. Dies lieferte folgende Ergebnisse:

| | Treffer | Fehlschläge | Trefferquote |
|---|---|---|---|
| Levenshtein | 652 | 1147 | 36,24% |
| gewichtet | 1156 | 643 | 64,26% |
| Levenshtein (vorher) | 389 | 1410 | 21,62% |
| gewichtet (vorher) | 769 | 1030 | 42,75% |

Die Trefferquote ging bei allen Versionen nach unten, da alle Treffer, bei denen mehrere nächste Nachbarn gefunden wurden, nicht mehr als Treffer gezählt wurden. Auffällig ist jedoch, dass nun der gewichtete Algorithmus eine fast doppelt so hohe Trefferquote hat, wie bei der Nutzung der Levenshtein-Distanz. Der große Vorteil am gewichteten Algorithmus ist also, dass er meistens nur Treffer liefert, was den Aufwand für den Nutzer bei der semi-automatischen Zuordnung zu den modernen Stadtnamen eine große Erleichterung ist.

## 5.4 Kreuzvalidierung

Man könnte zu Recht anmerken, dass bei den bisherigen Experimenten immer die gleiche Menge an Daten dazu benutzt wurde, sowohl die Gewichte zu extrahieren, als auch den Algorithmus durchzuführen und die modernen Städte zuzuordnen. Aus diesem Grund wird nun eine Kreuzvalidierung durchgeführt. Hierbei wurde die eine Hälfte der modernen Städtenamen mit ihren historischen Namen zur Bestimmung der Gewichte verwendet und auf der anderen Hälfte wurden beide Verfahren angewandt. Die Aufteilung in zwei gleich große Hälften wurde zufällig ausgewählt. Bei beiden Verfahren wurde der Vergleich mit modernen und historischen Städtenamen und die Wertung als Treffer, wenn nur ein nächster Nachbar vorhanden ist (siehe vorherigen Experiment), ausgewählt. Um ein verlässliches Ergebnis zu erzielen, wurde das Experiment 10 mal durchgeführt. Im Folgenden sind die Ergebnisse zu sehen:

| | durchschn. Trefferq. | min. Trefferq. | max. Trefferq. |
|---|---|---|---|
| Levenshtein | 42,25% | 38,23% | 46,69% |
| gewichtet | 69,24% | 66,41% | 72,75% |

Durchgängig durch alle Versuche hat der gewichtete Algorithmus knapp 1,5-mal so viele Treffer erzielt, wie bei der Benutzung der Levenshtein-Distanz. Zwar ist hier der Unterschied zwischen beiden Verfahren nicht so groß, wie bei den Versuchen vorher, jedoch immer noch wesentlich. Es ist also validiert, dass die Ergebnisse bei den vorherigen Experimenten nicht nur deshalb zugunsten des gewichteten Algorithmus ausfielen, weil die Gewichte aus der gleichen Menge extrahiert wurden, die auch für den Algorithmus benutzt wurde.

# 6 Zusammenfassung und Ausblick

Zusammenfassend kann man sagen, dass der gewichtete Algorithmus in allen Experimenten, wenn manchmal auch knapp, besser abgeschnitten hat als bei der Benutzung der Levenshtein-Distanz. Besonders bei der Eindeutigkeit der Ergebnisse ist der gewichtete Algorithmus wesentlich stärker und schont den Nutzer eines etwaigen darauf aufbauenden Programmes, indem dieser nicht so häufig zwischen vielen Alternativen auswählen muss.

Ein Problem des gewichteten Algorithmus, das in dieser Bachelorarbeit aber nicht genauer untersucht wurde, ist, dass er ein Vielfaches der Rechenzeit des Algorithmus zur Berechnung der Levenshtein-Distanz. In diesem Bereich ist es durchaus möglich noch Optimierungen durchzuführen. Eine Möglichkeit, die auf alle Fälle in Betracht gezogen werden sollte ist, nicht alle möglichen Städtenamen der Datenbank in Betracht zu ziehen, sondern dies von der geographischen Lage abhängig zu machen. So weiß man meistens bei alten Landkarten, welche Koordinaten diese in etwa abdeckt. Man könnte also nur diejenigen Städtenamen der Datenbank durchsuchen, die in diesem abgedeckten Bereich liegen.

Zwar macht die Berechnung der Gewichte aus den absoluten Häufigkeiten der Übergänge, wie schon in Kapitel 4 erklärt, Sinn. Jedoch ist sie noch nicht optimal und es können noch eine Reihe an Experimenten in dieser Richtung durchgeführt werden. Gibt es beispielsweise den Übergang von a nach b einmal in der Datenbank und die Zeichenkette a geht auch nur einmal in eine andere Zeichenkette über, so ist das Gewicht dieses Übergangs 0. Selbst wenn dieser Übergang rein zufällig aufgetreten ist, was bei einem einmaligen Auftreten durchaus der Fall sein kann, hat er nun keine Kosten. Eine Möglichkeit dies zu umgehen ist, eine größere Datenbank zum Extrahieren der Daten zu benutzen. Eine Andere ist es, die Formel für die Berechnung der Gewichte anders zu wählen.

# Literaturverzeichnis


[Bud15] Benedikt Budig: Efficient Algorithms and User Interaction for Metadata Extraction from Historical Maps. In: *Proceedings of the first ACM SIGSPATIAL PhD Symposium*, 2015. To be published.

[Fis74] Robert A. Wagner; Micheal J. Fischer: The String-to-String Correction Problem. *Journal of the Association for Computing Machinery*, 21(1):168–173, 1974.

[Her86] Horst Herrlich: *Topologie I: Topologische Räume.* Heldermann Verlag Berlin, 1986. Kapitel 1.

[Lev65] Vladimir I. Levenshtein: Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1965.

[Lis11] Bertrand Lisbach: *Linguistisches Indentity Matching - Paradigmenwechsel in der Suche und im Abgleich von Personendaten.* Vieweg+Teubner Verlag, 2011. Seiten 77-128.

[Ukk85] Esko Ukkonen: Algorithms for approximate string matching. *Elsevier*, 64(1-3):100–118, January–March 1985.

[Wil05] Martin Wilz: Aspekte der Kodierung phonetischer Ähnlichkeiten in deutschen Eigennamen. Magisterarbeit, Universität zu Köln, Germany, 2005.

[Zim15] Marlene Antretter; Dirk Eller; Hannes Elstermann; Stefan Geissler; Simon Grüning; Sebastian Horn; Matthias Kohler; Kevin Kuck; Anna Lingnau; Alexandra Nozik; Jakob Odenwald; Felix Rieger; Christopher Schubert; Felix Wenze; Karin Zimmermann: Georeferencing of the „Lorscher Codex“, 2015. Heidelberg Research Data Repository [Distributor] V1 [Version].

# Erklärung

Hiermit versichere ich die vorliegende Abschlussarbeit selbstständig verfasst zu haben, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben, und die Arbeit bisher oder gleichzeitig keiner anderen Prüfungsbehörde unter Erlangung eines akademischen Grades vorgelegt zu haben.

Würzburg, den 20. Mai 2016

. . . . . .
Martin Becker